INTRODUCTION TO SENSOR WEB TECHNOLOGIES

Simon Jirka
Geospatial Sensing 2020
Virtual, 2020-08-30
**Motivation**

- More and more sensors are deployed
- Human observations
- Lots of different phenomena
  - Weather
  - Pollution
  - Biodiversity
  - Hydrological Data
  - Traffic
  - ...

HTTPS://52NORTH.ORG
Motivation
**Motivation**

- Different structures and elements of observations
- Sensor interfaces are very heterogeneous
- Need for harmonisation → interoperability
- Provide a layer on top of observation databases and low level sensor network technology → integration into web applications
Motivation

• Traditional services allow to
  • request maps (images)
    • Web Mapping Service
  • request (binary) raster data
    • Web Coverage Service
  • request vector data
    • Web Feature Service

• Lack of a generic framework for integration observation data into spatial data infrastructures
Motivation
**Motivation**

- GIS
- Web Client
- Desktop System

OGC/ISO Observations & Measurements
- OGC SensorML
- OGC Sensor Observation Service

- SOS Observation Network 1
- SOS Observation Network 2
- SOS Observation Network 3
SWE BUILDING BLOCKS

Sensor Web Enablement

Data Models and Encodings
- SWE Common
- O&M
- SensorML
- WaterML 2.0
- TimeSeriesML

Interfaces
- SensorThings API
- SWE Service Model
- SOS
- SPS
- (OGC Pub/Sub)
Observation & Measurements

- Common ISO (ISO 19156:2011) and OGC (OGC 10-004r3) Standard
  - Geographic information — Observations and measurements
  - ISO 19156:2011
  - OGC 10-004r3
- Split into an abstract specification and an XML implementation
- Defines a conceptual model for observations and for features involved in sampling when making observations
A sensor mounted on a buoy located at 53.8112°N 9.3632°E measures the water temperature of the River Elbe as 6°C on 2020-01-20 at 17:45. It was validated on 2020-01-20 at 18:00.
**Observation**

- At a specific time instant or period, a process assigns a value to a phenomenon that is a property of a feature.
PHENOMENON TIME

• A time instant or time period that describes the time the result applies to the property of the feature of interest

• When analyzing data this is timestamp the you typically associate with the value
**Result Time**

- A time instant that describes when the result became available
- Most of the time identical to the phenomenon time
- Typical use cases:
  - Post processing steps
  - Simulations
  - Specimen retrieval vs. analysis
**Valid Time**

- Describes the time period during which the result is intended to be used
- Typical use case:
  - The output of a forecasting model is only valid until the next model run
- Optional
**Observation Process / Procedure**

- Abstract entity that represents the observation process and produces the observation result
- O&M doesn’t make any assumptions or restrictions
- Can be anything:
  - Instrument or Sensor
  - Human Observer
  - Process or Algorithm
  - Simulation
- \( \rightarrow \) SensorML
FEATURE OF INTEREST

• The feature of interest is the subject of the observation and carries the observed property
• The real-world object whose properties are under observation
• A feature intended to sample the real-world object

Examples
• Measurement station
• River
• River basin
• ...
**PHENOMENON / OBSERVED PROPERTY**

- The phenomenon is a property of the observed feature for which the result of the observation provides an estimate.
- May be organized into hierarchies or ontologies and managed in catalogues.
- Should support semantica or thematic classification.
- Examples:
  - “temperature”
  - `http://vocab.nerc.ac.uk/collection/P01/current/CATAVT10/`
**PARAMETER**

- Can be used to embed arbitrary information into the observation that is not bound to the feature of interest or observation procedure.

- Example use case:
  - Trajectories with the complete track as the feature of interest can describe the current position along the track.
  - Measurements along a water column can describe the depth as a parameter.
RESULT

• The value generated by the procedure and assigned to the observed property of the feature of interest
• O&M doesn’t restrict the type the value
• But it defines some basic observation types
**ENCODINGS**

- Plain O&M 2.0
- OGC WaterML 2.0
- OGC TimeseriesML 1.0
- INSPIRE D2.9 Guidelines for the use of O&M
PLAIN O&M 2.0 – GENERAL XML

```xml
<om:OM_Observation>
  <gml:description> <!-- description of the observation (optional) -->
  </gml:description>
  <gml:identifier> <!-- identifier, used by GetObservationById (optional) -->
  </gml:identifier>
  <gml:name> <!-- name of the observation (optional) -->
  </gml:name>
  <om:type> <!-- the type of the observation (optional) -->
  </om:type>
  <om:metadata> <!-- some metadata for the observation (optional) -->
  </om:metadata>
  <om:relatedObservation> <!-- a related observation, e.g. link to the raw values (optional) -->
  </om:relatedObservation>
  <om:phenomenonTime> <!-- time when the measurement takes place -->
  </om:phenomenonTime>
  <om:resultTime> <!-- time when the observation became available -->
  </om:resultTime>
  <validTime> <!-- time when the observation is valid, e.g. for forecast observations (optional) -->
  </validTime>
  <om:procedure> <!-- the procedure that creates the observation -->
  </om:procedure>
  <om:parameter> <!-- some additional parameter (optional) -->
  </om:parameter>
  <om:observedProperty> <!-- the observed phenomenon -->
  </om:observedProperty>
  <om:featureOfInterest> <!-- the location where the measurement takes place -->
  </om:featureOfInterest>
  <om:resultQuality> <!-- information of the result quality, complex (optional) -->
  </om:resultQuality>
  <om:result> <!-- the result of the observation -->
  </om:result>
</om:OM_Observation>
```
**WATERML 2.0**

- Data model format for exchanging hydrological data
- Based on XML
- Specialization of a more generic standard: ISO/OGC Observations and Measurement 2.0
WATERML 2.0

<result>
  <wml2:MeasurementTimeseries xlink:id="K1.Ts.1">
    <wml2:metadata>
      <wml2:MeasurementTimeseriesMetadata>
        <wml2:temporalExtent xlink:href="#K1.Wm1.pTime.1"/>
        <wml2:isCumulative False="true"/>
      </wml2:MeasurementTimeseriesMetadata>
    </wml2:metadata>
    <wml2:defaultPointMetadata>
      <wml2:DefaultTVPMeasurementMetadata>
        <wml2: uom code="#m/s"/>
        <wml2: interpolationType xlink:href="http://www.openGIS.net/def/waterml/2.0/interpolationType/AverageTemp" xlink:title="Average In Proceeding Interval"/>
      </wml2:DefaultTVPMeasurementMetadata>
    </wml2:defaultPointMetadata>
    <wml2:point>
      <wml2:MeasurementTVP>
        <wml2:value xlink:href="#K1.Wm1.pTime.1" xlink:title=""/>
      </wml2:MeasurementTVP>
    </wml2:point>
    <wml2:point>
      <wml2:MeasurementTVP>
        <wml2:value xlink:href="#K1.Wm1.pTime.1" xlink:title=""/>
      </wml2:MeasurementTVP>
    </wml2:point>
  </wml2:MeasurementTimeseries>
</result>
**TimeseriesML 1.0**

- TimeseriesML → standard derived from WaterML 2.0
- Data model format for exchanging timeseries data
- Based on XML
INSPIRE - D2.9 GUIDELINES FOR THE USE OF O&M

- INSPIRE – Spatial Data Infrastructure of the European Commission
- Several themes with measured, modelled and simulated data
- Technical Guidelines for data interoperability
  - Services (e.g. SOS as Download service)
  - Data formats
- D2.9 - Guideline for observation data
  - Based on OGC O&M
**POINTOBSERVATION**

- Similar to a simple O&M observation (e.g. measurement)
- Result additionally contains a point (location)
**POINTTIME SERIES OBSERVATION**

- Result is a WaterML 2.0 Timeseries
**MultiPointObservation**

- Measuring the same parameter at the same time at different location

Diagram:
- BBox containing stations
- Feature
- featureOfInterest
- observedProperty
- Process
- Values
- Thermometer Type XY

Domain: MultiPoint(52.1 6.89, 52.0 7.15, 52.2 7.52, 51.1 6.63, 51.1 7.32)
Range (°C): 10.5 11.0 10.9 10.7 11.2
**Profile Observation**

- Measuring values at varying depths along a water column.
TRAJECTORY OBSERVATION

- Result is an extension of WaterML 2.0 Timeseries with a TimeLocationValuePair.
SWE Services

- OGC Sensor Observation Service (SOS)
- OGC SensorThings API (STA) Part 1: Sensing
OGC SENSOR OBSERVATION SERVICE (SOS)
SOS Overview

- Access to sensor data
- Consistent interface and data format for all kinds of sensors
- Returns O&M
  - Contrast to WFS: no generic schemas
  - Interoperability
  - A priori-knowledge
- SOS 1.0
- SOS 2.0 → latest version of the standard
SOS Introduction

• Pull-based access to observations
• Mediator between:
  • client ⇔ data archive / simulation / real-time sensor system
• Hides the heterogeneous structure of proprietary sensor data formats and protocols
• Data formats: O&M and SensorML
  • Versions: 1.0 and 2.0
SOS 2.0 Model
SOS - Core

• GetCapabilities operation
  • Returns a metadata document describing the SOS instance, e.g.
    • Supported operations
    • Spatial, temporal and thematic content → offerings
  • Mandatory for every OGC Web Service
  • Structure and syntax from OWS Web Services Common specification
**GetCapabilities**

- Examples based on XML implementation:
  - Request: [http://schemas.opengis.net/sos/2.0/examples/core/GetCapabilities1.xml](http://schemas.opengis.net/sos/2.0/examples/core/GetCapabilities1.xml)
  - Response: [http://schemas.opengis.net/sos/2.0/examples/core/GetCapabilities1_response.xml](http://schemas.opengis.net/sos/2.0/examples/core/GetCapabilities1_response.xml)

- Example request encoded as KVP:
  - `http://hostname:port/path?service=SOS&request=GetCapabilities&AcceptVersions=2.0.0`
**GetObservation**

- GetObservation operation
  - Returns sensor data
  - Response format is usually a O&M-document
  - Supports comprehensive filter criteria
    - Feature of interest
    - Temporal filters
    - Spatial filters
    - Observed property
    - Sensor/procedure
    - Offerings
**DESCRIBE SENSOR**

- **Describe Sensor operation**
  - Returns detailed metadata of sensors and measurement processes
  - Response format is usually SensorML
  - Query parameters
    - Sensor/procedure id
    - Temporal filter (optional)
DESCRIBESENSOR

• Example based on XML implementation:
  • Request: http://schemas.opengis.net/sos/2.0/examples/SOAP/DescribeSensor1_SOAP.xml

• Example request encoded as KVP:
  • http://www.myserver.org:port/path?
    service=SOS&version=2.0.0&
    request=DescribeSensor&
    procedure=urn:ogc:object:Sensor:
      MyOrg:thermometer1&
    procedureDescriptionFormat=
      http://www.opengis.net/sensorML/1.0.1&
    validTime=2010-01-01T18:31:42Z
SOS 2.0 MODEL
SOS 2.0 MODEL

Extensions
- Transactional
- Result Handling
- Enhanced Operations
- Binding

Profiles
- Spatial Filtering

SOS Core

Basis
- OWS Common 1.1.0 [OGC 06-121r3]
- SWE Service Model 2.0 [OGC 09-001]
- Filter Encoding [ISO 19143:2010]
- O&M 2.0 [ISO 19156:2010]
SOS 2.0 Model
SOS 2.0 Model
Spatial Filtering Profile

- O&M XML – SpatialObservation
  - om:parameter with sampling location
    ```xml
    <om:parameter>
      <om:NamedValue>
        <om:name xlink:href="http://www.opengis.net/def/param-name/OGCOM/2.0/samplingGeometry"/>
        <om:value>
          <gml:Point gml:id="SamplingPoint">
            <gml:pos srsName="urn:ogc:def:crs:EPSG::4326">52.9 7.52</gml:pos>
          </gml:Point>
        </om:value>
      </om:NamedValue>
    </om:parameter>
    ```

- Filtering with valueReference value:
  - http://www.opengis.net/req/omxml/2.0/data/samplingGeometry
BINDINGS

• Official bindings
  • KVP
  • SOAP

• 52N SOS additionally supported bindings
  • POX (official only for SOS 1.0.0)
  • JSON (52N SOS, not yet specified)
OGC SensorThings API (STA) Part 1: Sensing
STA - Overview

• OGC SensorThings API
• Complementary specification to enhance the OGC SWE framework for Internet of Things applications
• Simplified approach
• Based on REST and JSON
  ~ REST binding for SOS functionalities
  ~ JSON binding for the O&M model
• Two parts
  • Data access (Part 1: Sensing)
  • Sensor tasking
• CRUD
## STA - Entities

<table>
<thead>
<tr>
<th>SensorThings API Entities</th>
<th>O&amp;M 2.0 Concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thing (and Locations, HistoricalLocations)</td>
<td>-</td>
</tr>
<tr>
<td>Datastream</td>
<td>-</td>
</tr>
<tr>
<td>Sensor</td>
<td>Procedure</td>
</tr>
<tr>
<td>Observation</td>
<td>Observation</td>
</tr>
<tr>
<td>Observed Property</td>
<td>Observed Property</td>
</tr>
<tr>
<td>FeatureOfInterest</td>
<td>FeatureOfInterest</td>
</tr>
</tbody>
</table>
STA - ENTITIES

• Thing
  • IoT device

• Location
  • The location where the Thing is currently placed

• HistoricalLocation
  • Locations where the Thing was placed before

• Datastream
  • Groups observations for the same Thing, ObservedProperty and Sensor
  • Provides some metadata (unit, phenomenonTime, observedArea, …)
STA - Entities

- Sensor
  - The instrument that observes a phenomenon
- ObservedProperty
  - The phenomenon of the observation
- FeatureOfInterest
  - The observed location
  - Can be the Thing-Location
  - A room, a geographical area, …
STA -Entities

• Observation
  • The measurement
  • Values
    • quantity, count, text, category, boolean

• MultiDatastream Extension
  • Similar to O&M ComplexObservation

• DataArray Extension
  • Similar to O&M SweDataArrayObservation
STA - QUERYING

- http://example.org/v1.0/Things
- http://example.org/v1.0/Things(1)
- http://example.org/v1.0/Things(1)/Locations
- http://example.org/v1.0/Things(1)/description
- http://example.org/v1.0/Things(1)/description/$value
STA - FILTERING

- \$skip
  - number for the queries items that shall be excluded from the result
- \$top
  - limit on the number of items returned
- \$expand
  - Represent entity inline
- \$select
  - Return only the selected properties
STA - FILTERING

• $count
  • total count of items within a collection matching the request

• $orderby
  • The order in which the items should be returned

• $filter
  • Filter the resulting items
  • Comparison, Logical and Arithmetic Operators
  • String, Date, Math and Geospatial functions
STA - MQTT

• MQTT - Message Queuing Telemetry Transport

• Publish
  • Observations

• Subscribe
  • Receive notification for updated entities
  • Examples
    • Datastreams(1)/Observations (new observation added)
    • Datastreams(1) (property changed)
    • Datastreams(1)/observedArea (value of property changed)
STA – MQTT Publish

MQTT Subscribe to Datastreams(id)/Observations

Observation pushed via MQTT

Observation pushed via MQTT

Create(Entity)

Observation pushed via MQTT

HTTP POST Observation

HTTP POST Observation

MQTT Publish Observation

Sensor

Sensor

SensorThings
STA – MQTT SUBSCRIBE
SENSORTHINGS API

• Demo Servers:
  • https://cos4cloud.demo.52north.org/sta/
  • https://aqsens.52north.org/data/reference/sta/
**SensorThings API - Root**

• /  
  • Root query

• Click the URL of the entity you are interested in.
SENSORTHINGS API - THING

- /Things
  - Get all Things
- /Things(1)
  - Get the Thing with ID „1“
- Example

```json
{
  "@iot.id": "ID_163",
  "@iot.selfLink": "https://aqsens.52north.org/data/reference/sta/Things(ID_163)",
  "name": "Rv 4, Aker sykehus",
  "description": "null",
  "properties": "null",
  "Datastreams@iot.navigationLink": "https://aqsens.52north.org/data/reference/sta/Things(ID_163)/Datastreams",
  "Locations@iot.navigationLink": "https://aqsens.52north.org/data/reference/sta/Things(ID_163)/Locations",
  "HistoricalLocations@iot.navigationLink": "https://aqsens.52north.org/data/reference/sta/Things(ID_163)/HistoricalLocations"
}
```
**SensorThings API - Location**

- `/Locations`
  - Get all Locations
- `/Locations(1)`
  - Get the Location with ID „1“

**Example**
- [https://aqsens.52north.org/data/reference/sta/Things(ID_163)/Locations](https://aqsens.52north.org/data/reference/sta/Things(ID_163)/Locations)
**SENSOR THINGS API - HISTORICAL LOCATION**

- / HistoricalLocations
  - Get all HistoricalLocations
- / HistoricalLocations(1)
  - Get the HistoricalLocations with ID „1“

```json
{
   "@iot.id": "02098019-e8c2-494e-a25f-5dbb10b39747",
   "@iot.selfLink": "https://aqsens.52north.org/data/reference/sta/HistoricalLocations(02098019-e8c2-494e-a25f-5dbb10b39747)",
   "time": "2020-06-05T16:44:36.537Z",
   "Locations@iot.navigationLink": "https://aqsens.52north.org/data/reference/sta/HistoricalLocations(02098019-e8c2-494e-a25f-5dbb10b39747)/Locations",
   "Thing@iot.navigationLink": "https://aqsens.52north.org/data/reference/sta/HistoricalLocations(02098019-e8c2-494e-a25f-5dbb10b39747)/Thing"
}
```
SENSORTHINGS API - DATASTREAM

• / Datastreams
  • Get all Datastreams

• / Datastreams(1)
  • Get the Datastreams with ID „1“

• Example
  • https://aqsens.52north.org/data/reference/sta/Datastreams(019722ba-89cf-4bf8-88d8-47614d43b34b)
SensorThings API - Datastream

- Links to Thing, ObservedProperty, Sensor and Observations

```
{  
    "@iot.id": "3d2db226-1580-4e6b-a372-a8bf4fe5796a",  
    "@iot.selfLink": "https://aqsens.52north.org/data/reference/sta/Datastreams(3d2db226-1580-4e6b-a372-a8bf4fe5796a)",
    "name": "Rv 4, Aker sykehus_Station Rv 4, Aker sykehus (Stor-Oslo), PM2.5_PM2.5",
    "description": "Datastream for Thing 'Rv 4, Aker sykehus' and Sensor 'Station Rv 4, Aker sykehus (Stor-Oslo), PM2.5' and ObservedProperty 'PM2.5'.",
    "observationType": "http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement",

    "unitOfMeasurement": {  
        "name": "null",
        "symbol": "µg/m³",
        "definition": "null"
    },

    "observedArea": "null",
    "resultTime": "2020-05-01T02:00:00.000Z/2020-06-08T13:00:00.000Z",
    "phenomenonTime": "2020-05-01T02:00:00.000Z/2020-06-08T13:00:00.000Z",

    "ObservedProperty@iot.navigationLink": "https://aqsens.52north.org/data/reference/sta/Datastreams(3d2db226-1580-4e6b-a372-a8bf4fe5796a)/ObservedProperty",
    "Observations@iot.navigationLink": "https://aqsens.52north.org/data/reference/sta/Datastreams(3d2db226-1580-4e6b-a372-a8bf4fe5796a)/Observations",

    "Thing@iot.navigationLink": "https://aqsens.52north.org/data/reference/sta/Datastreams(3d2db226-1580-4e6b-a372-a8bf4fe5796a)/Thing",
    "Sensor@iot.navigationLink": "https://aqsens.52north.org/data/reference/sta/Datastreams(3d2db226-1580-4e6b-a372-a8bf4fe5796a)/Sensor"
}
```
SENSOR THINGS API - SENSOR

• / Sensors
  • Get all Sensors
• / Sensors(1)
  • Get the Sensors with ID „1“
• Example
  • https://aqsens.52north.org/data/reference/sta/Sensors(station.stor-oslo.rv_4_aker_sykehus.pm2.5)
"iot.id": "station.stor-oslo.rv_4_aaker_sykehus.pm2.5",
"iot.selfLink": "https://aqsens.52north.org/data/reference/sta/Sensors(station.stor-oslo.rv_4_aaker_sykehus.pm2.5)",
"name": "Station Rv 4, Aker sykehus (Stor-Oslo), PM2.5",
"description": "null",
"encodingType": "http://www.opengis.net/doc/IS/SensorML/2.0",
"metadata": "<sml:PhysicalComponent xmlns:sml="http://www.opengis.net/sensorml/2.0" xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:swe="http://www.opengis.net/swe/2.0" xmlns:xlink="http://www.w3.org/1999/xlink" gml:id="gmlid_1">

<sml:Identification>
  <sml:IdentifierList>
    <sml:Identifier>
      <sml:TermDefinition
        definition="urn:ogc:def:identifier:OGC:1.0:longName">
        <sml:label>longName</sml:label>
        <sml:value>Station Rv 4, Aker sykehus (Stor-Oslo), PM2.5</sml:value>
      </sml:Identifier>
    </sml:IdentifierList>
  </sml:Identification>

<sml:IdentifierList>
  <sml:Identifier>
    <sml:TermDefinition
      definition="urn:ogc:def:identifier:OGC:1.0:shortName">
      <sml:label>shortName</sml:label>
      <sml:value>Station Rv 4, Aker sykehus (Stor-Oslo), PM2.5</sml:value>
    </sml:Identifier>
  </sml:IdentifierList>
</sml:PhysicalComponent>

<swe:Boolean definition="institu">
  <swe:value>true</swe:value>
</swe:Boolean>

<swe:Boolean definition="mobile">
  <swe:value>false</swe:value>
</swe:Boolean>

<swe:Quantity definition="PM2.5">
  <swe:Code definition="μg/m³">
    <swe:Quantity>
      <swe:value>0</swe:value>
    </swe:Quantity>
  </swe:Code>
</swe:Quantity>

"Datastreams@iot.navigationLink": "https://aqsens.52north.org/data/reference/sta/Sensors(station.stor-oslo.rv_4_aaker_sykehus.pm2.5)/Datastreams"
SENSOR THINGS API - OBSERVED PROPERTY

- / ObservedProperties
  - Get all ObservedProperties
- / ObservedProperties(1)
  - Get the ObservedProperties with ID „1“

Example

- [https://aqsens.52north.org/data/reference/sta/ObservedProperties(PM2.5)](https://aqsens.52north.org/data/reference/sta/ObservedProperties(PM2.5))

```json
{
    "@iot.id": "PM2.5",
    "@iot.selfLink": "https://aqsens.52north.org/data/reference/sta/ObservedProperties(PM2.5)",
    "name": "PM2.5",
    "description": null,
    "definition": "PM2.5",
    "Datastreams@iot.navigationLink": "https://aqsens.52north.org/data/reference/sta/ObservedProperties(PM2.5)/Datastreams"
}
```
**SENSORTHINGS API - OBSERVATION**

- / Observations
  - Get all Observations
- / Observations(1)
  - Get the Observations with ID „1“
- Example
  - [https://aqsens.52north.org/data/reference/sta/Observations(0038bb98-d608-4f76-ab54-1efc86dec89a)](https://aqsens.52north.org/data/reference/sta/Observations(0038bb98-d608-4f76-ab54-1efc86dec89a))

```json
{
  "iot.id": "42772",
  "iot.selfLink": "https://aqsens.52north.org/data/reference/sta/Observations(42772)",
  "result": "1.9769960000",
  "resultTime": null,
  "phenomenonTime": "2020-05-01T02:00:00.000Z",
  "resultQuality": null,
  "validTime": null,
  "parameters": [],
  "DataStream@iot.navigationLink": "https://aqsens.52north.org/data/reference/sta/Observations(42772)/DataStream",
  "FeatureOfInterest@iot.navigationLink": "https://aqsens.52north.org/data/reference/sta/Observations(42772)/FeatureOfInterest"
}
```
SENSOR THINGS API – OBSERVATION FOR DATASTREAM

- https://aqsens.52north.org/data/reference/sta/Datastreams(019722ba-89cf-4bf8-88d8-47614d43b34b)/Observations?$top=2
SensorThings API - FeatureOfInterest

• / FeaturesOfInterest
  • Get all FeaturesOfInterest

• / FeatureOfInterest(1)
  • Get the FeaturesOfInterest with ID "1"

• Example
  • https://aqsens.52north.org/data/reference/sta/FeaturesOfInterest(ID_163)
SENSORTHINGS API - FILTER

• $top
  • https://aqsens.52north.org/data/reference/sta/FeaturesOfInterest?$top=3
• $skip
  • https://aqsens.52north.org/data/reference/sta/FeaturesOfInterest?$skip=3
• $top and $skip
  • https://aqsens.52north.org/data/reference/sta/FeaturesOfInterest?$skip=3&$top=3
**SENSORTHINGS API - FILTER**

- `$orderby`

- `$select`
  - Select only the name property
  - Name and description
Typical Set Ups

Existing database via Hibernate

- Sensor Web Client
  - GetObservation
  - O&M
- Sensor Web Server
  - Hibernate
- Database
  (e.g. PostgreSQL, MySQL, Oracle)
**Typical Set Ups**

Existing database via Database Views

- **Sensor Web Client**
  - GetObservation
  - O&M

- **Sensor Web Server**

- **Database Views**

- **Database**
  (e.g. PostgreSQL, MySQL, Oracle)
**Typical Set Ups**

Existing database with customised SOS

![Diagram of Sensor Web Client, Sensor Web Server, and Database with GetObservation and O&M interactions.]
**Typical Set Ups**

MQTT Feeding

```
Sensor Web Client

GetObservation       O&M

Sensor Web Server

MQTT Broker

Device

Default DB
```
TYPICAL SET UPS

Default SOS database, standardised feeding (transactional operations)

Sensor Web Client

GetObservation

O&M

Sensor Web Server

Data Feeder, e.g. FME

InsertObservation (O&M) or InsertResult

Input Data, e.g. CSV

Default DB
TYPICAL SET UPS

Default SOS database, data feeding by SQL script
**Typical Set Ups**

SOS as proxy for proprietary data access service

![Diagram of typical setup with Sensor Web Client, Sensor Web Server, and Proprietary Service with GetObservation and O&M arrows between them.](image-url)
QUESTIONS!