SDIs for the Internet of Things The integration of spatio-temporal data in INSPIRE

Alexander Kotsev, Katharina Schleidt, Hylke van Der Schaaf, Sylvain Grellet, Simon Jirka, Mickaël Beaufils, Ilkka Rinne

SDIs for the Internet of Things

The integration of spatio-temporal data in INSPIRE

Alexander Kotsev, Katharina Schleidt, Hylke van der Schaaf, Sylvain Grellet, Simon Jirka, Mickaël Beaufils, Ilkka Rinne

Workshop outline

Part 1. Scene setter

- Context
- Available guidance
- Standardisation initiatives

Part 2. O&M clinic

- O&M Simple features
- FROST-Server
- 52North SOS & STA
- AIT SOS

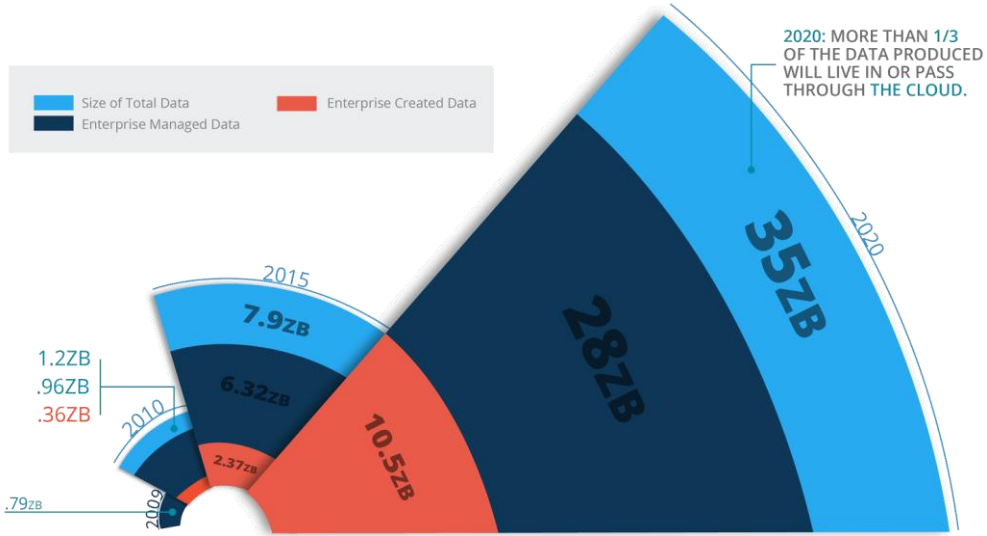
Part 1. Scene-setter

Context

- The “Data Revolution”
- Exponential growth
  - Size of digital universe doubles every two years
  - 50-fold growth (2010-2020)
- Velocity
- Multiple channels (Variety)
- Noise/Signal ratio
- New actors
  - Private sector
  - Citizens
  - Public sector (open data)

New technology

- IoT
- Cloud, etc. etc. etc.



Observations and the EU policy agenda

- "Building a European Data Economy"
  - COM(2017) 9 final & SWD(2017) 2 final
  - The role of INSPIRE is acknowledged
  - Goes beyond the public sector and considers
    - Raw machine-generated (IoT) content
    - Private data
    - Industrial data platforms
    - Citizens

Machine generated data are 'spatial data' as sensors usually also transmit their direct or indirect position (location) together with their measurement. (page 16)

- INSPIRE is recognised as a best practice

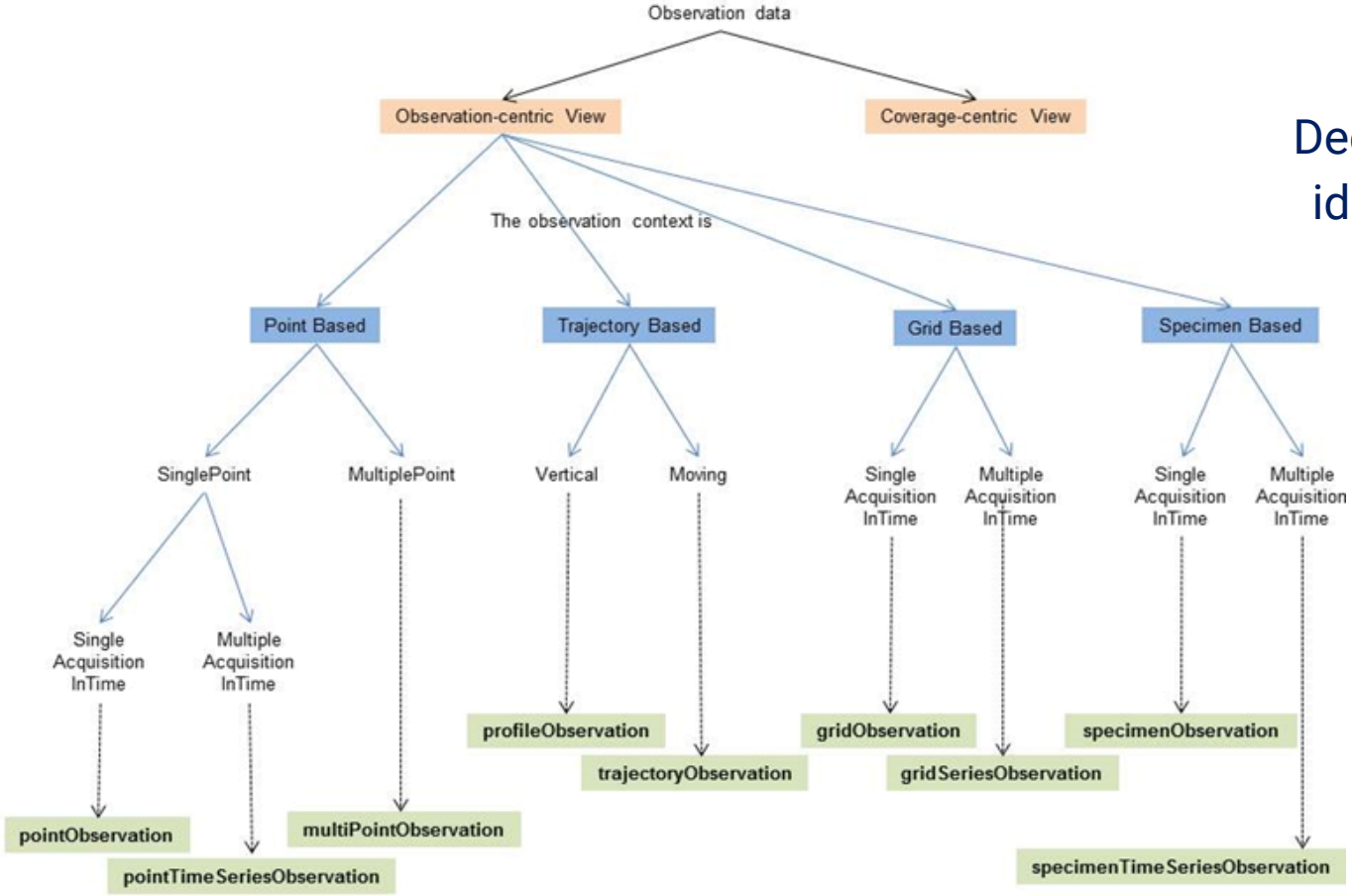
Spatio-temporal data in INSPIRE

- Guidelines for the use of Observations & Measurements and Sensor Web Enablement-related standards in INSPIRE (D2.9)
- Technical Guidance for implementing download services using the OGC Sensor Observation Service and ISO 19143 Filter Encoding

Officially endorsed as INSPIRE Technical Guidance documents:

- Guidelines for O&M (D2.9): http://inspire.ec.europa.eu/id/document/tg/d2.9-o&m-swe
- Technical Guidance SOS as a download service: http://inspire.ec.europa.eu/id/document/tg/download-sos

INSPIRE Guidance Documents on O&M and SOS Guidelines for O&M (D2.9)



Decision Tree for simple identification of correct observational model

Challenges 1/3

- Most IoT devices are constrained (low computational capability)
- They face issues with
  - Computational capabilities
  - Limited connectivity
- Use-cases are data-intensive
- Asynchronous transactions are needed
- Data platforms are often proprietary and lock users in

Challenges 2/3

Implementation issues

- TimeLocationValueTriple Encoding
- Standardized usage of INSPIRE BASE Types
- Referencing between Features
- Codelists
- Geometry Encoding with SF Types

Challenges 3/3

Data Model Issues

- Out-of-Band encoding and external result formats
- Suitability of Observation Types
  - Fitness for purpose
  - Constraint errors
- Coverage Issues
  - Unclear which coverage version to be utilized
  - Provision via WCS
- Observation ID – what is a “dataset”
- lifeCycleInfo – should this be added?
- Geometry per OperationalActivityPeriod

Future directions. Action 2017.2

Tasks:

1. Encoding rule for GeoJSON (as a first example) → Good Practice document 1
2. Generic rules / approaches for flattening of data models (useful for a number of alternative encodings) → Good Practice document 2
3. Procedure for proposing and endorsing additional encodings

Future directions. Action 2017.3

Tasks:

1. **Collect issues** with the consumption of data (incl. data itself)

2. Conduct a **study on the usability** of the test INSPIRE datasets identified

Tools (OGR/GDAL), desktop and web clients (e.g. Quantum GIS, ESRI ArcGIS for Desktop, LeafletJS, OpenLayers) and analytical or ETL tools for data processing (e.g. HALE, FME, R)

3. Organise **an event with software vendors**

Prioritise tools and specific functionalities

4. Investigate good practices for **direct use of data** based on its metadata) from national and INSPIRE metadata and catalogues.

Future directions. collaborative workspaces

- All available on GitHub
  - Openness
  - Agility
  - Implementations first
  - Exhaustiveness *versus* Simplicity

- New tools
  - GitHub
  - Hackathons

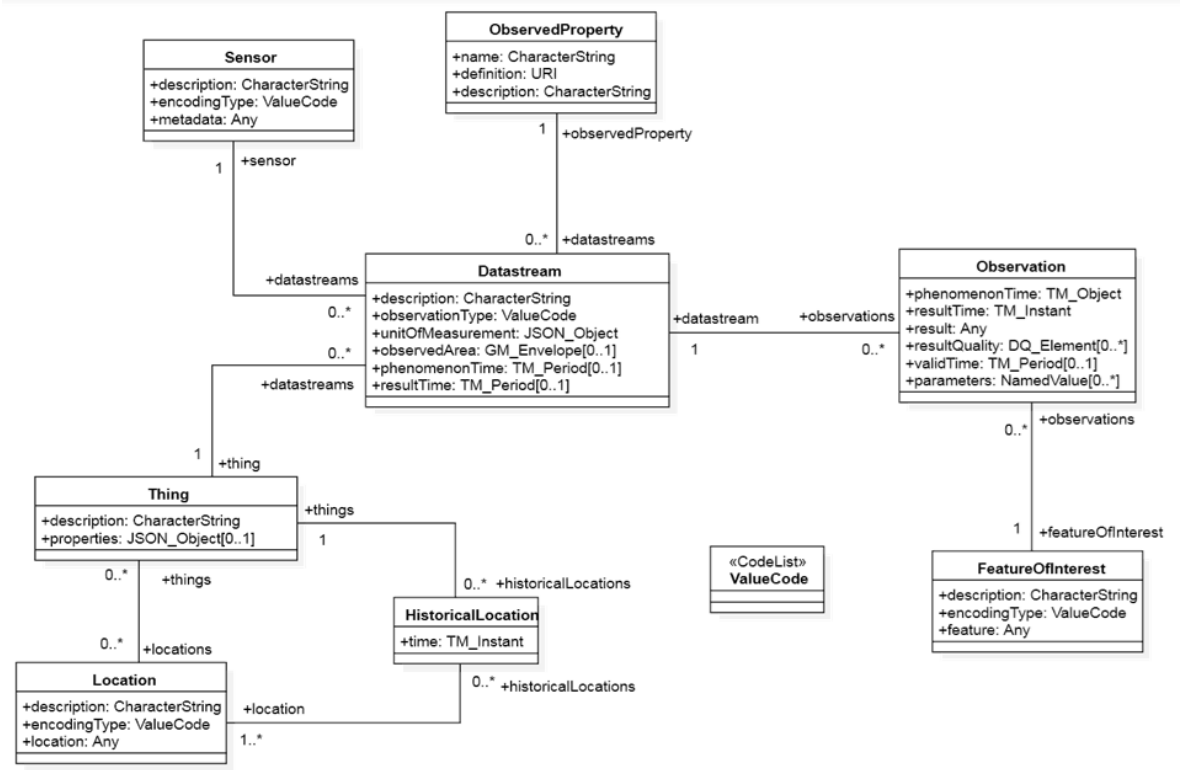
Emerging standards

SensorThings API

- New OGC standard
- Developer friendly approach
- Features
  - RESTful interface
  - JSON for data encoding
  - Support for MQTT

OGC®
Making location count.

SensorThings API: The data model



SensorThings API: Restful service

Base URI provides entry point to all Classes

http://.../v1.0/: all Classes contained in STA

URIs follow data model graph:

http://.../v1.0/Things: all Things

http://.../v1.0/Things(1): Thing with the id 1

http://.../v1.0/Things(1)/Locations: all Locations associated with Thing 1

http://.../v1.0/Things(1)/Locations(3): Location 3 associated with Thing

SensorThings API: Restful service 2

Use **expand** to include nested classes:

http://.../v1.0/Thing?$expand=Datastreams/ObservedProperty

Use **skip** to page through the data

http://.../v1.0/Datastreams(517)/Observations?$skip=200

Use **filter** to select parts of the data

http://.../v1.0/Datastreams(308)/Observations?$filter=phenomenonTime lt 2017-12-02T14:37:01.000Z

Message Queuing Telemetry Transport – MQTT

Updates on all Things, Observations, etc:

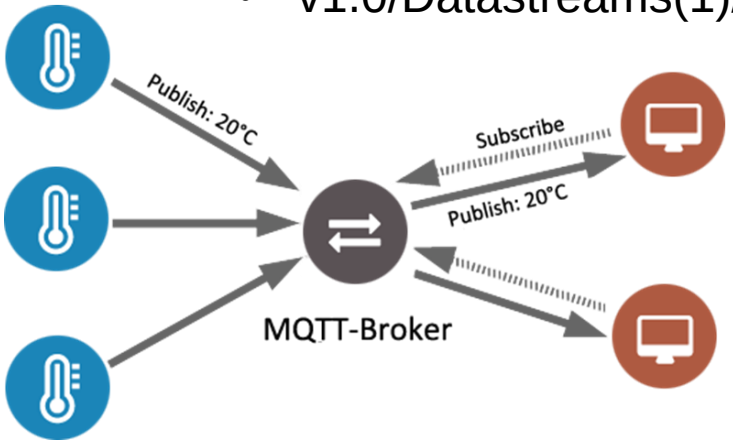
- v1.0/Things or v1.0/Observations

Updates on the Observations of a given Datastream:

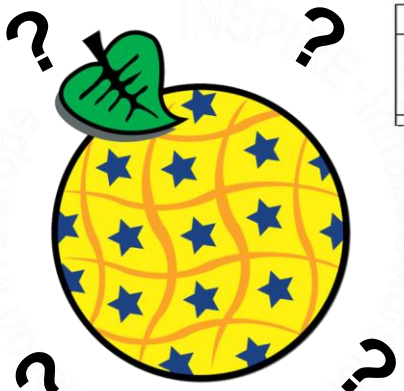
- v1.0/Datastreams(1)/Observations

Only receive certain fields:

- v1.0/Datastreams(1)/Observations?$select=phenomenonTime,result

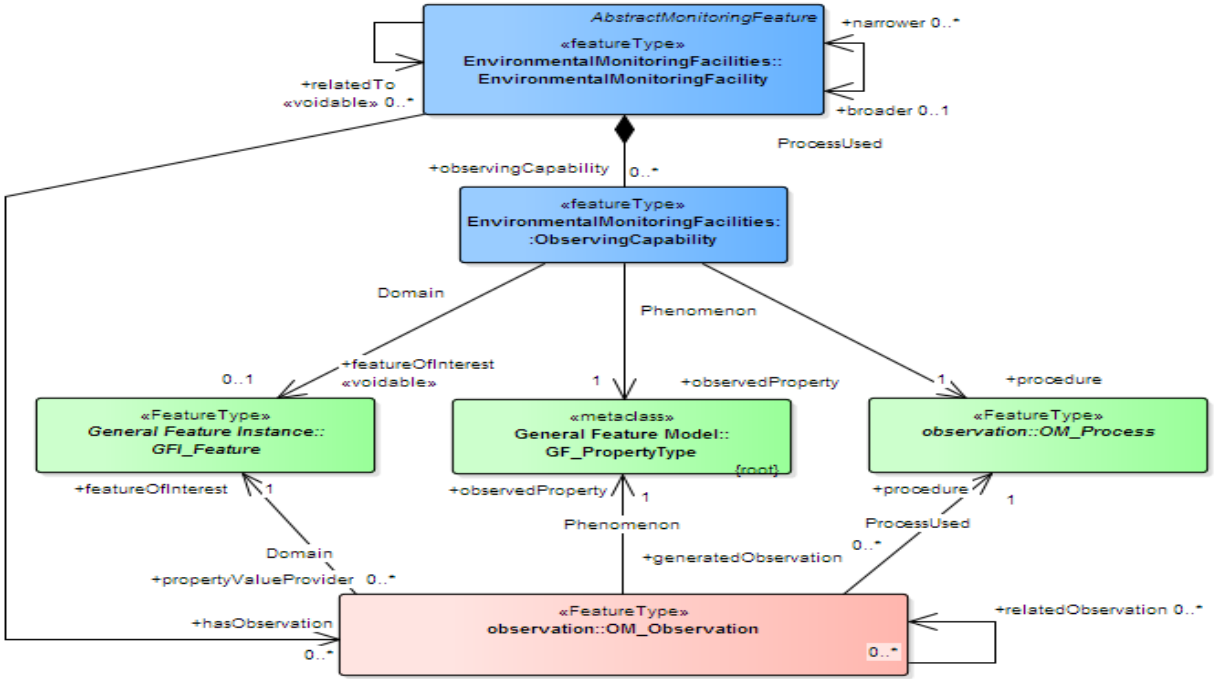


SensorThings API and INSPIRE

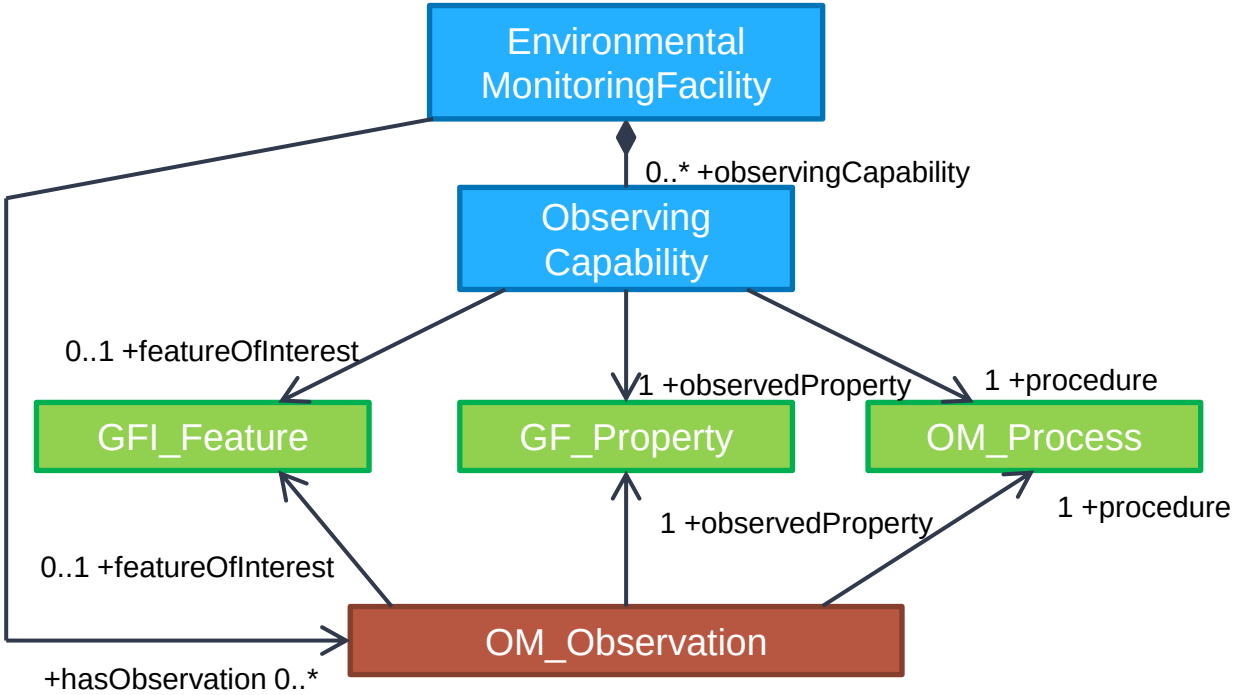


sensorup

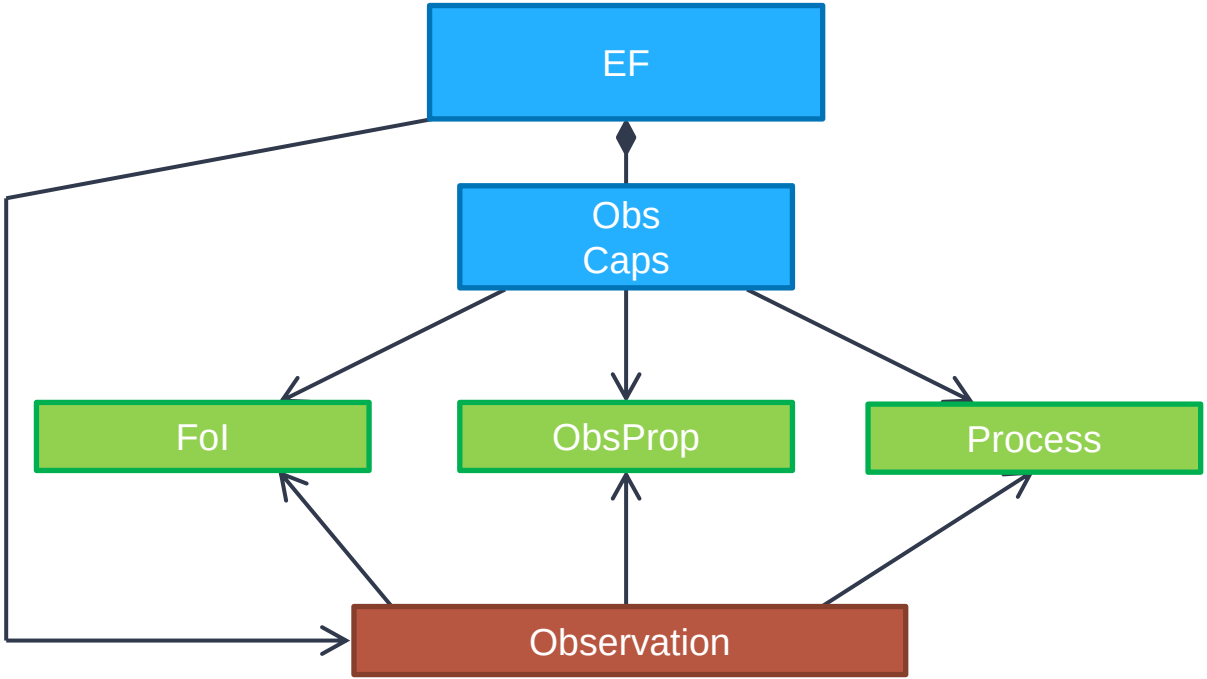
INSPIRE EF


AbstractMonitoringFeature
«featureType»
EnvironmentalMonitoringFacilities::
EnvironmentalMonitoringFacility
+narrower 0..*
+broader 0..1
+relatedTo
«voidable» 0..*
ProcessUsed
+observingCapability
0..*
«featureType»
EnvironmentalMonitoringFacilities:
:ObservingCapability
Domain
Phenomenon
+featureOfInterest
«voidable»
0..1
1
+observedProperty
1
+procedure
«FeatureType»
General Feature Instance::
GFI_Feature
«metaclass»
General Feature Model::
GF_PropertyType
{root}
«FeatureType»
observation::OM_Process
+featureOfInterest
1
+observedProperty
1
+procedure
1
Phenomenon
ProcessUsed
Domain
+propertyValueProvider
0..*
0..*
+generatedObservation
+hasObservation
0..*
«FeatureType»
observation::OM_Observation
0..*
+relatedObservation 0..*

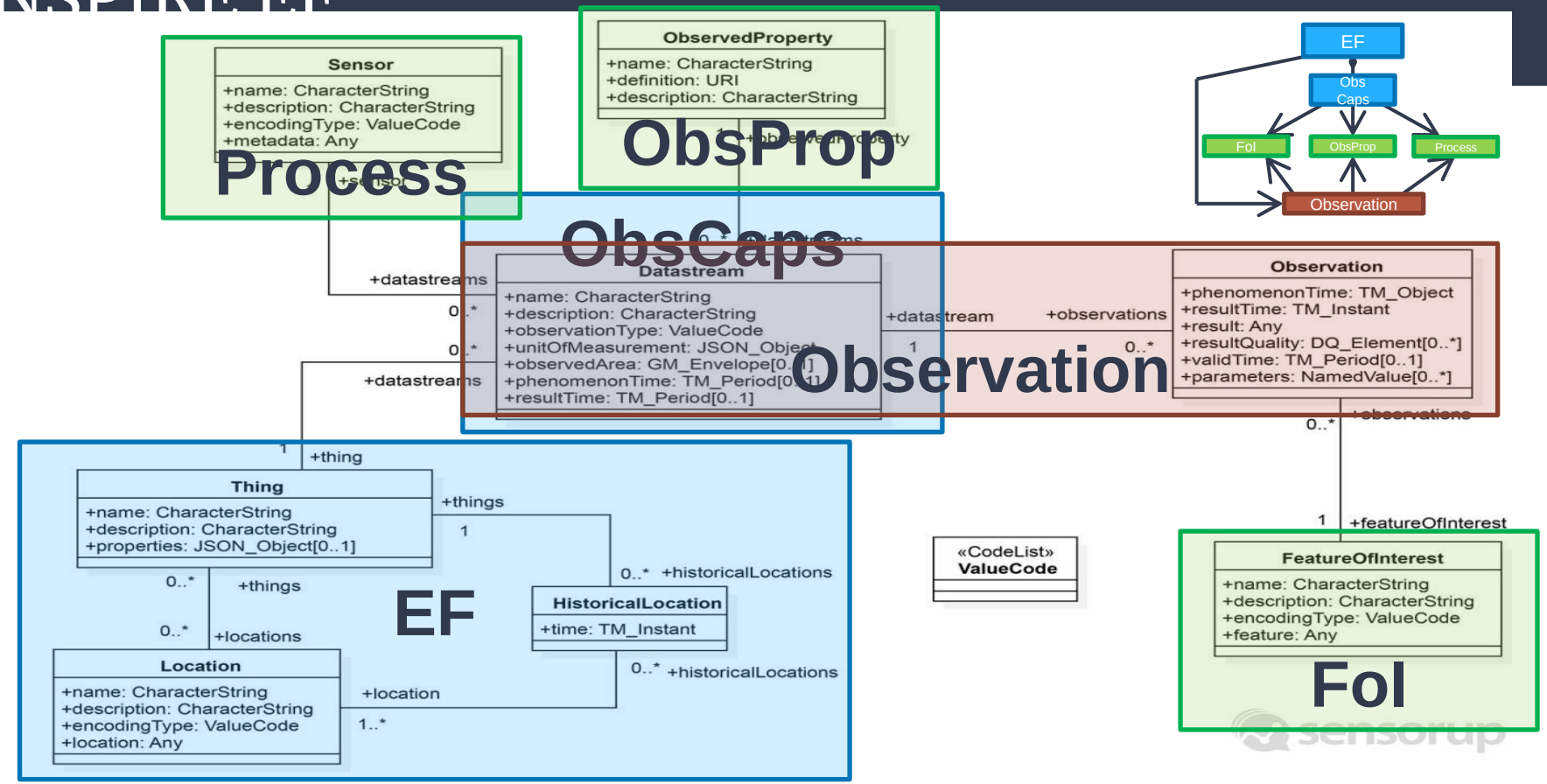
INSPIRE EF


Environmental MonitoringFacility
0..* +observingCapability
Observing Capability
0..1 +featureOfInterest
1 +observedProperty
1 +procedure
GFI_Feature
GF_Property
OM_Process
1 +observedProperty
1 +procedure
0..1 +featureOfInterest
OM_Observation
+hasObservation 0..*

INSPIRE EF



INSPIRE EF


Sensor
+name: CharacterString
+description: CharacterString
+encodingType: ValueCode
+metadata: Any
Process
ObservedProperty
+name: CharacterString
+definition: URI
+description: CharacterString
ObsProp
ObsCaps
Datastream
+name: CharacterString
+description: CharacterString
+observationType: ValueCode
+unitOfMeasurement: JSON_Object
+observedArea: GM_Envelope[0..1]
+phenomenonTime: TM_Period[0..1]
+resultTime: TM_Period[0..1]
+datastreams
+datastream
+observations
Observation
+phenomenonTime: TM_Object
+resultTime: TM_Instant
+result: Any
+resultQuality: DQ_Element[0..*]
+validTime: TM_Period[0..1]
+parameters: NamedValue[0..*]
Observation
EF
Obs Caps
FoI
ObsProp
Process
Observation
+thing
Thing
+name: CharacterString
+description: CharacterString
+properties: JSON_Object[0..1]
+things
+historicalLocations
HistoricalLocation
+time: TM_Instant
+locations
Location
+name: CharacterString
+description: CharacterString
+encodingType: ValueCode
+location: Any
+location
EF
«CodeList»
ValueCode
+featureOfInterest
FeatureOfInterest
+name: CharacterString
+description: CharacterString
+encodingType: ValueCode
+feature: Any
FoI

INSPIRE EF and STA

- Publication „Extending INSPIRE to the Internet of Things through SensorThings API“ in Geosciences provides full information on mapping between INSPIRE EF and SensorThings API
- www.mdpi.com/2076-3263/8/6/221
- Next step is setting up Good Practice Examples

Current SOS Specification Enhancements

- SOS Result Filtering functionality
- Not yet supported by the SOS 2.0 standard
- Relevant for two SOS operations
  - GetObservation
  - GetDataAvailability
- Approach was implemented by 52°North, supported and funded by BRGM
- Incorporated into the open source 52°North SOS server implementation
- In progress: Submission of OGC Discussion Paper/Best Practice

Expression	Applied to value type
Equals	numeric, textual
Greater Than	numeric
Greater Than or Equal To	numeric
Less Than	numeric
Less Than or Equal To	numeric
Between	numeric
Contains	textual

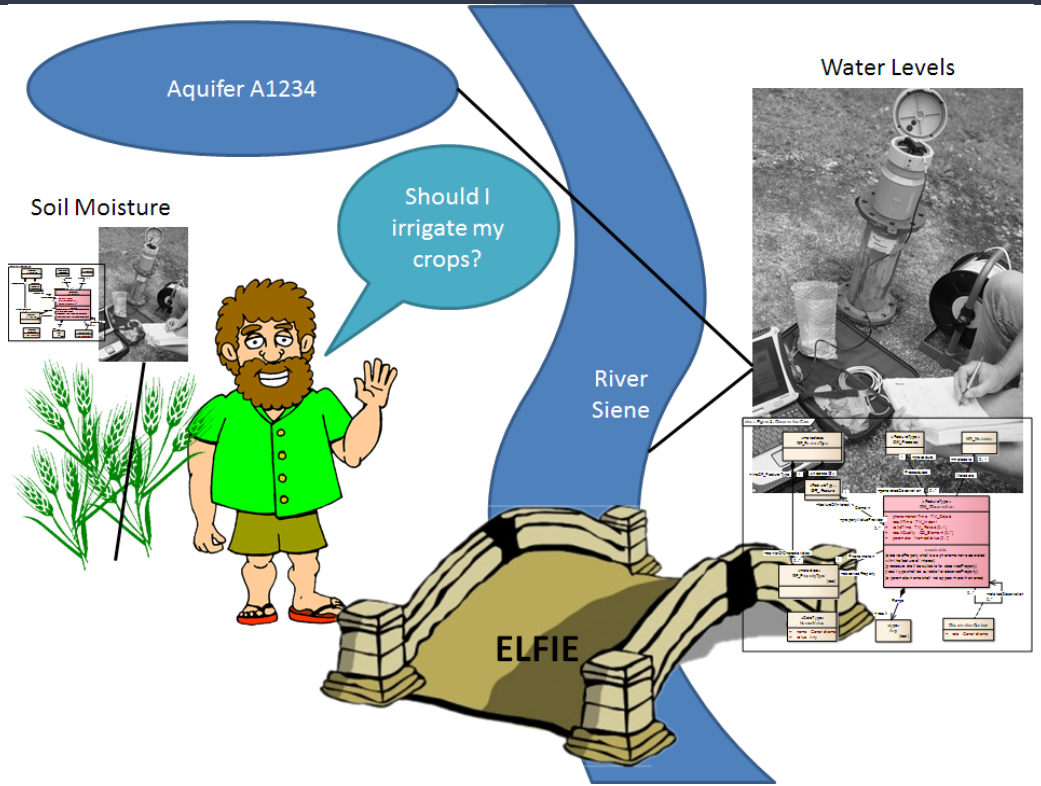
OGC ELFIE

- Data from sensors ubiquitous (IoT), defined as ‚spatial data' (see COM(2017) 9)
- Diverse thematic data models being developed, no common approach for interlinkage between observational data and domain features
- OGC Metadata (CSW, WFS & SOS Capabilities) not indexed by standard search engines reduces uptake
- Relationships to domain features (i.e. rivers, aquifers, or soils) relevant for data discovery and use (in addition to location)
- Linked Data requires standardisation of relationships for cross system sharing

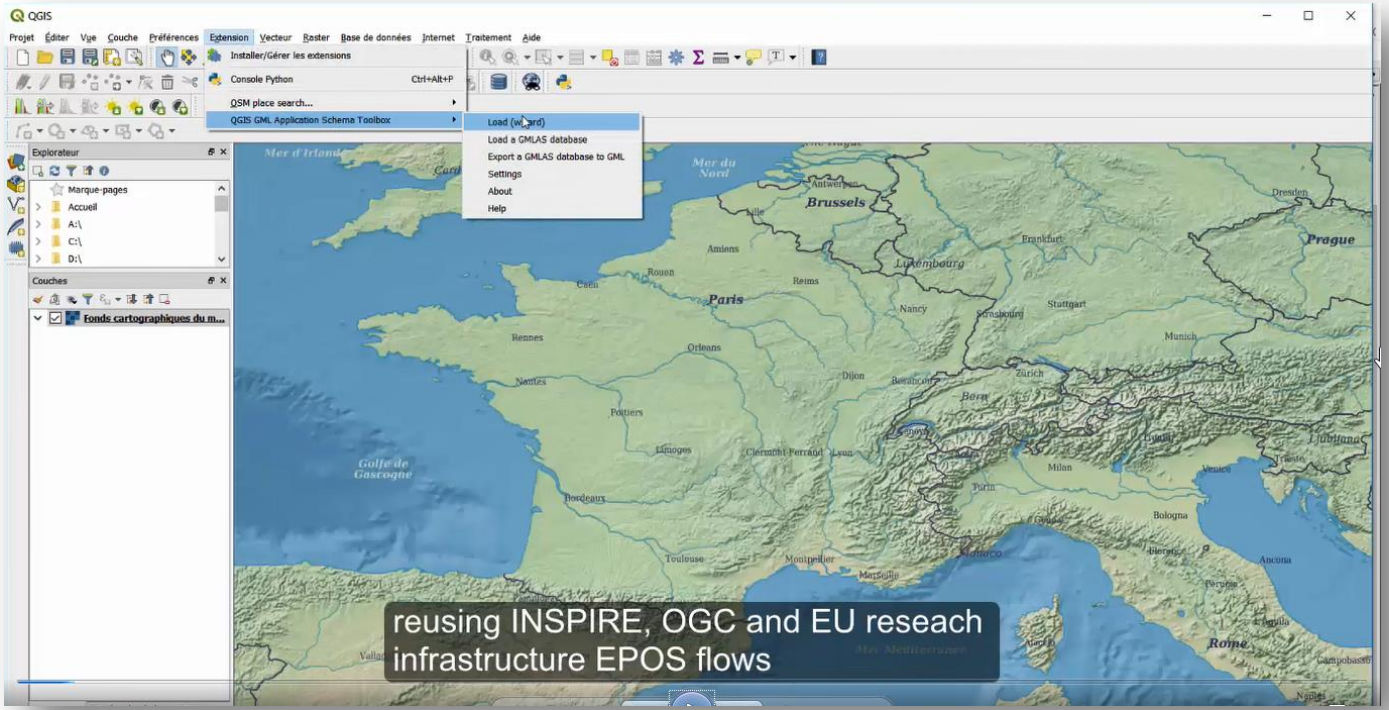
Objectives:

- Demonstrate integration of environmental observation data with domain features (ReSTful and Linked Data principles)
- Prepare OGC engineering report on interlinkages identified between observation data and domain features
- Provide draft linked data encodings to relevant standards working groups

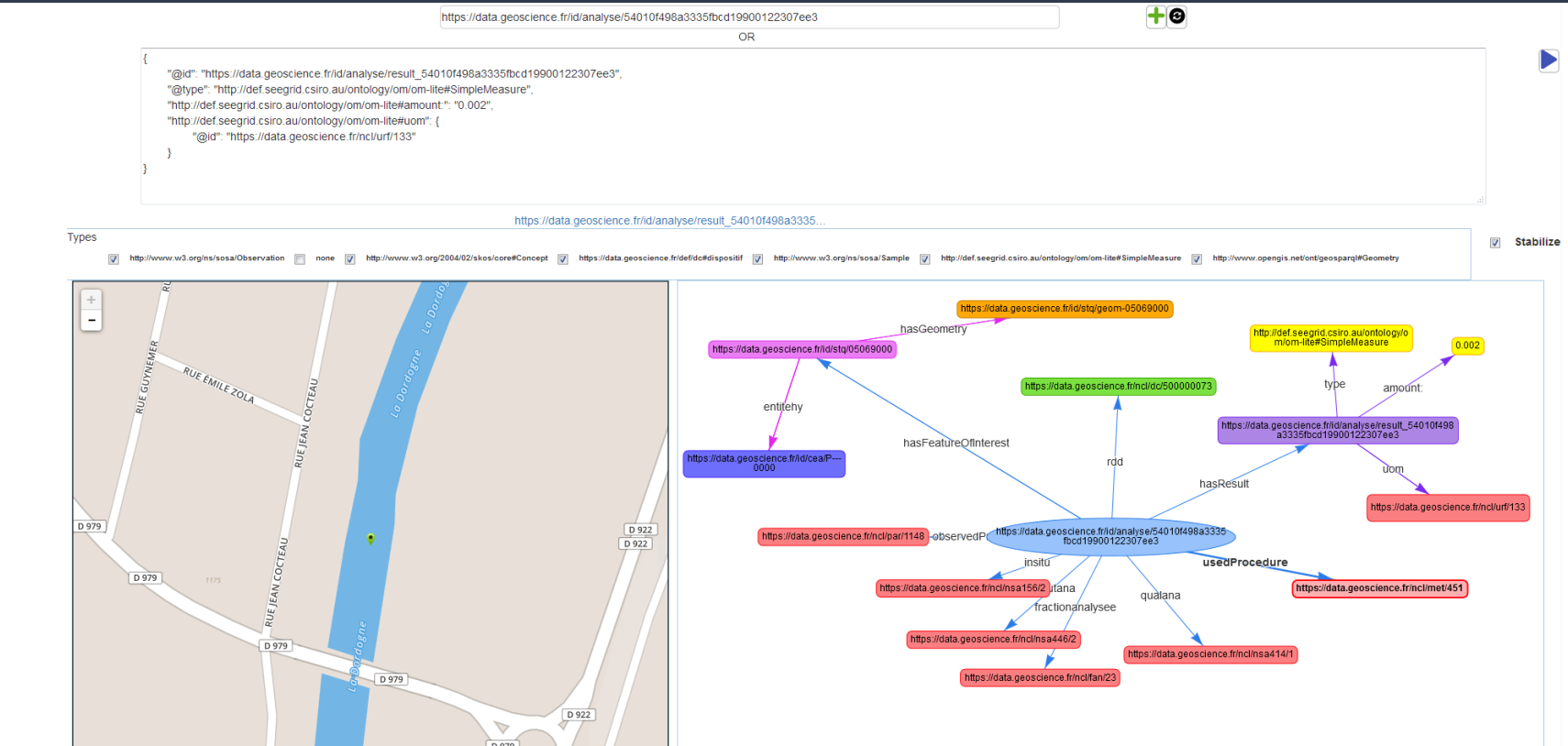
OGC ELFIE


Aquifer A1234
Water Levels
Soil Moisture
Should I irrigate my crops?
River Siene
ELFIE

OGC ELFIE


QGIS
Projet Éditer Vue Couche Préférences Extension Vecteur Raster Base de données Internet Traitement Aide
Installer/Gérer les extensions
Console Python Ctrl+Alt+P
QSM place search...
QGIS GML Application Schema Toolbox
Load (wizard)
Load a GMLAS database
Export a GMLAS database to GML
Settings
About
Help
Explorateur
Marque-pages
Accueil
A:\
C:\
D:\
Couches
Fonds cartographiques du m...
Brussels
Paris
Prague
Luxembourg
Frankfurt
Stuttgart
Munich
Zürich
Bern
Milan
Bologna
Ancona
Rome
Amiens
Rouen
Reims
Nancy
Rennes
Orleans
Nantes
Dijon
Poitiers
Limoges
Bordeaux
Toulouse
Montpellier
Marseille
Golfe de Gascogne
reusing INSPIRE, OGC and EU reseach
infrastructure EPOS flows

ELFIE – Bliv Viewer from BRGM allows exploration



Bliv Viewer and QGIS GML AS Plugin from BRGM allows exploration

Bliv video:
https://github.com/INSIDE-information-systems/EnvironmentalSemanticWeb/blob/master/demos/chemical_observation.mp4

QGIS video:
https://github.com/BRGM/gml_application_schema_toolbox/blob/master/presentations/2018_INSPIRE_conference/1.2.0_video_INSPIRE_PPI_conf_2018.mp4

O&M Simple Feature Encodings

- Need to share observation data with a variety of existing GIS client software (OpenLayers, GDAL & QGIS etc.)
- Complex feature type XML is not well supported in desktop GIS/Web GIS client applications: needs app. schema specific parsing code. OGC OMXML encoding (10-025r1) schema is complex (deeply structural) -> limited client software support.
- Simpler (~flat) GML/GeoJSON structures facilitate easier data provisioning and use:
  - Data ingestion & viewing easier for the client applications.
  - GIS data storage easier at the server side (one table row / observation event).
  - Data provisioning technically simpler for the WFS 2.0 and the upcoming WFS3 server applications.

Simple, Interoperable GML + GeoJSON

- Simplified encodings for the O&M 2.0 Observation model compliant with the GML Simple Features Profile 2.0 and the GeoJSON specification (IETF RFC 7946)
- Follows the O&M model structure and property naming as far as possible, some properties split to keep the encoding flat.
  - **GML encoding** uses SF-0 when possible, SF-1 when necessary (like repeated properties for timeseries).
  - **GeoJSON encoding** uses the standard “properties” object for the O&M properties, interoperable plain GeoJSON.
- A WFS3 server for serving these encodings is being implemented in a Vaisala/Finnish Meteorological Institute co-funded PoC project.
- More info at https://github.com/opengeospatial/omsf-profile

OMSF GeoJSON Example (MeasureObservation)


```
{
 "type": "Feature",
  "id": "f-1",
   "geometry": { "type": "Point", "coordinates": [ 24.96131, 60.20307 ] },
   "properties": {
      "observationType": "MeasureObservation",
      "phenomenonTime": "2017-08-17T12:00:00Z",
      "resultTime": "2017-08-17T12:01:25Z",
      "usedProcedureName": "Meteorological surface observations",
      "usedProcedureReference": "http://xml.fmi.fi/process/met-surface-observations",
      "observedPropertyName": "Air temperature",
      "observedPropertyReference": "http://vocab.nerc.ac.uk/collection/P07/current/CFSN0023/",
      "ultimateFeatureOfInterestName": "Helsinki Kumpula",
      "ultimateFeatureOfInterestReference": "http://sws.geonames.org/843429/about.rdf",
      "unitOfMeasureName": "Degree Celsius”,
      “unitOfMeasureReference”: “http://www.opengis.net/def/uom/UCUM/degC”
      "result": 12.5
  }
```

Part 2. O&M clinic

Issue / problem

- How to provide simple, O&M based data in a format that's directly usable by existing, generic GIS client applications / libraries?
- Can I use my existing server software / files-based access (WFS 2.0, Atom etc.)?
- Can I also use it with WFS3 in the future?

Cure: O&M Simple Feature encodings

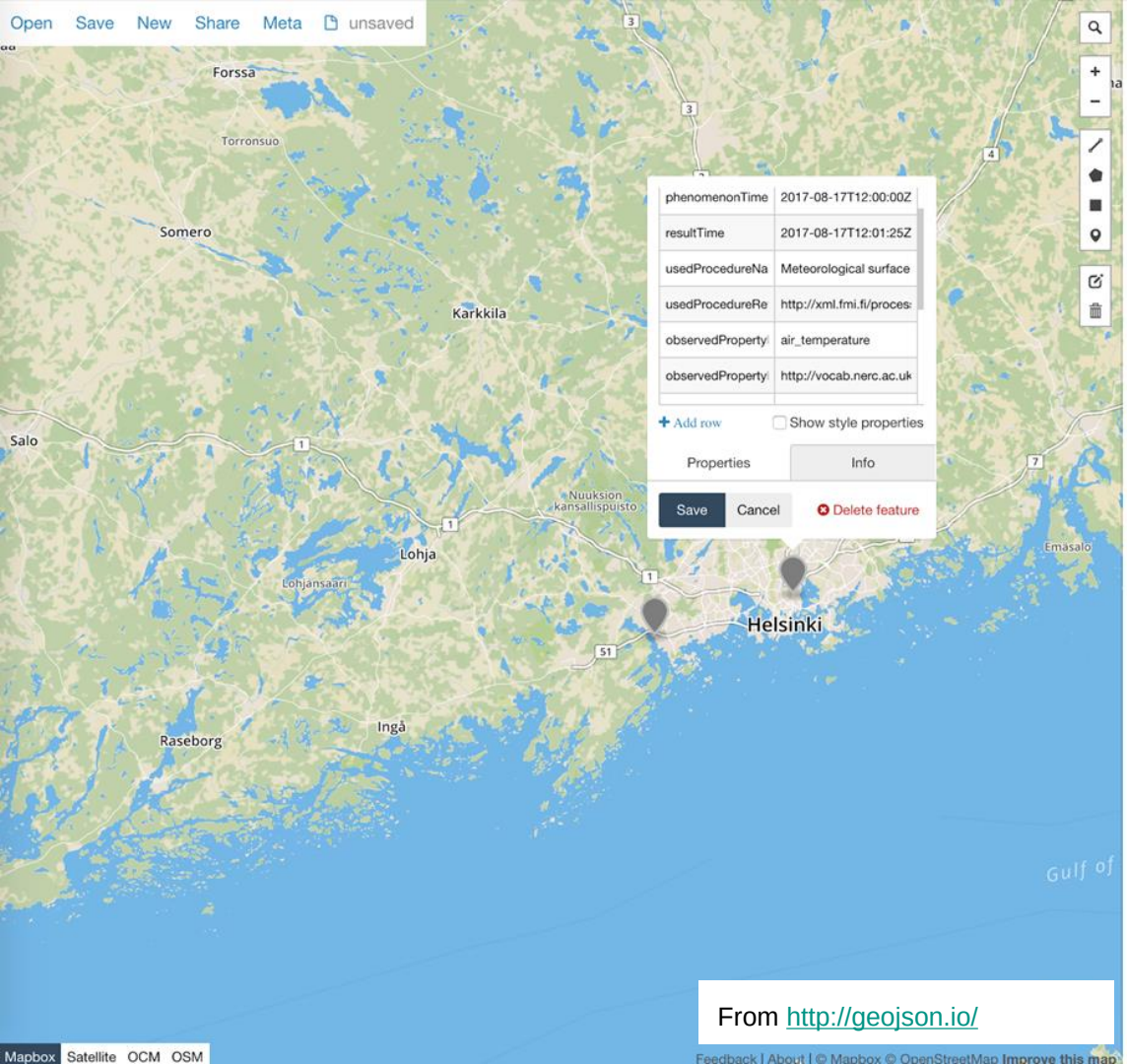
- Designed to be easily usable by application developers, flat data structure, simple property values.
- Strong standard basis in the ISO 19156 (Observations & Measurements).
- Independent of the APIs to be used for providing the data
  - The sama data encoding to be used for WFS 2.0, Atom etc.
- Two encoding options (~same properties):
  - GML Simple Features profile compatible encoding, and
  - GeoJSON Feature encoding
- Work-in-progress (Sept 2018):
  - INSPIRE 2017.2 Alternative encodings
  - OGC Sensor Web Enablement working groups
- More info at https://github.com/opengeospatial/omsf-profile

OMSF GeoJSON Example (MeasureObservation)

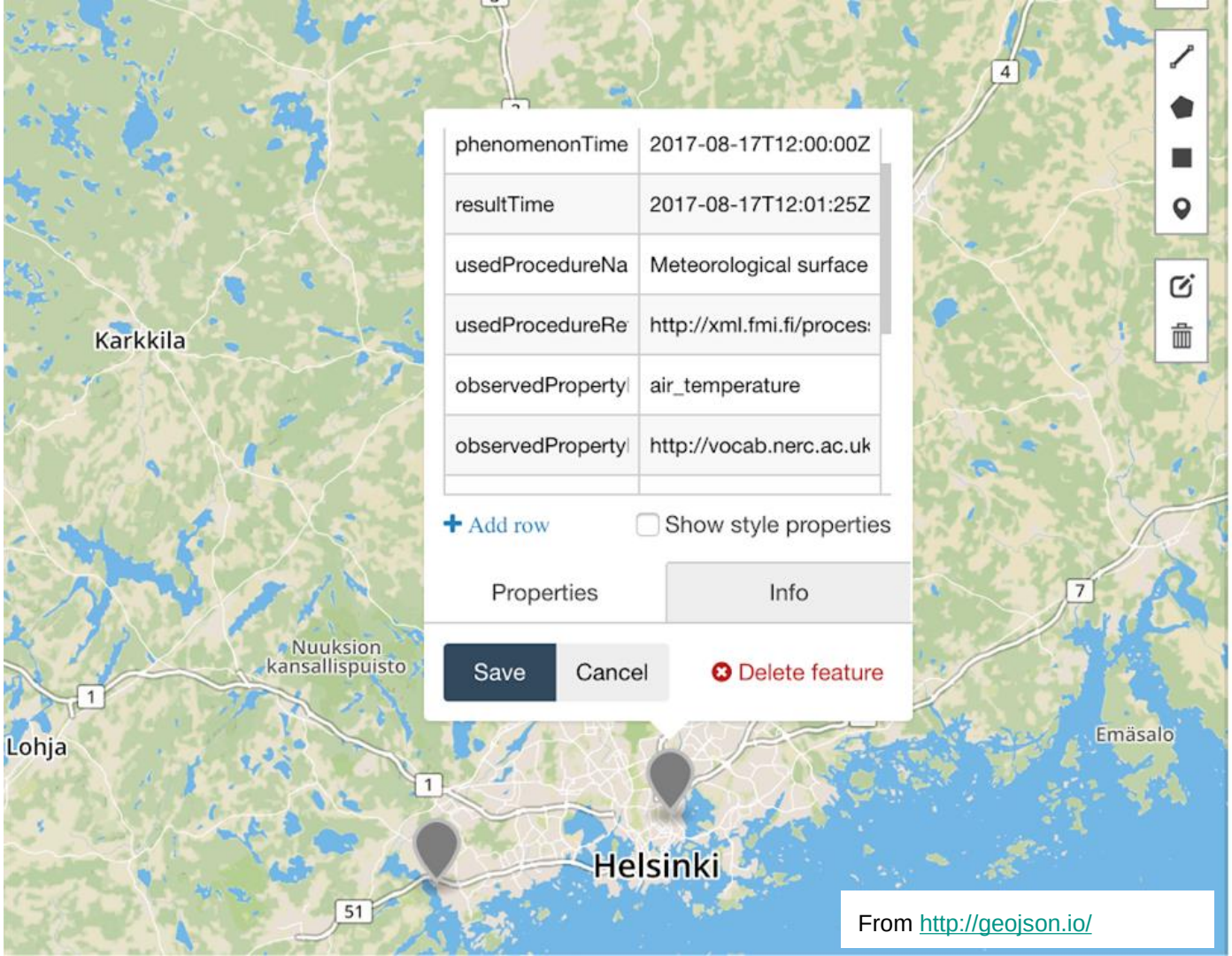

```
{
 "type": "Feature",
  "id": "f-1",
   "geometry": { "type": "Point", "coordinates": [ 24.96131, 60.20307 ] },
   "properties": {
      "observationType": "MeasureObservation",
      "phenomenonTime": "2017-08-17T12:00:00Z",
      "resultTime": "2017-08-17T12:01:25Z",
      "usedProcedureName": "Meteorological surface observations",
      "usedProcedureReference": "http://xml.fmi.fi/process/met-surface-observations",
      "observedPropertyName": "Air temperature",
      "observedPropertyReference": "http://vocab.nerc.ac.uk/collection/P07/current/CFSN0023/",
      "ultimateFeatureOfInterestName": "Helsinki Kumpula",
      "ultimateFeatureOfInterestReference": "http://sws.geonames.org/843429/about.rdf",
      "unitOfMeasureName": "Degree Celsius”,
      “unitOfMeasureReference”: “http://www.opengis.net/def/uom/UCUM/degC”
      "result": 12.5
```

OMSF GML Example (MeasureObservation)

```
<omsf:MeasureObservation gml:id="f-1">
  <omsf:phenomenonTime>2017-08-17T12:00:00Z</omsf:phenomenonTime>
  <omsf:resultTime>2017-08-17T12:01:25Z</omsf:resultTime>
  <omsf:usedProcedure
    xlink:href="http://xml.fmi.fi/process/met-surface-observations" xlink:title="Surface observations" />
  <omsf:observedProperty
    xlink:href="http://vocab.nerc.ac.uk/collection/P07/current/CFSN0023/" xlink:title="air_temperature" />
  <omsf:geometry>
    <gml:Point gml:id="p-1" srsName="http://www.opengis.net/def/crs/EPSG/0/4258" srsDimension="2">
                    <gml:pos>60.20307 24.96131</gml:pos>
    </gml:Point>
  </omsf:geometry>
  <omsf:ultimateFeatureOfInterestName>Helsinki Kumpula</omsf:ultimateFeatureOfInterestName>
  <omsf:ultimateFeatureOfInterestReference
          xlink:href="http://sws.geonames.org/843429/about.rdf"/>
  <omsf:result uom="Cel">12.5</omsf:result>
</omsf:MeasureObservation>
```



From http://geojson.io/




```
"features" : [
{
  "type" : "Fe
  "id": "f-1",
  "geometry":
      "type":
      "coordin
  },
  "properties"
      "observa
      "phenome
      "resultT
      "usedPro
      "usedPro
      "observe
      "observe
      "samplin
      "ultimat
      "ultimat
      "result"
      "unitOfM
      "unitOfM
  }
```

From http://geojson.io/

WFS3/OMSF Proof-of-concept Project

- Technology proof-of-concept project co-funded by Vaisala and the Finnish Meteorological Institute. Timeline: August 2018 - late spring 2019.
- Key goals:
  - Design, test, promote the use of O&M Simple Feature encodings (OMSF GeoJSON & GML) in providing environmental observation/forecast datasets to the public.
  - Design, implement and demonstrate a WFS3 PoC server for providing OMSF observation & forecast data (GeoJSON / GML).
- OGC & INSPIRE alignment & engagement very important to the success of the project.

VAISALA

FINNISH METEOROLOGICAL INSTITUTE

WFS3 PoC Design & Technologies

- Common core server delegating to runtime-integrated backend modules. Implementation is based on node.js (using Google's V8 high-performance JavaScript engine).
- TypeScript likely to be used for coding the core (automatically compiled into portable JavaScript).
- Core server code licenced under a permissible open source license.
- Docker images to be provided for simple deployment.
- The implementation started in the end of August 2018, the first public release expected in Dec 2018.
- Code, documentation and issues available at https://github.com/spatineo/sofp-core (may change yet).

FROST-server

- Complete SensorThings API implementation
- Open Source (https://github.com/FraunhoferIOSB/FROST-Server)
- Quick-Start:
  a. wget https://raw.githubusercontent.com/FraunhoferIOSB/FROST-Server/master/docker-compose.yaml
  b. docker-compose up

  Add demo entities:
  c. wget https://gist.githubusercontent.com/hylkevds/4ffba774fe0128305047b7bcbcd2672e/raw/demoEntities.json
  d. curl -X POST -H "Content-Type: application/json" -d @demoEntities.json http://localhost:8080/FROST-Server/v1.0/Things

  Explore:
  e. Open http://localhost:8080/FROST-Server/v1.0 in your browser

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

sandbox.datacove.eu:8080/rasdama | sandbox.datacove.eu:8080/rasdama | sandbox.datacove.eu:8080/rasdama | service.datacove.eu/ | service.datacove.eu:8080/SensorThi

service.datacove.eu:8080/SensorThingsServer-1.0_SK/v1.0 120% Suchen

JSON Rohdaten Kopfzeilen

Speichern Kopieren Alle einklappen Alle ausklappen JSON durchsuchen


```
value:
  0:
    name:   "Datastreams"
    url:    "http://service.datacove.eu:8080/SensorThingsServer-1.0_SK/v1.0/Datastreams"
  1:
    name:   "MultiDatastreams"
    url:    "http://service.datacove.eu:8080/SensorThingsServer-1.0_SK/v1.0/MultiDatastreams"
  2:
    name:   "FeaturesOfInterest"
    url:    "http://service.datacove.eu:8080/SensorThingsServer-1.0_SK/v1.0/FeaturesOfInterest"
  3:
    name:   "HistoricalLocations"
    url:    "http://service.datacove.eu:8080/SensorThingsServer-1.0_SK/v1.0/HistoricalLocations"
  4:
    name:   "Locations"
    url:    "http://service.datacove.eu:8080/SensorThingsServer-1.0_SK/v1.0/Locations"
  5:
    name:   "Observations"
    url:    "http://service.datacove.eu:8080/SensorThingsServer-1.0_SK/v1.0/Observations"
  6:
    name:   "ObservedProperties"
    url:    "http://service.datacove.eu:8080/SensorThingsServer-1.0_SK/v1.0/ObservedProperties"
  7:
    name:   "Sensors"
    url:    "http://service.datacove.eu:8080/SensorThingsServer-1.0_SK/v1.0/Sensors"
  8:
    name:   "Things"
    url:    "http://service.datacove.eu:8080/SensorThingsServer-1.0_SK/v1.0/Things"
```

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

sandbox.datacove.eu:8080/rasdama | sandbox.datacove.eu:8080/rasdama | sandbox.datacove.eu:8080/rasdama | service.datacove.eu/ | service.datacove.eu:8080/SensorThi | service.datacove.eu:8080/SensorThi

service.datacove.eu:8080/SensorThingsServer-1.0_SK/v1.0/Things 120% Suchen

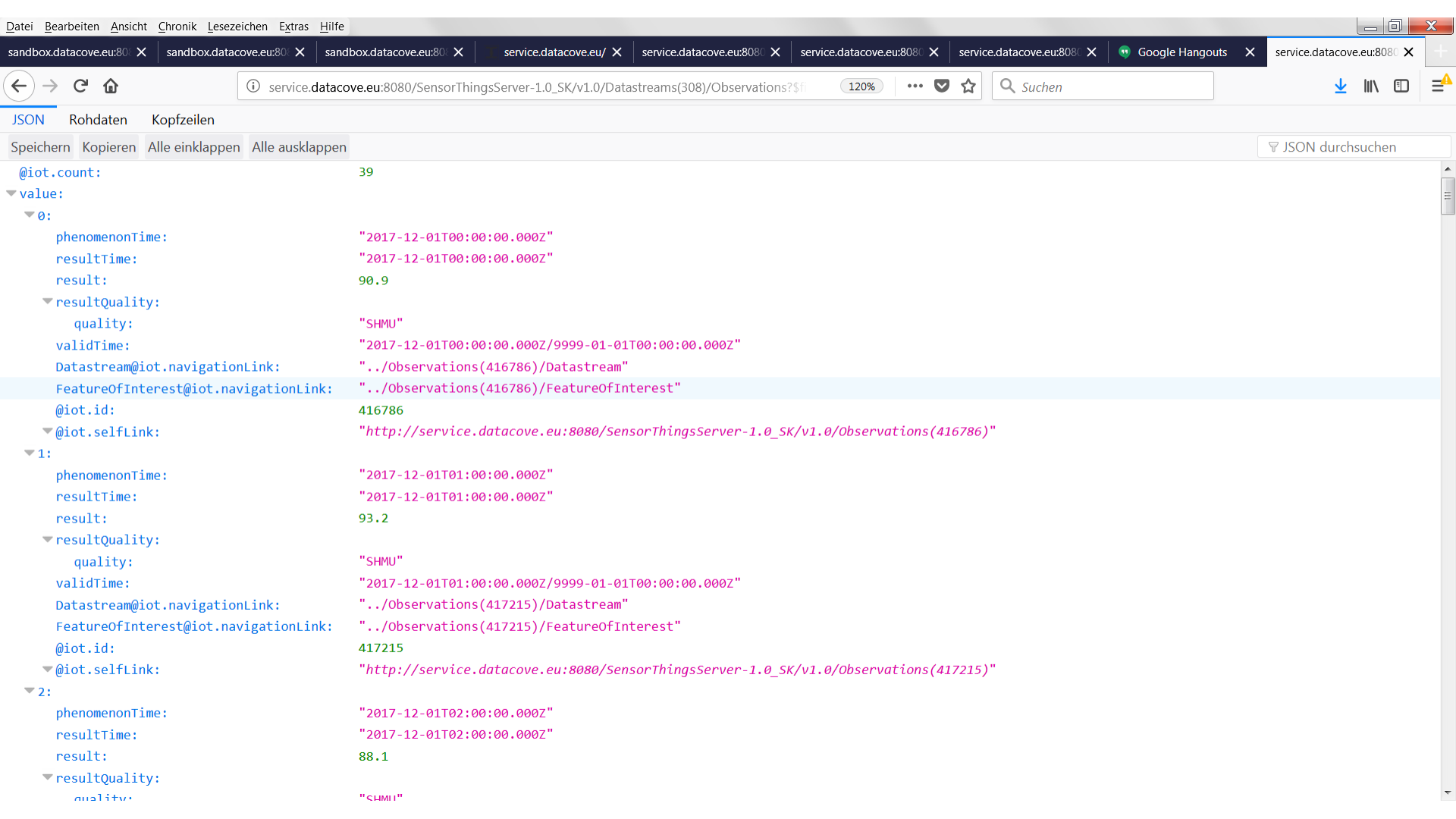
JSON Rohdaten Kopfzeilen

Speichern Kopieren Alle einklappen Alle ausklappen JSON durchsuchen


```
@iot.count:                          51
value:
  0:
    name:                            "BANSKA BYSTRICA "
    description:                     "BANSKA BYSTRICA "
    properties:
      code:                          "11:11898"
      icao:                          "ASBB"
    Locations@iot.navigationLink:    "Things(1)/Locations"
    HistoricalLocations@iot.navigationLink: "Things(1)/HistoricalLocations"
    Datastreams@iot.navigationLink:  "Things(1)/Datastreams"
    MultiDatastreams@iot.navigationLink: "Things(1)/MultiDatastreams"
    @iot.id:                         1
    @iot.selfLink:                   "http://service.datacove.eu:8080/SensorThingsServer-1.0_SK/v1.0/Things(1)"
  1:
    name:                            "PODOLINEC "
    description:                     "PODOLINEC "
    properties:
      code:                          "11:11950"
      icao:                          "ASPO"
    Locations@iot.navigationLink:    "Things(2)/Locations"
    HistoricalLocations@iot.navigationLink: "Things(2)/HistoricalLocations"
    Datastreams@iot.navigationLink:  "Things(2)/Datastreams"
    MultiDatastreams@iot.navigationLink: "Things(2)/MultiDatastreams"
    @iot.id:                         2
    @iot.selfLink:                   "http://service.datacove.eu:8080/SensorThingsServer-1.0_SK/v1.0/Things(2)"
  2:
    name:                            "MOLDAVA NAD BODVOU "
    description:                     "MOLDAVA NAD BODVOU "
    properties:
```

Filter

http://service.datacove.eu:8080/SensorThingsServer-1.0_SK/v1.0/Datastreams(308)/Observations?$filter=phenomenonTime%20lt%202017-12-02T14:37:01.000Z


Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
sandbox.datacove.eu:80
sandbox.datacove.eu:80
sandbox.datacove.eu:80
service.datacove.eu/
service.datacove.eu:808
service.datacove.eu:808
service.datacove.eu:808
Google Hangouts
service.datacove.eu:808
service.datacove.eu:8080/SensorThingsServer-1.0_SK/v1.0/Datastreams(308)/Observations?$f
120%
Suchen
JSON
Rohdaten
Kopfzeilen
Speichern
Kopieren
Alle einklappen
Alle ausklappen
JSON durchsuchen
@iot.count: 39
value:
0:
phenomenonTime: "2017-12-01T00:00:00.000Z"
resultTime: "2017-12-01T00:00:00.000Z"
result: 90.9
resultQuality:
quality: "SHMU"
validTime: "2017-12-01T00:00:00.000Z/9999-01-01T00:00:00.000Z"
Datastream@iot.navigationLink: "../Observations(416786)/Datastream"
FeatureOfInterest@iot.navigationLink: "../Observations(416786)/FeatureOfInterest"
@iot.id: 416786
@iot.selfLink: "http://service.datacove.eu:8080/SensorThingsServer-1.0_SK/v1.0/Observations(416786)"
1:
phenomenonTime: "2017-12-01T01:00:00.000Z"
resultTime: "2017-12-01T01:00:00.000Z"
result: 93.2
resultQuality:
quality: "SHMU"
validTime: "2017-12-01T01:00:00.000Z/9999-01-01T00:00:00.000Z"
Datastream@iot.navigationLink: "../Observations(417215)/Datastream"
FeatureOfInterest@iot.navigationLink: "../Observations(417215)/FeatureOfInterest"
@iot.id: 417215
@iot.selfLink: "http://service.datacove.eu:8080/SensorThingsServer-1.0_SK/v1.0/Observations(417215)"
2:
phenomenonTime: "2017-12-01T02:00:00.000Z"
resultTime: "2017-12-01T02:00:00.000Z"
result: 88.1
resultQuality:

52N SOS and SensorThings API

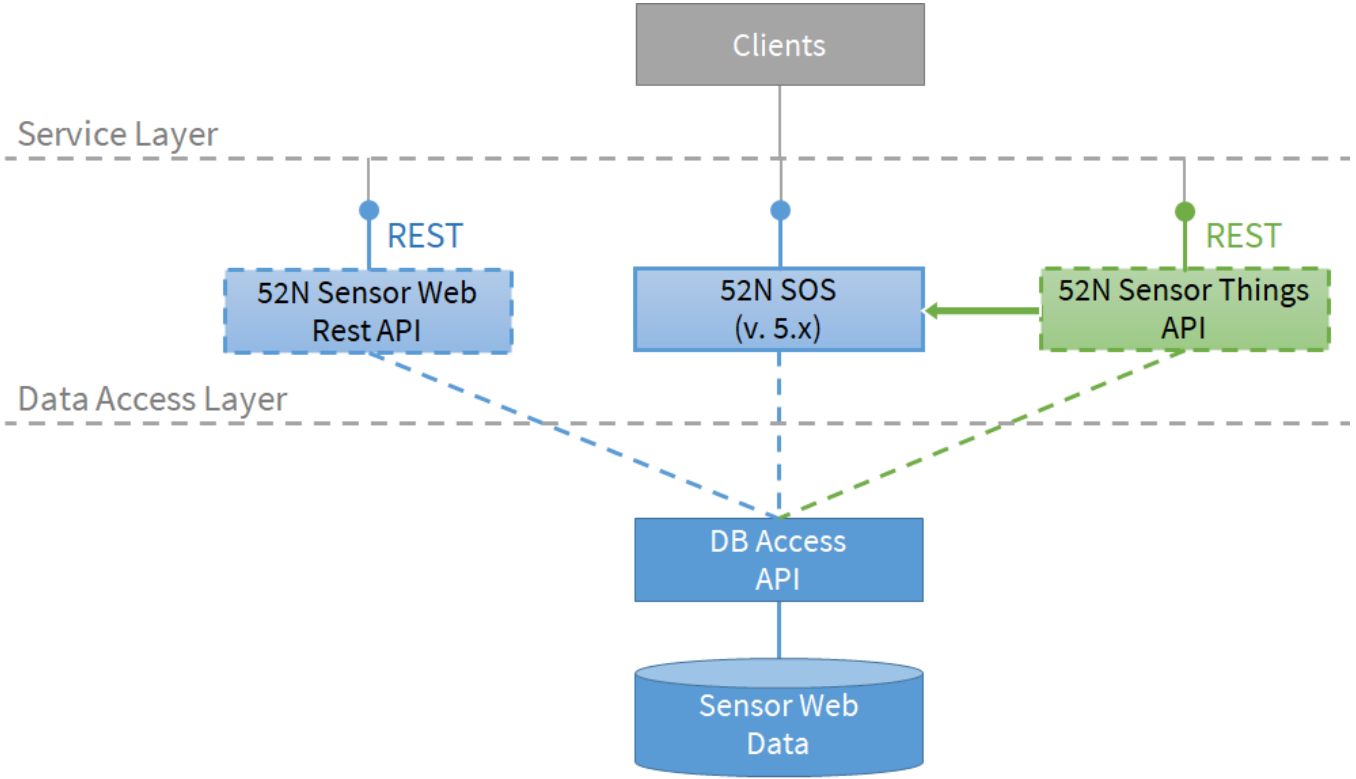
52N SOS and SensorThings API

- Full implementation of the SOS 2.0 Standard
- Extensions
  - INSPIRE-compliant Download Service
  - Domain-extensions for air quality, hydrology and marine observation data
  - Result filtering
- Flexible integration into existing IT infrastructures
  - Database abstraction
    - Object relational mappings
    - Support of different database management systems (e.g. PostgreSQL, Oracle, MySQL, Microsoft SQL server)
  - Different deployment strategies available
  - Tutorial: https://52north.github.io/sensor-web-tutorial/06_sos-setup.html
- Included in the OSGeo Live DVD

52N SOS and SensorThings API

- Implementation of the SensorThings API standard in progress
- First beta version expected for late Autumn 2018
- Current status
  - Integration of SensorThings data model into existing 52°North Sensor Web data model
  - Implementation of SensorThings entity requesting
    - Read-only access to the data storage by Spring Data repositories
    - Resource path parsing for GET-requests and providing entity data

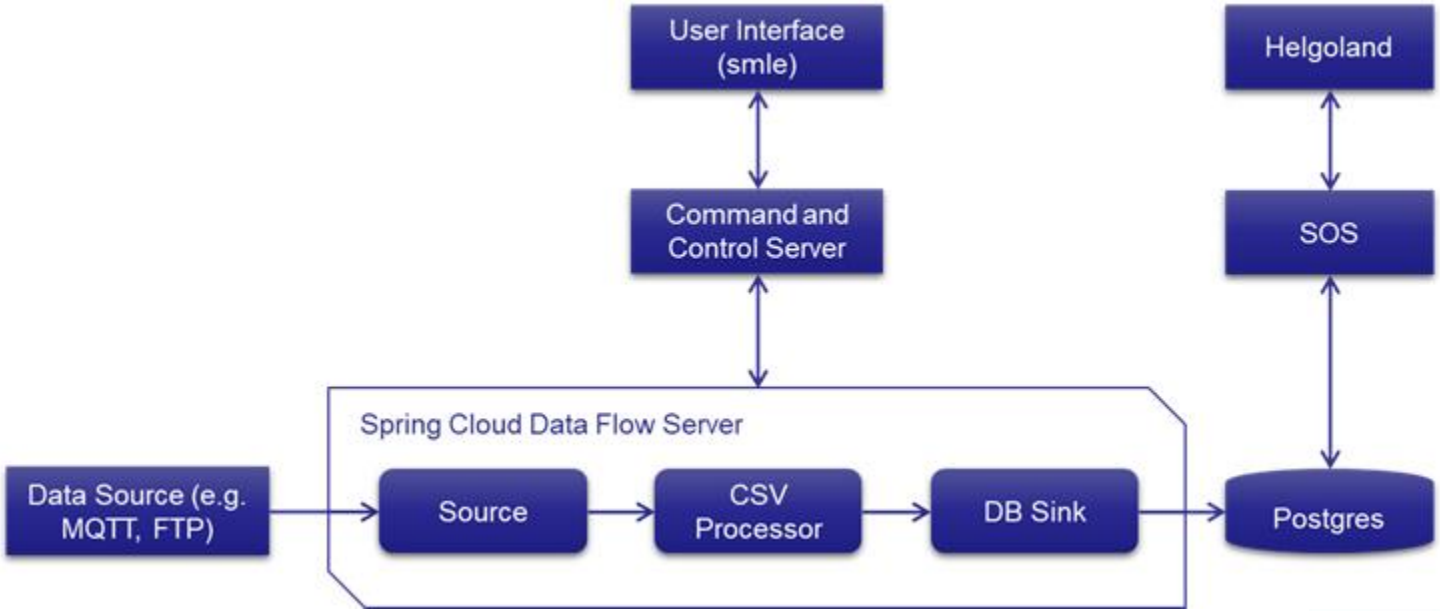
52N SOS and SensorThings API



Application Example: Marine Observations

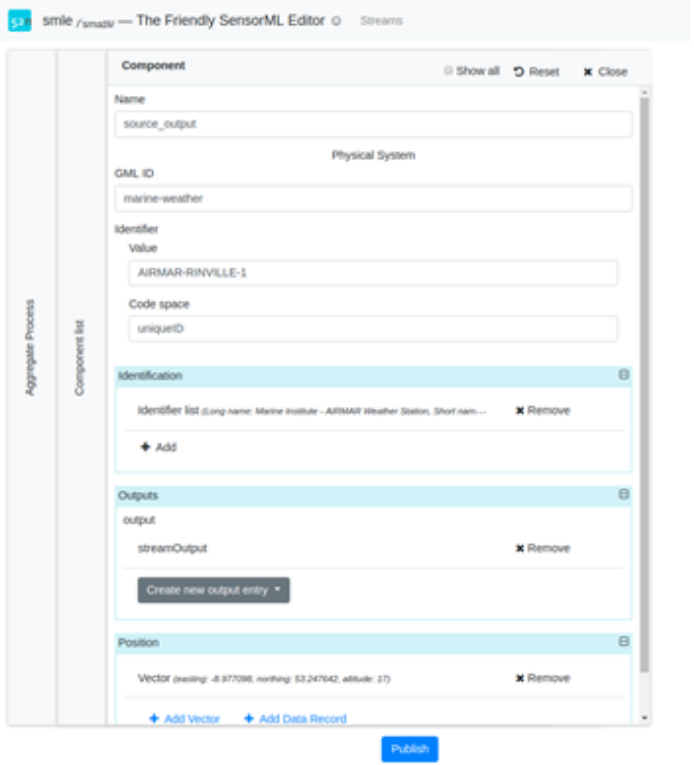
- Ocean observation data is collected by a broad range of institutes and organisations
- Science needs efficient discovery and access to collected observation data sets
- SeaDataCloud addresses this challenge
  - H2020 project
  - Started in November 2016
  - Follow-up project of SeaDataNet
  - Pan-European infrastructure, developed by NODCs and major research institutes from 34 countries
  - Infrastructure driving several portals of the European Marine Observation and Data network (EMODnet)
- Different challenges
  - Data discovery and metadata
  - Data access
  - Data publication
- In the following: SWE Ingestion Service for facilitating data publication

Application Example: Marine Observations



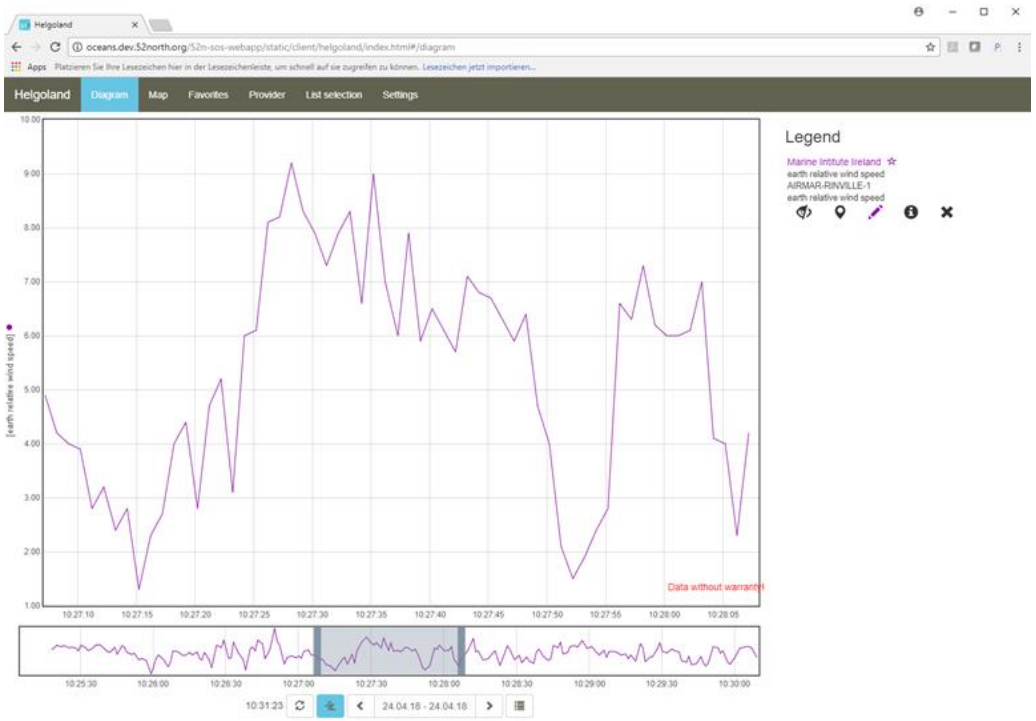
SeaDataCloud

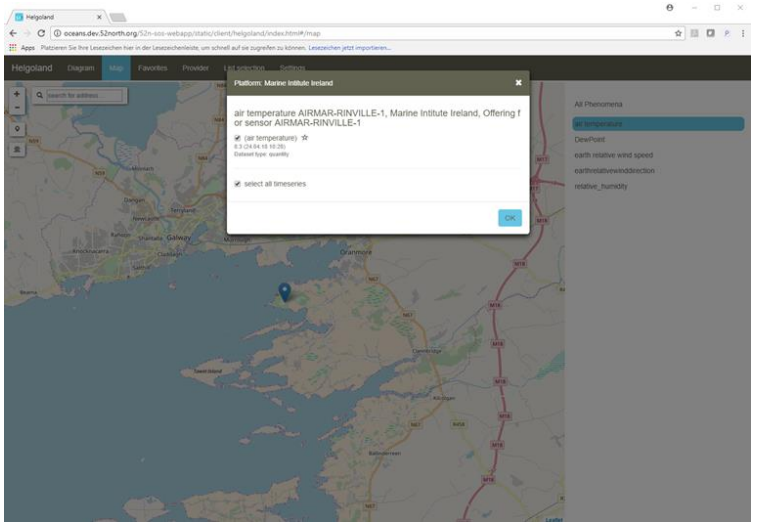
Application Example: Marine Observations



SeaDataCloud

Application Example: Marine Observations



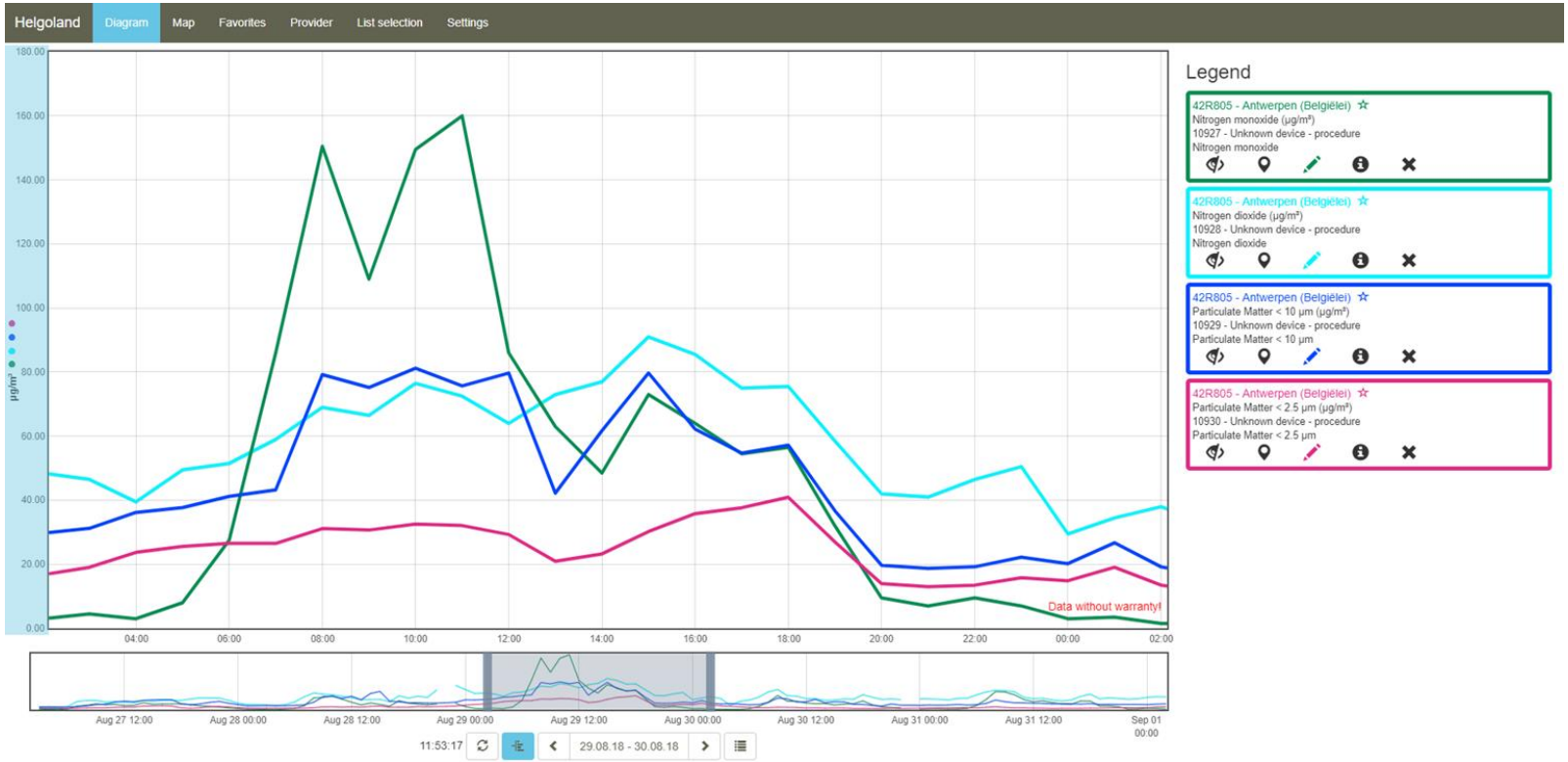


SeaDataCloud

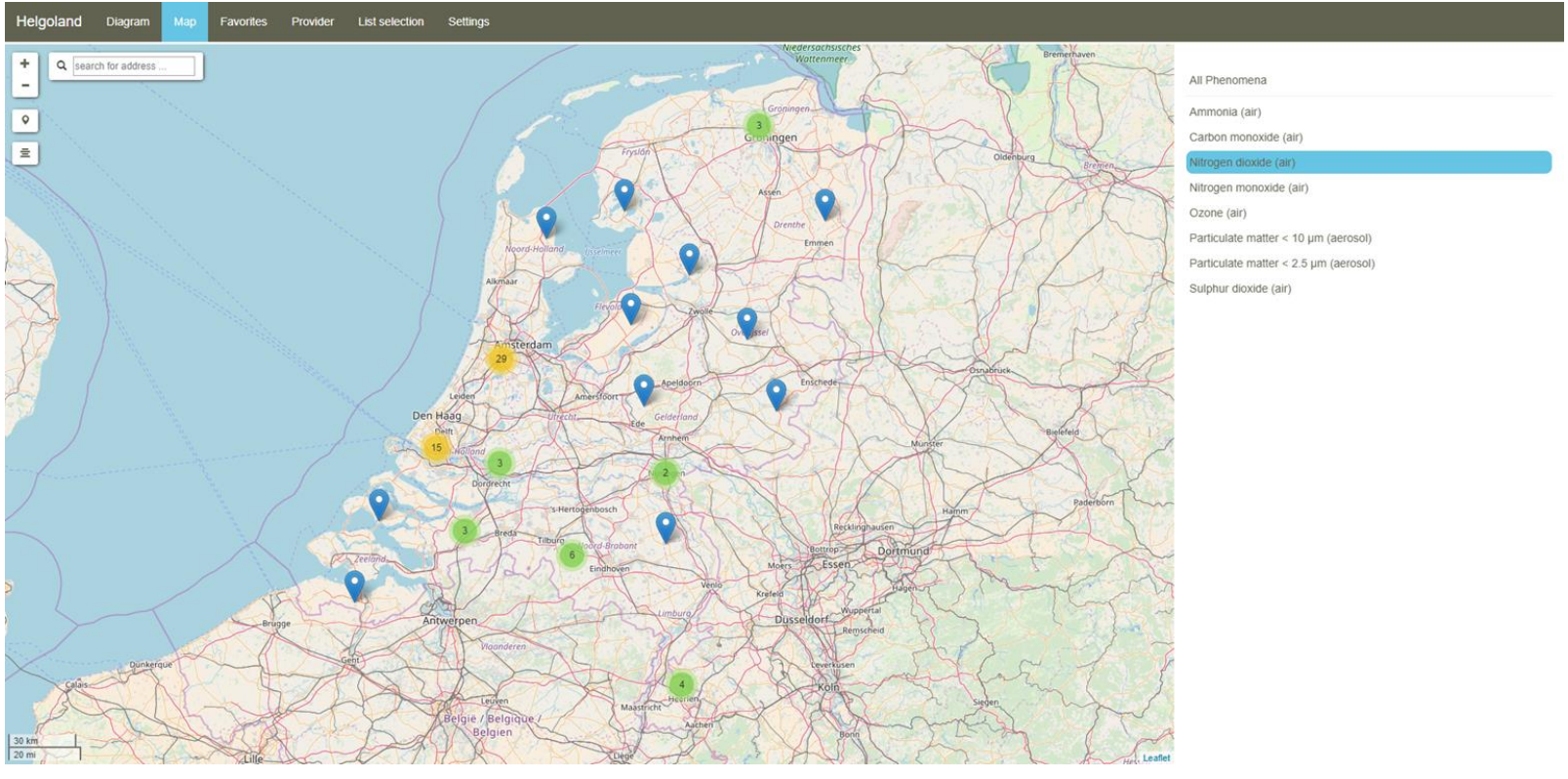
Application Example: Air Quality

- Several European countries are providing air quality data via SOS servers
- Different use cases
  - Service-based e-Reporting of air quality data to the European Environment Agency via the SOS interface
  - Development of additional applications based on the SOS-interface/additional REST-interface offered by SOS implementation
- Countries operation SOS servers include
  - The Netherlands
  - Belgium
  - United Kingdom
  - Sweden
  - Lithuania

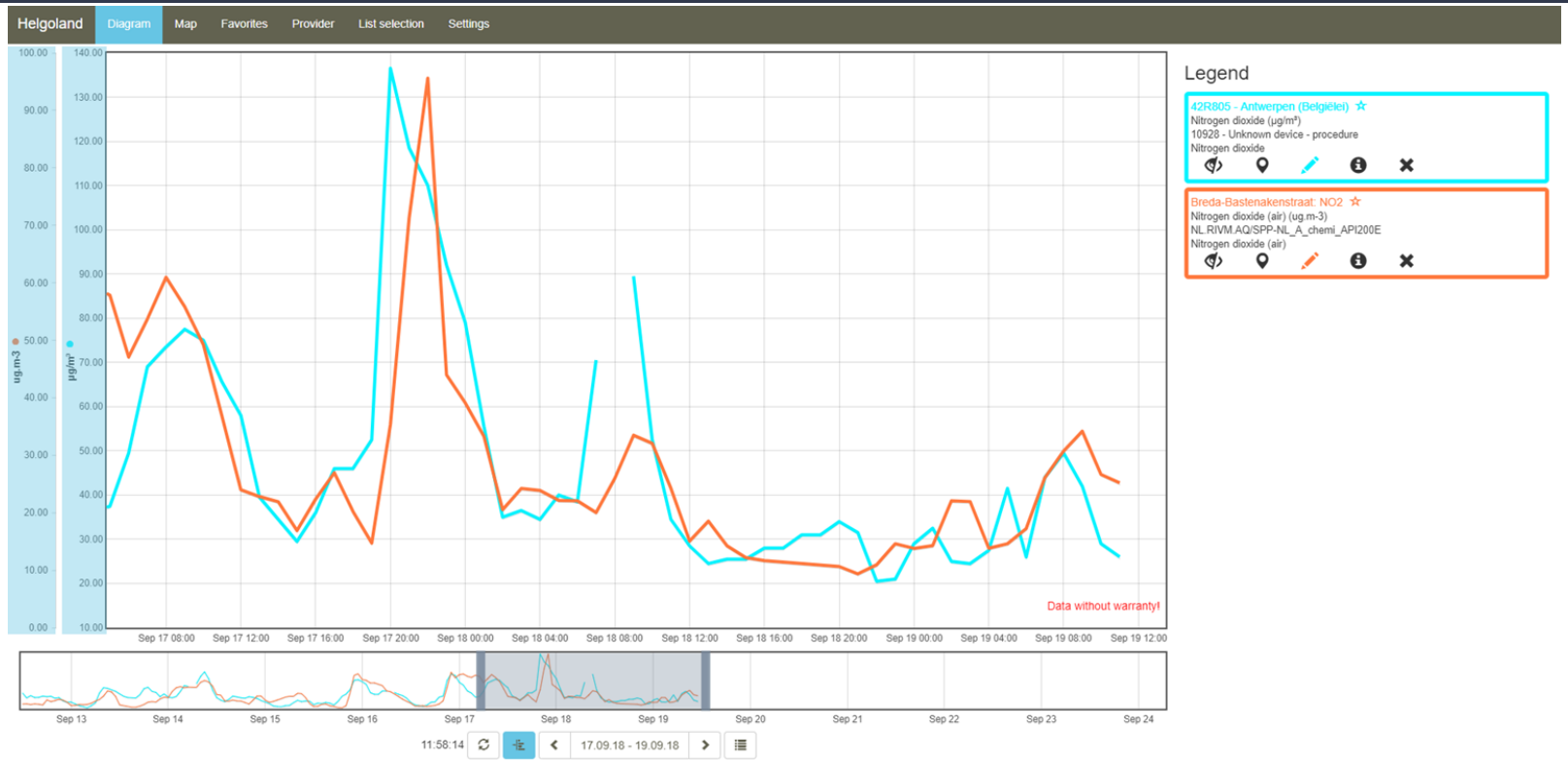
Application Example: Air Quality



Application Example: Air Quality



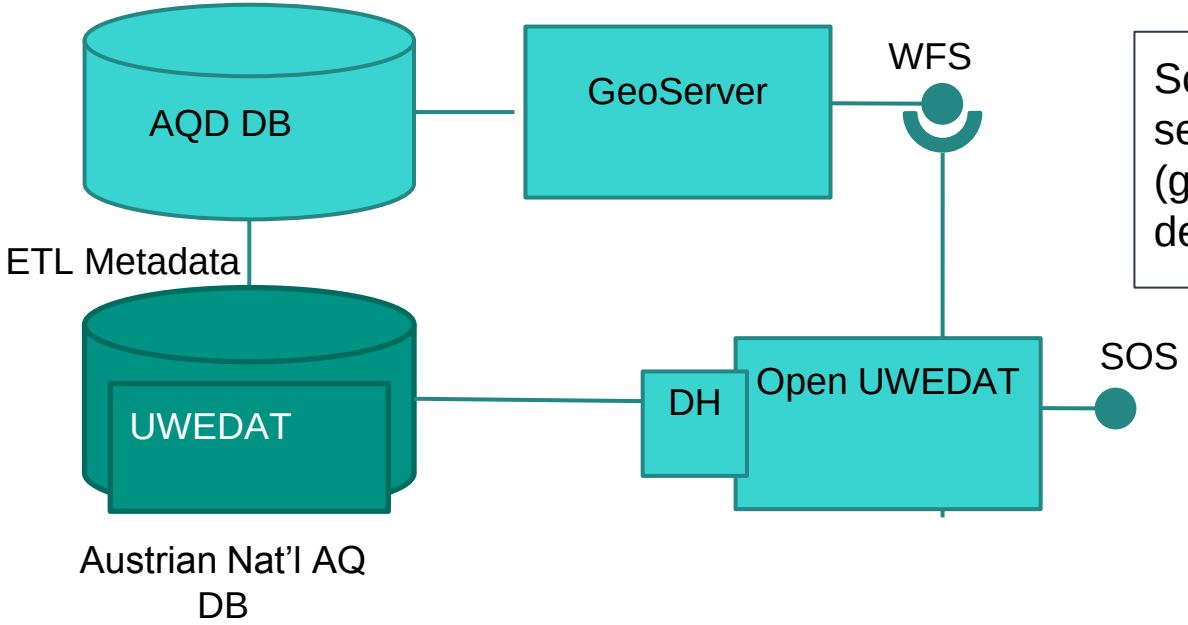
Application Example: Air Quality



AIT SOS

AIT/UBA AT - air quality

http://luft.umweltbundesamt.at/inspire/sos?service=SOS&version=2.0.0&request=getObservation&offering=urn:STA/SPO.06.036.64292.7.1&eventTime=2012-10-11T00:00/2012-10-12T00:00



Solution: Redirect for overlapping service requests (getFeatureOfInterest & describeSensor)

AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY

References

- Sensor Web Tutorial: https://52north.github.io/sensor-web-tutorial/
- Kotsev, A., Schleidt, K., Liang, S., van der Schaaf, H., Khalafbeigi, T., Grellet, S., ... & Beaufils, M. (2018). Extending INSPIRE to the Internet of Things through SensorThings API. Geosciences, 8(6), 221.
- OGC OM- SF: https://github.com/opengeospatial/omsf-profile/tree/master/omsf-gml

How everything fits together

- **Main objective - make observation data available in a SDI!**
- Multiple possible ways for doing that
- ETL is needed

Many commonalities

- All examples are based on the same abstract model (O&M)
- JSON is increasingly the preferred data encoding
- RESTful approach (incl. groupings of observations - Offering, Datatream, data-specific api endpoint)

Need help with your observation data? Ask us ...

THANK YOU!

Geoscience for a sustainable Earth
brgm

52north
exploring horizons

Fraunhofer
IOSB

VAISALA

FINNISH METEOROLOGICAL INSTITUTE

DATA COVE

European Commission