SOSPilot Documentation

Release 1.0.1

Thijs Brentjens, Just van den Broecke, Michel Grothe

16 November 2017 at 12:14:18

Contents

Contents:

CHAPTER 1

Intro

This is the main (technical) documentation for the SOS Pilot Project by Geonovum and RIVM. It can always be found at sospilot.readthedocs.org.

The project ran in the period 2014-2015 and is now (2017) archived. Website and documentation are kept for reference. See http://data.smartemission.nl for the follow-up project Smart Emission.

The home page for the SOSPilot project is http://sensors.geonovum.nl

The project GitHub repository is at https://github.com/Geonovum/sospilot.

Background reading, see this article: *"Building bridges: experiences and lessons learned from the implementation of INSPIRE and e-reporting of air quality data in Europe"* [PDF].

This is document version 1.0.1 generated on 16 November 2017 at 12:14:17.

1.1 About this Document

This document is written in Restructured Text (rst) generated by Sphinx and hosted by ReadTheDocs.org (RTD).

The sources of this document are (.rst) text files maintained in the Project's GitHub: https://github.com/Geonovum/sospilot/docs

You can also download a PDF version of this document and even an Ebook version.

This document is automatically generated whenever a commit is performed on the above GitHub repository (via a "Post-Commit-Hook")

Using Sphinx with RTD one effectively has a living document like a Wiki but with the structure and versioning characteristics of a real document or book.

Basically we let "The Cloud" (GitHub and RTD) work for us!

CHAPTER 2

Data Management

This chapter describes all technical aspects related to data within the SOSPilot project. The "data" as mentioned here applied initially to LML RIVM data, i.e. Dutch Air Quality Data. In 2015 this setup was also used to handle air quality data from the Smart Emission (Nijmegen) project. This is described in a separate chapter: *Smart Emission Project*.

- obtaining raw source data: observations&measurements (O&M), metadata (stations etc).
- data transformation (ETL) to target various (OWS) services, INSPIRE, E-reporting and custom services
- tooling (databases and ETL-tools) for the above

Source code for data management and ETL can be found in GitHub: https://github.com/Geonovum/sospilot/tree/master/src

2.1 Architecture

Figure 1 sketches the overall SOSPilot architecture with emphasis on the flow of data (arrows). Circles depict harvesting/ETL processes. Server-instances are in rectangles. Datastores are shown with "DB"-icons.

The main challenge/flow of data is from *Raw Source APIs* such as raw XML files with Air Quality data provided by the RIVM LML server. In a later stage of the project this architecture was reused for other raw O&M sources such as the CityGIS Sensor REST API in the Smart Emission project (see *Smart Emission Project*).

The RIVM LML file server provides a directory of XML files with hourly measurements of the past month. The big circle "ETL" embeds the data transformation processes and storage that is required to deliver OGC services for WMS, WFS (via *GeoServer*) and SOS (via the *52 North SOS Server*) and other *Custom Services* at a later stage (dashed rectangle) such as EU IPR-compliant E-reporting.

2.2 ETL Design

In this section the ETL as used for RIVM LML data is elaborated in more detail as depicted in the figure below.

The ETL design comprises three main processing steps and three datastores. The three ETL Steps are:

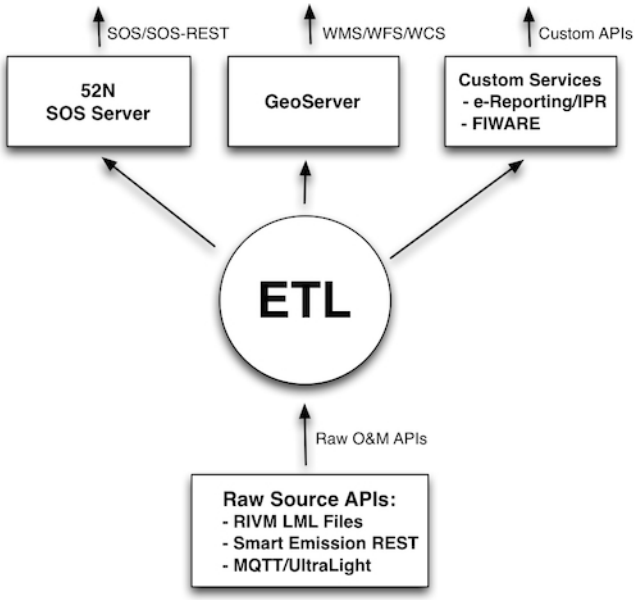


Fig. 2.1: *Figure 1 - Overall Architecture*

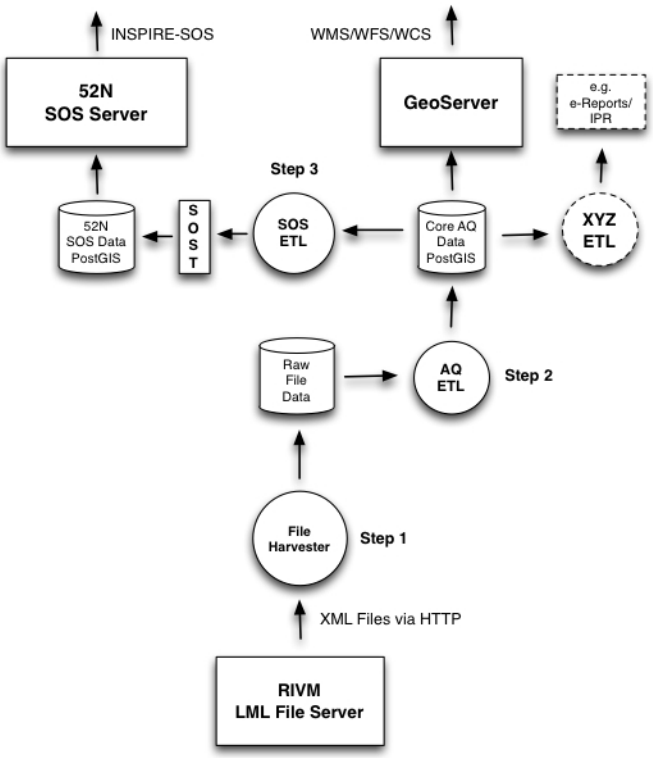


Fig. 2.2: *Figure 2 - Overall Architecture with ETL Steps*

1. File Harvester: fetch source data from RIVM e.g. from http://lml.rivm.nl/xml and store locally
2. AQ ETL: transform this local source data to intermediate "Core AQ Data" in PostGIS
3. SOS ETL: transform and publish "Core AQ Data" to the 52N SOS DB via SOS-Transactions (SOS-T)

The detailed dataflow from source to destination is as follows:

1. The File Harvester fetches XML files with AQ/LML measurements from the RIVM server
2. The File Harvester puts these files as XML blobs 1-1 in a Postgres/PostGIS database
3. The AQ ETL process reads these file blobs and transforms these to the Core AQ DB (Raw Measurements)
4. The Core AQ DB contains measurements + stations in regular tables 1-1 with original data, including a Time column
5. The Core AQ DB can be used for OWS (WMS/WFS) services via GeoServer (possibly using VIEW by Measurements/Stations JOIN)
6. The SOS ETL process transforms core AQ data to SOS Observations and publishes Observations using SOS-T InsertObservation
7. These three processes run continuously (via cron)
8. Each process always knows its progress and where it needs to resume, even after it has been stopped (by storing a progress/checkpoint info)

These last two ETL processes manage their `last sync-time` using a separate `progress table` within the database. The first (Harvester) only needs to check if a particular XML file (as they have a unique file name) has already been stored.

Advantages of this approach:

- backups of source data possible
- incrementally build up of history past the last month
- in case of (design) errors we can always reset the 'progress timestamp(s)' and restart anew
- simpler ETL scripts than "all-in-one", e.g. from "Core AQ DB" to "52N SOS DB" may even be in plain SQL
- migration with changed in 52N SOS DB schema simpler
- prepared for op IPR/INSPIRE ETL (source is Core OM DB)
- OWS server (WMS/WFS evt WCS) can directly use op Core OM DB (possibly via Measurements/Stations JOIN VIEW evt, see below)

The Open Source ETL tool Stetl, Streaming ETL , is used for most of the transformation steps. Stetl provides standard modules for building an ETL Chain via a configuration file. This ETL Chain is a linkage of Input, Filter and Output modules. Each module is a Python class derived from Stetl base classes. In addition a developer may add custom modules where standard Stetl modules are not available or to specialize processing aspects.

Stetl has been used sucessfully to publish BAG (Dutch Addresses and Buildings) to INSPIRE Addresses via XSLT and WFS-T (to the `deegree WFS server`) but also for transformation of Dutch topography (Top10NL and BGT) to PostGIS. As Stetl is written in Python it is well-integrated with standard ETL and Geo-tools like GDAl/OGR, XSLT and PostGIS.

At runtime Stetl (via the `stetl` command) basically reads the config file, creates all modules and links their inputs and outputs. This also makes for an easy programming model as one only needs to concentrate on a single ETL step.

2.2.1 ETL Step 1. - Harvester

The RIVM data server provides measurements of the past month in a collection of XML files served by an Apache HTTP server. See figure below.

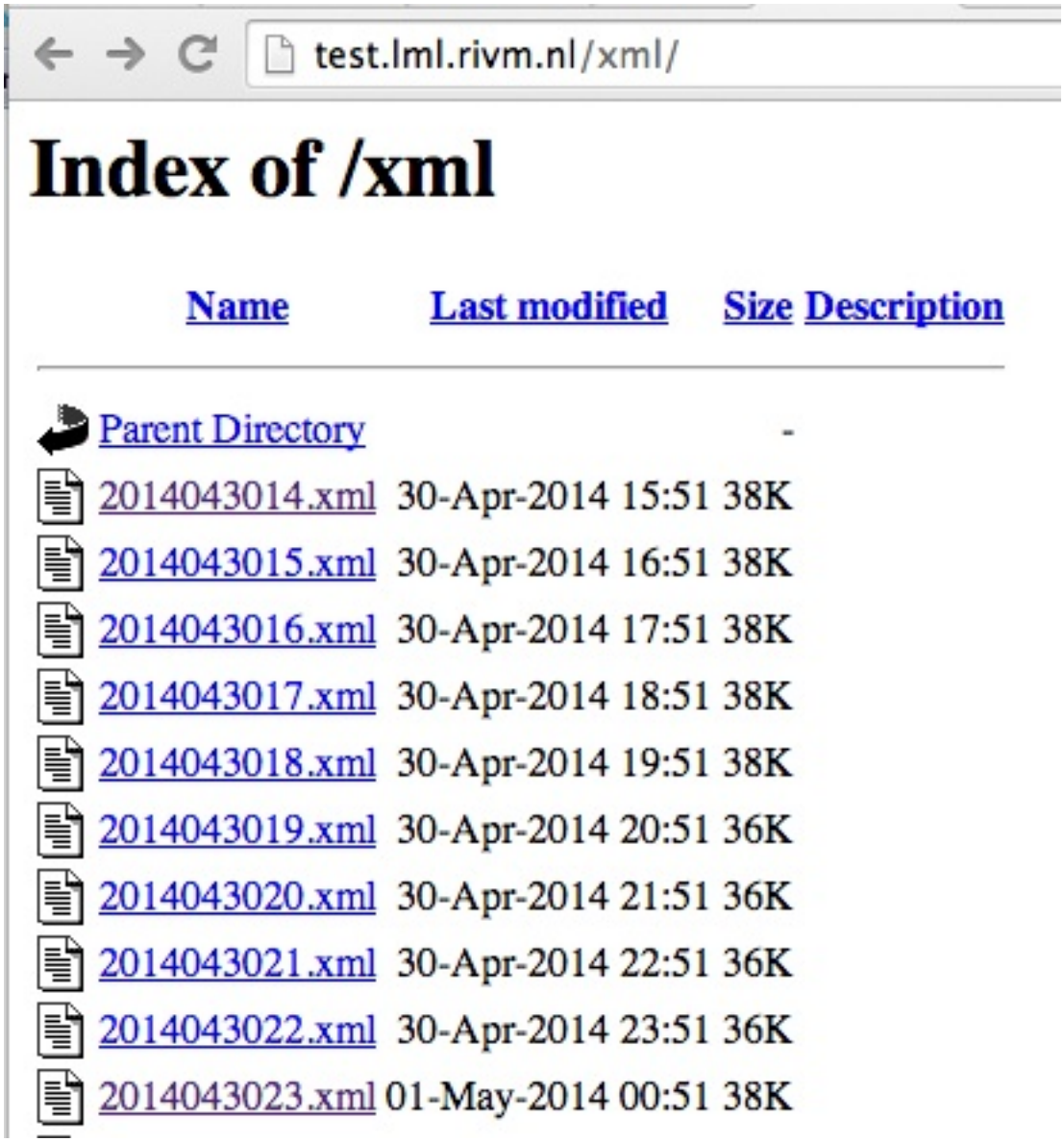


Fig. 2.3: *Figure - Apache Server Raw File Listing*

The LML Harvester will continuously read these XML files and store these in the DB as XML Blobs with their filename in the Postgres table `lml_files`.

This can be effected by a simple Stetl process activated every 30 mins via the linux `cron` service. Stetl has a built-in module for Apache dir listing reading. Only a derived version needed to be developed in order to track which files have been read already. This is implemented in the file https://github.com/Geonovum/sospilot/blob/master/src/rivm-lml/apachedirinput.py.

Note: there are two data streams with AQ Data from RIVM: "XML" oriented and "SOS" oriented. We will use the "XML" oriented as the file format is simpler to process and less redundant with station info. The URL is http:

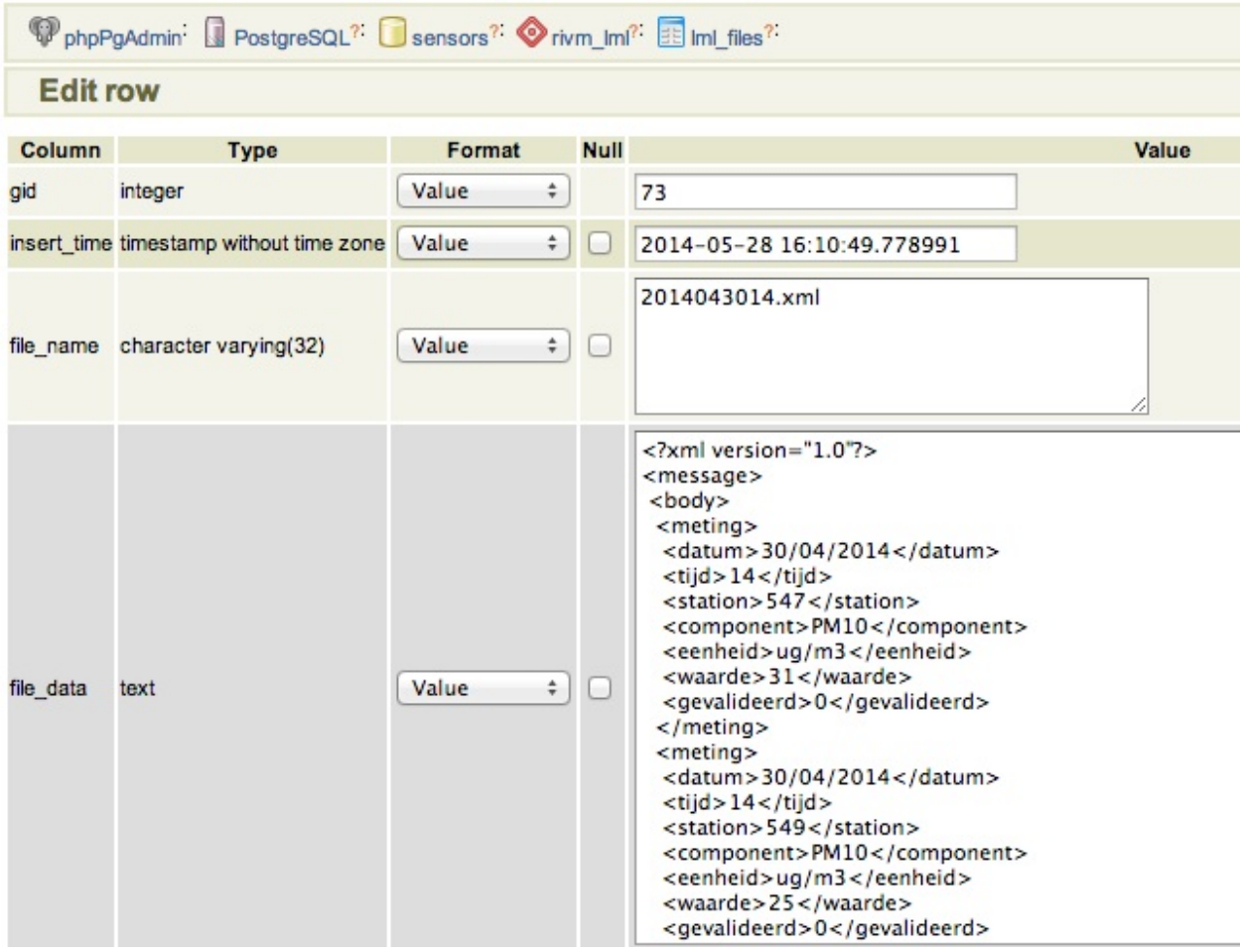


Fig. 2.4: *Figure - Raw File Record Harvested into DB*

//www.lml.rivm.nl/xml.

For completeness, the "SOS" oriented are identical in measurements, though not rounded, but that should be within error range.

There also seem to be differences, for example "SOS":

```
<ROW>
    <OPST_OPDR_ORGA_CODE>RIVM</OPST_OPDR_ORGA_CODE>
    <STAT_NUMMER>633</STAT_NUMMER>
    <STAT_NAAM>Zegveld-Oude Meije</STAT_NAAM>
    <MCLA_CODE>regio achtergr</MCLA_CODE>
    <MWAA_WAARDE>-999</MWAA_WAARDE>
    <MWAA_BEGINDATUMTIJD>20140527120000</MWAA_BEGINDATUMTIJD>
    <MWAA_EINDDATUMTIJD>20140527130000</MWAA_EINDDATUMTIJD>
</ROW>
```

vs "XML":

```
<meting>
    <datum>27/05/2014</datum>
    <tijd>13</tijd>
    <station>633</station>
    <component>CO</component>
    <eenheid>ug/m3</eenheid>
    <waarde>223</waarde>
    <gevalideerd>0</gevalideerd>
</meting>
```

Gotcha: there is a file called `actueel.xml` in the XML stream. This file has to be skipped to avoid double records.

2.2.2 ETL Step 2 - Raw Measurements

This step produces raw AQ measurements, "AQ ETL" in Figure 2, from raw source (file) data harvested in the table `lml_files` (see Step 1).

Two tables: `stations` and `measurements`. This is a 1:1 transformation from the raw XML. The `measurements` refers to the `stations` by a FK `station_id`. The table `etl_progress` is used to track the last file processed from `lml_files`.

Stations

Station info is available from Eionet as a CSV file. Coordinates are in EPSG:4258 (also used in INSPIRE).

To create "clean" version of eionet RIVM stations understood by ogr2ogr to read into PostGIS:

- download CSV from http://cdr.eionet.europa.eu/Converters/run_conversion?file=nl/eu/aqd/d/envurreqq/REP_D-NL_RIVM_20131220_D-001.xml&conv=450&source=remote
- this file saves as `REP_D-NL_RIVM_20131220_D-001.csv`
- copy to stations.csv for cleaning
- stations.csv: remove excess quotes, e.g. ""
- stations.csv: replace in CSV header `Pos` with `Lat,Lon`
- stations.csv: replace space between coordinates with comma: e.g `,51.566389 4.932792,` becomes `,51.566389,4.932792,`

- fix DateTime formatting to comply with OGR: replace `T00:00:00` to ‘‘ 00:00:00‘‘ so `1976-04-02T00:00:00+01:00` becomes `1976-04-02 00:00:00+01:00`
- test result stations.csv by uploading in e.g. Geoviewer: http://kadviewer.kademo.nl
- create or update `stations.vrt` for OGR mapping
- use stations2postgis.sh to map to PostGIS table
- use stations2gml.sh to map to GML file

See details in GitHub: https://github.com/Geonovum/sospilot/tree/master/data/rivm-lml/stations

Test first by uploading and viewing in a geoviewer, for example in http://kadviewer.kademo.nl See result.

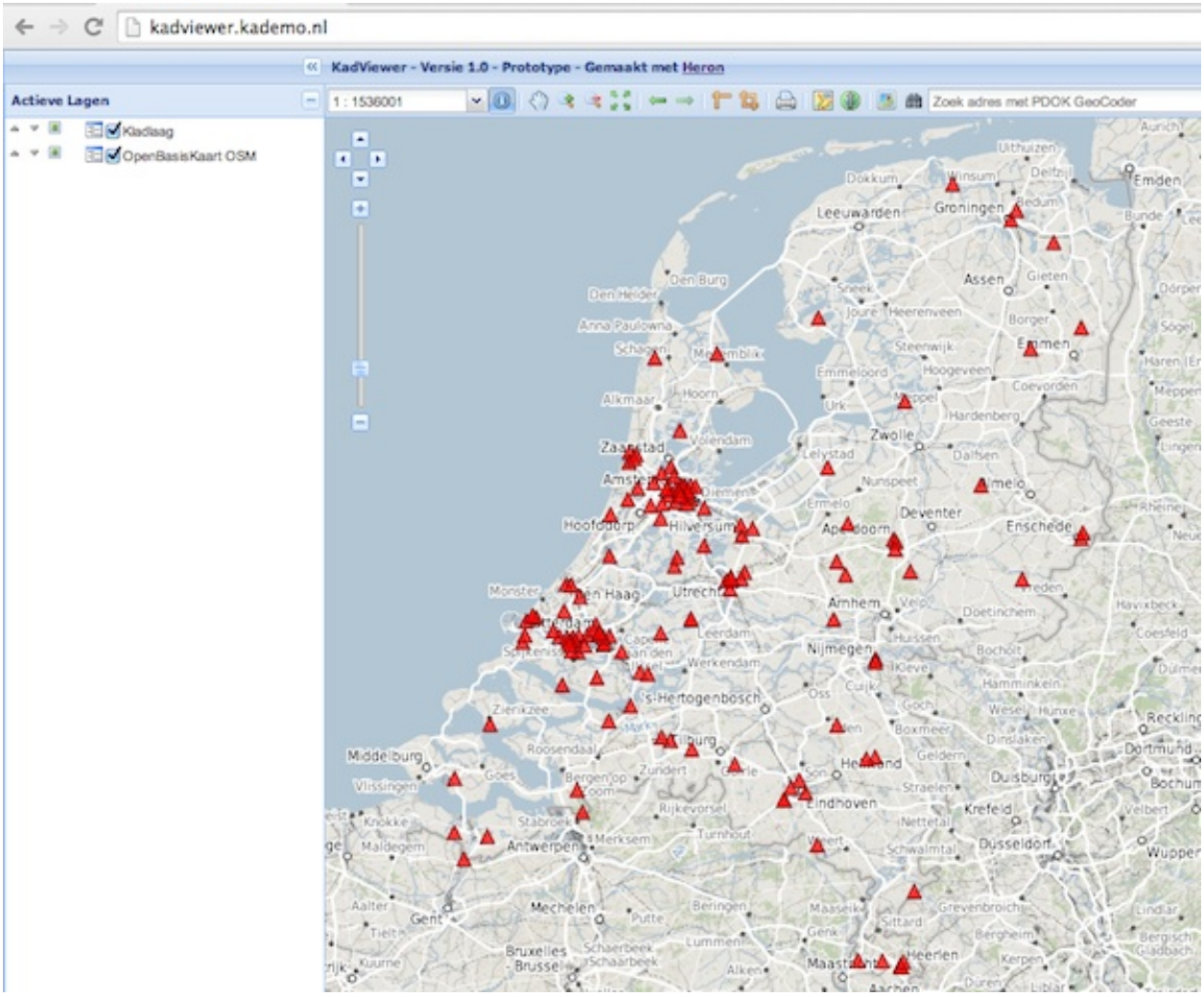


Fig. 2.5: *Figure - RIVM Eionet Stations uploaded/viewed in Heron-based Viewer*

Reading into PostGIS

Note that not all stations may be active. This is indicated by the `activity_end` column. We create a VIEW with only active stations to be used for SOS Sensors (see below).

```
-- create a view with active stations
DROP VIEW IF EXISTS rivm_lml.active_stations CASCADE;
```

lmin PostgreSQL sensors rivm_lml stations

1 2 3 4 5 Next > Last >>

gid	point	local_id	natl_station_code	eu_station_code	municipality	altitude	altitude_un
1	0101000020E6100000970688FF4C9D1940725A0940D0314A40	STA-NL00807	807	NL00807	Hellendoorn	7	m
2	0101000020E61000008E63B9DF80121840336CF71FD7534A40	STA-NL00818	818	NL00818	Steenwijkerland	1	m
3	0101000020E61000002E1A321EA5321B408081204086924A40	STA-NL00913	913	NL00913		1	m
4	0101000020E61000000EBAC65F414B1640CA5DF87F5D754A40	STA-NL00918	918	NL00918		1	m
5	0101000020E61000002041F163CCAD1A403B55BE6724684A40	STA-NL00928	928	NL00928		17	m
6	0101000020E6100000DE9D4A402B1C19406503020074AA4A40	STA-NL00934	934	NL00934	Kollumerland en Nieuwkruisland	1	m
7	0101000020E61000008EA6C4BF196A174031C7FA1F43C54940	STA-NL00131	131	NL00131	Venray	28	m
8	0101000020E6100000596ABDDF68871740D675036097734940	STA-NL00133	133	NL00133	Nuth	96	m
9	0101000020E6100000D41055F8330C1740E4141DC9E5734940	STA-NL00134	134	NL00134		111	m
10	0101000020E6100000DF878384283F16400AD7A3703DA24940	STA-NL00227	227	NL00227		32	m
11	0101000020E6100000F1C740E079981440E4C8FB1F7DC24940	STA-NL00230	230	NL00230	Hilvarenbeek	15	m
12	0101000020E6100000B9E177D32DBB134036E84B6F7FC84940	STA-NL00231	231	NL00231		3	m
13	0101000020E6100000EF53556820A6164012F6ED2422D24940	STA-NL00232	232	NL00232		20	m
14	0101000020E61000001570CFF3A78D11409C4EB2D5E5AE4940	STA-NL00234	234	NL00234		16	m

Fig. 2.6: *Figure - RIVM Eionet Stations Read into Postgres/PostGIS*

```
CREATE VIEW rivm_lml.active_stations AS
   SELECT * FROM rivm_lml.stations WHERE activity_end is NULL;
```

Measurements

Reading raw measurements from the files stored in the `lml_files` table is done with a `Stetl` process. A specific Stetl Input module was developed to effect reading and parsing the files and tracking the last id of the file processed. https://github.com/Geonovum/sospilot/blob/master/src/rivm-lml/lmlfiledbinput.py

The Stetl process is defined in https://github.com/Geonovum/sospilot/blob/master/src/rivm-lml/files2measurements.cfg

The invokation of that Stetl process is via shell script: https://github.com/Geonovum/sospilot/blob/master/src/rivm-lml/files2measurements.sh

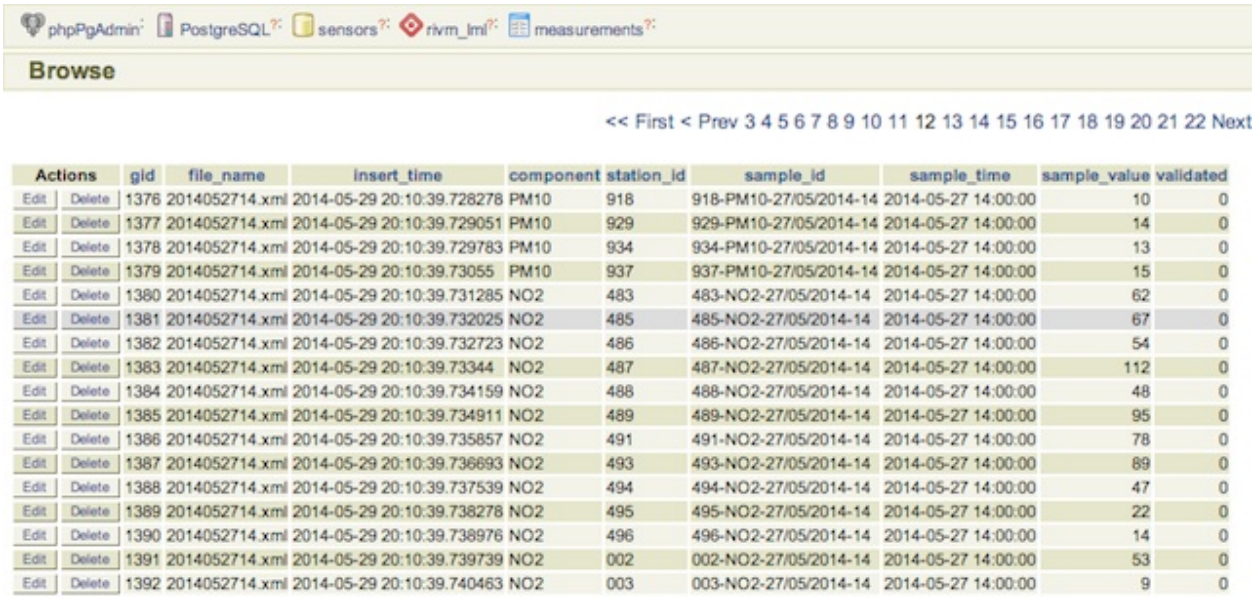
The data is stored in the `measurements` table, as below. `station_id` is a foreign key into the `stations` table.

Tracking ETL progress for the worker `files2measurements` is done in the `etl_progress` table. The `last_id` field is the identifier of the last record in the `lml_files` table processed. On each new run the ETL process starts from new records since that last record.

Some query examples:

```
-- Laatste 24 uur aan metingen voor station en component
SELECT  * FROM  rivm_lml.measurements
   WHERE sample_time >  current_timestamp::timestamp without time zone - '1 day
↪'::INTERVAL
      AND component = 'NO' AND station_id = '136' order by sample_time desc;

-- Laatste meting voor station en component
 SELECT  * FROM  rivm_lml.measurements
   WHERE sample_time >  current_timestamp::timestamp without time zone - '1 day
↪'::INTERVAL
      AND component = 'NO' AND station_id = '136' order by sample_time desc limit 1;
```


phpPgAdmin: PostgreSQL: sensors: rivm_lml: measurements:

Browse

<< First < Prev 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 Next

Actions	gid	file_name	insert_time	component	station_id	sample_id	sample_time	sample_value	validated
Edit Delete	1376	2014052714.xml	2014-05-29 20:10:39.728278	PM10	918	918-PM10-27/05/2014-14	2014-05-27 14:00:00	10	0
Edit Delete	1377	2014052714.xml	2014-05-29 20:10:39.729051	PM10	929	929-PM10-27/05/2014-14	2014-05-27 14:00:00	14	0
Edit Delete	1378	2014052714.xml	2014-05-29 20:10:39.729783	PM10	934	934-PM10-27/05/2014-14	2014-05-27 14:00:00	13	0
Edit Delete	1379	2014052714.xml	2014-05-29 20:10:39.73055	PM10	937	937-PM10-27/05/2014-14	2014-05-27 14:00:00	15	0
Edit Delete	1380	2014052714.xml	2014-05-29 20:10:39.731285	NO2	483	483-NO2-27/05/2014-14	2014-05-27 14:00:00	62	0
Edit Delete	1381	2014052714.xml	2014-05-29 20:10:39.732025	NO2	485	485-NO2-27/05/2014-14	2014-05-27 14:00:00	67	0
Edit Delete	1382	2014052714.xml	2014-05-29 20:10:39.732723	NO2	486	486-NO2-27/05/2014-14	2014-05-27 14:00:00	54	0
Edit Delete	1383	2014052714.xml	2014-05-29 20:10:39.73344	NO2	487	487-NO2-27/05/2014-14	2014-05-27 14:00:00	112	0
Edit Delete	1384	2014052714.xml	2014-05-29 20:10:39.734159	NO2	488	488-NO2-27/05/2014-14	2014-05-27 14:00:00	48	0
Edit Delete	1385	2014052714.xml	2014-05-29 20:10:39.734911	NO2	489	489-NO2-27/05/2014-14	2014-05-27 14:00:00	95	0
Edit Delete	1386	2014052714.xml	2014-05-29 20:10:39.735857	NO2	491	491-NO2-27/05/2014-14	2014-05-27 14:00:00	78	0
Edit Delete	1387	2014052714.xml	2014-05-29 20:10:39.736693	NO2	493	493-NO2-27/05/2014-14	2014-05-27 14:00:00	89	0
Edit Delete	1388	2014052714.xml	2014-05-29 20:10:39.737539	NO2	494	494-NO2-27/05/2014-14	2014-05-27 14:00:00	47	0
Edit Delete	1389	2014052714.xml	2014-05-29 20:10:39.738278	NO2	495	495-NO2-27/05/2014-14	2014-05-27 14:00:00	22	0
Edit Delete	1390	2014052714.xml	2014-05-29 20:10:39.738976	NO2	496	496-NO2-27/05/2014-14	2014-05-27 14:00:00	14	0
Edit Delete	1391	2014052714.xml	2014-05-29 20:10:39.739739	NO2	002	002-NO2-27/05/2014-14	2014-05-27 14:00:00	53	0
Edit Delete	1392	2014052714.xml	2014-05-29 20:10:39.740463	NO2	003	003-NO2-27/05/2014-14	2014-05-27 14:00:00	9	0

Fig. 2.7: *Figure - LML raw measurements stored in Postgres*


phpPgAdmin: PostgreSQL: sensors: rivm_lml: etl_progress:

Browse

Actions	gid	worker	source_table	last_id	last_update
Edit Delete	2	measurements2sos	measurements	-1	2014-05-29 21:20:43.020645
Edit Delete	1	files2measurements	lml_files	673	2014-05-29 21:30:18.535398

Fig. 2.8: *Figure - LML ETL Progress Tracked in Postgres*

2.2.3 ETL Step 3 - SOS Publication

In this step the Raw Measurements data (see Step 2) is transformed to "SOS Ready Data", i.e. data that can be handled by the 52North SOS server. Three options:

1. direct transform into the SOS database of the 52N SOS server
2. via "SOS Transactions" i.e. publishing via SOS-protocol (ala WFS-T)
3. via REST

Discussion:

1. Direct publication into the SOS DB (39 tables!) seems to be cumbersome and error prone and not future-proof
2. via "SOS Transactions" (SOS-T) seems a good and standard option
3. Using the REST-API seems the quickest/most efficient way to go, but the status of the REST implementation is unsure.

So from here on publication via SOS-T is further expanded.

SOS Transaction - PoC

A small Proof-of-Concept using the available requests and sensor ML as example was quite promising. This also provides an example for the mapping strategy.

We have created JSON `insert-sensor` and `insert-observation` requests and executed these in the Admin SOS webclient. Each Sensor denotes a single station with Input just "Air" and one Output for each chemical Component (here O3, MO, NO2, PM10). These files can serve later as templates for the ETL via Stetl. The `insert-sensor` needs to be done once per Station before invoking any `InsertObservation`. The `insert-observation` is performed per measurement, though we may consider using an `insert-result-template` and then `insert-result` or `SOS-Batch` operations for efficiency.

See the images below.

And the observation insert below.

SOS Publication - Stetl Strategy

As Stetl only supports WFS-T, not yet SOS, a SOS Output module `sosoutput.py` was developed derived from the standard `httpoutput.py` module. See https://github.com/Geonovum/sospilot/blob/master/src/rivm-lml/sosoutput.py.

Most importantly, the raw RIVM-LML data from Step 2 needs to be transformed to OWS O&M data. The easiest is to use `substitutable templates`, like the Stetl config itself also applies. This means we develop files with SOS Requests in which all variable parts get a symbolic value like `{sample_value}`. These templates can be found under https://github.com/Geonovum/sospilot/tree/master/src/rivm-lml/sostemplates in particular

- https://github.com/Geonovum/sospilot/blob/master/src/rivm-lml/sostemplates/insert-sensor.json InsertSensor
- https://github.com/Geonovum/sospilot/blob/master/src/rivm-lml/sostemplates/procedure-desc.xml Sensor ML
- https://github.com/Geonovum/sospilot/blob/master/src/rivm-lml/sostemplates/insert-observation.json InsertObservation

These templates were derived from the sample SOS requests available in the 52N SOS Admin Client. Note that we use JSON for the requests, as this is simpler than XML. The Sensor ML is embedded in the insert-sensor JSON request.

http://sensors.geonovum.nl/sos/service

Request

POST | application/json | application/json | Permalink | Syntax


```
{
  "request": "InsertSensor",
  "service": "SOS",
  "version": "2.0.0",
  "procedureDescriptionFormat": "http://www.opengis.net/sensorML/1.0.1",
  "procedureDescription": "<sml:SensorML xmlns:swes=\"http://www.opengis.net/swes/2.0\"
xmlns:sos=\"http://www.opengis.net/sos/2.0\" xmlns:swe=\"http://www.opengis.net/swe/1.0.1\"
xmlns:sml=\"http://www.opengis.net/sensorML/1.0.1\" xmlns:gml=\"http://www.opengis.net/gml\"
xmlns:xlink=\"http://www.w3.org/1999/xlink\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
version=\"1.0.1\"><sml:member><sml:System><sml:identification><sml:IdentifierList><sml:identifier
name=\"uniqueID\"><sml:Term definition=\"urn:ogc:def:identifier:OGC:1.0:uniqueID\">
<sml:value>urn:ogc:object:feature:Sensor:RIVM:rivm-sensor-807</sml:value></sml:Term></sml:identifier>
<sml:identifier name=\"longName\"><sml:Term definition=\"urn:ogc:def:identifier:OGC:1.0:longName\">
<sml:value>RIVM Rijksinstituut voor Volksgezondheid en Milieu (http://www.rivm.nl) - Station
Hellendoorn</sml:value></sml:Term></sml:identifier><sml:identifier name=\"shortName\"><sml:Term
definition=\"urn:ogc:def:identifier:OGC:1.0:shortName\"><sml:value>RIVM - Station Hellendoorn</sml:value>
</sml:Term></sml:identifier></sml:IdentifierList></sml:identification><sml:capabilities name=\"offerings\">
<swe:SimpleDataRecord><swe:field name=\"Offering for sensor 807\"><swe:Text
definition=\"urn:ogc:def:identifier:OGC:offeringID\"><swe:value>http://sensors.geonovum.nl/rivm-
lml/offering/lml/807</swe:value></swe:Text></swe:field></swe:SimpleDataRecord></sml:capabilities>
<sml:capabilities name=\"featuresOfInterest\"><swe:SimpleDataRecord><swe:field name=\"featureOfInterestID\">
<swe:Text><swe:value>http://sensors.geonovum.nl/rivm-lml/featureOfInterest/9</swe:value></swe:Text></swe:field>
</swe:SimpleDataRecord></sml:capabilities><sml:position name=\"sensorPosition\"><swe:Position
referenceFrame=\"urn:ogc:def:crs:EPSG::4326\"><swe:location><swe:Vector gml:id=\"STATION_LOCATION\">
<swe:coordinate name=\"easting\"><swe:Quantity axisID=\"x\"><swe:uom code=\"degree\"/>
<swe:value>6.40361404</swe:value></swe:Quantity></swe:coordinate><swe:coordinate name=\"northing\">
<swe:Quantity axisID=\"y\"><swe:uom code=\"degree\"/><swe:value>52.38916779</swe:value></swe:Quantity>
</swe:coordinate><swe:coordinate name=\"altitude\"><swe:Quantity axisID=\"z\"><swe:uom code=\"m\"/>
<swe:value>52.0</swe:value></swe:Quantity></swe:coordinate></swe:Vector></swe:location></swe:Position>
</sml:position><sml:inputs><sml:InputList><sml:input name=\"air\"><swe:ObservableProperty
definition=\"http://sensors.geonovum.nl/rivm-lml/observableProperty/air\"/></sml:input></sml:InputList>
</sml:inputs><sml:outputs><sml:OutputList><sml:output name=\"value_NO\"><swe:Quantity
definition=\"http://sensors.geonovum.nl/rivm-lml/observableProperty/NO\"><swe:uom code=\"ug/m3\"/>
</swe:Quantity></sml:output><sml:output name=\"value_NO2\"><swe:Quantity
definition=\"http://sensors.geonovum.nl/rivm-lml/observableProperty/NO2\"><swe:uom code=\"ug/m3\"/>
</swe:Quantity></sml:output><sml:output name=\"value_O3\"><swe:Quantity
definition=\"http://sensors.geonovum.nl/rivm-lml/observableProperty/O3\"><swe:uom code=\"ug/m3\"/>
</swe:Quantity></sml:output><sml:output name=\"value_PM10\"><swe:Quantity
definition=\"http://sensors.geonovum.nl/rivm-lml/observableProperty/PM10\"><swe:uom code=\"ug/m3\"/>
</swe:Quantity></sml:output></sml:OutputList></sml:outputs></sml:System></sml:member></sml:SensorML>",
  "observableProperty": [
    "http://sensors.geonovum.nl/rivm-lml/observableProperty/NO",
    "http://sensors.geonovum.nl/rivm-lml/observableProperty/NO2",
    "http://sensors.geonovum.nl/rivm-lml/observableProperty/O3",
    "http://sensors.geonovum.nl/rivm-lml/observableProperty/PM10"
  ],
  "observationType": [
    "http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement"
  ],
  "featureOfInterestType": "http://www.opengis.net/def/samplingFeatureType/OGC-OM/2.0/SF_SamplingPoint"
}
```


Send

Response

```
200 OK
Date: Wed, 04 Jun 2014 15:32:32 GMT
Content-Encoding: gzip
Server: Apache/2.4.7
Connection: Keep-Alive
Keep-Alive: timeout=5, max=100
Content-Length: 175
Content-Type: application/json;charset=UTF-8
```


```
{
  "request" : "InsertSensor",
  "version" : "2.0.0",
  "service" : "SOS",
  "assignedProcedure" : "urn:ogc:object:feature:Sensor:RIVM:rivm-sensor-807",
  "assignedOffering" : "http://sensors.geonovum.nl/rivm-lml/offering/lml/807"
}
```


Fig. 2.9: *Figure - Inserting a Station as sensor definition using SOS via 52N SOS Admin webclient*

http://sensors.geonovum.nl/sos/service

Request

POST | application/json | application/json | Permalink | Syntax

```
{
  "request": "InsertObservation",
  "service": "SOS",
  "version": "2.0.0",
  "offering": "http://sensors.geonovum.nl/rivm-lml/offering/lml/807",
  "observation": {
    "identifier": {
      "value": "http://sensors.geonovum.nl/rivm-lml/observation/807/9",
      "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown"
    },
    "type": "http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement",
    "procedure": "urn:ogc:object:feature:Sensor:RIVM:rivm-sensor-807",
    "observedProperty": "http://sensors.geonovum.nl/rivm-lml/observableProperty/O3",
    "featureOfInterest": {
      "identifier": {
        "value": "http://sensors.geonovum.nl/rivm-lml/featureOfInterest/9",
        "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown"
      },
      "name": [
        {
          "value": "52°North",
          "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown"
        }
      ],
      "sampledFeature": [
        "http://sensors.geonovum.nl/rivm-lml/featureOfInterest/world"
      ],
      "geometry": {
        "type": "Point",
        "coordinates": [
          52.38916779,
          6.40361404
        ],
        "crs": {
          "type": "name",
          "properties": {
            "name": "EPSG:4326"
          }
        }
      }
    },
    "phenomenonTime": "012-11-19T17:45:15+00:00",
    "resultTime": "012-11-19T17:45:15+00:00",
    "result": {
      "uom": "ug/m3",
      "value": 42
    }
  }
}
```

Send

Response

```
200 OK
Date: Wed, 04 Jun 2014 15:36:32 GMT
Content-Encoding: gzip
Server: Apache/2.4.7
Connection: Keep-Alive
Keep-Alive: timeout=5, max=100
Content-Length: 84
Content-Type: application/json;charset=UTF-8
```

```
{
  "request" : "InsertObservation",
  "version" : "2.0.0",
  "service" : "SOS"
}
```

Fig. 2.10: *Figure - Inserting a single measured value (O3) as an Observation as using SOS via 52N SOS Admin webclient*

SOS Publication - Sensors

This step needs to be performed only once, or when any of the original Station data (CSV) changes.

The Stetl config https://github.com/Geonovum/sospilot/blob/master/src/rivm-lml/stations2sensors.cfg uses a Standard Stetl module, `inputs.dbinput.PostgresDbInput` for obtaining Record data from a Postgres database.

```
{{
  "request": "InsertSensor",
  "service": "SOS",
  "version": "2.0.0",
  "procedureDescriptionFormat": "http://www.opengis.net/sensorML/1.0.1",
  "procedureDescription": "{procedure-desc.xml}",
   "observableProperty": [
    "http://sensors.geonovum.nl/rivm-lml/observableProperty/benzeen",
    "http://sensors.geonovum.nl/rivm-lml/observableProperty/CO",
    "http://sensors.geonovum.nl/rivm-lml/observableProperty/NH3",
    "http://sensors.geonovum.nl/rivm-lml/observableProperty/NO",
    "http://sensors.geonovum.nl/rivm-lml/observableProperty/NO2",
    "http://sensors.geonovum.nl/rivm-lml/observableProperty/O3",
    "http://sensors.geonovum.nl/rivm-lml/observableProperty/PM10",
    "http://sensors.geonovum.nl/rivm-lml/observableProperty/PM25",
    "http://sensors.geonovum.nl/rivm-lml/observableProperty/SO2"
  ],
  "observationType": [
    "http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement"
  ],
  "featureOfInterestType": "http://www.opengis.net/def/samplingFeatureType/OGC-OM/2.0/
↪SF_SamplingPoint"
}}
```

The SOSTOutput module will expand `{procedure-desc.xml}` with the Sensor ML template.

SOS Publication - Observations

The Stetl config https://github.com/Geonovum/sospilot/blob/master/src/rivm-lml/measurements2sos.cfg uses an extended Stetl module (`inputs.dbinput.PostgresDbInput`) for obtaining Record data from a Postgres database: https://github.com/Geonovum/sospilot/blob/master/src/rivm-lml/measurementsdbinput.py. This is required to track progress in the `etl_progress` table similar as in Step 2. The `last_id` is remembered.

The Observation template looks as follows.

```
{{
   "request": "InsertObservation",
   "service": "SOS",
   "version": "2.0.0",
   "offering": "http://sensors.geonovum.nl/rivm-lml/offering/{station_id}",
   "observation": {{
     "identifier": {{
       "value": "{unique_id}",
       "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown"
     }},
     "type": "http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement",
     "procedure": "http://sensors.geonovum.nl/rivm-lml/procedure/{station_id}",
     "observedProperty": "http://sensors.geonovum.nl/rivm-lml/observableProperty/
↪{component}",
     "featureOfInterest": {{
       "identifier": {{
```

```
          "value": "http://sensors.geonovum.nl/rivm-lml/featureOfInterest/{station_id}
↪",
          "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown"
        }},
        "name": [
          {{
            "value": "{municipality}",
            "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown"
          }}
        ],
        "geometry": {{
          "type": "Point",
          "coordinates": [
            {station_lat},
            {station_lon}
          ],
          "crs": {{
            "type": "name",
            "properties": {{
              "name": "EPSG:4326"
            }}
          }}
        }}
      }},
      "phenomenonTime": "{sample_time}",
      "resultTime": "{sample_time}",
      "result": {{
        "uom": "ug/m3",
        "value": {sample_value}
      }}
    }}
}}
```

It is quite trivial in `sosoutput.py` to substitute these values from the `measurements`-table records.

Like in ETL Step 2 the progress is remembered in the table `rivm_lml.etl_progress` by updating the `last_id` field after publication, where that value represents the `gid` value of `rivm_lml.measurements`.

SOS Publication - Results

We can observe the database being filled:

Via the standard SOS protocol the results can be tested:

- GetCapabilities: http://sensors.geonovum.nl/sos/service?service=SOS&request=GetCapabilities
- DescribeSensor (station 807, Hellendoorn): http://tinyurl.com/mmsr9hl (URL shortened)
- GetObservation: http://tinyurl.com/ol82sxv (URL shortened)

REST API

For now the REST API will **not** be used since SOS-T is used (see above). Below is for possible future reference.

Documentation REST API: http://52north.org/files/sensorweb/docs/sos/restful/restful_sos_documentation.pdf

REST root URL: http://sensors.geonovum.nl/sos/service/rest

From the documentation the mapping seems to make sense as follows:

☐	numericvalue	sensors		97450	Browse
☐	observableproperty	sensors		1260	Browse
☐	observation	sensors		97450	Browse
☐	observationconstellation	sensors		1260	Browse
☐	observationhasoffering	sensors		97450	Browse
☐	observationtype	sensors		0	Browse
☐	offering	sensors		140	Browse
☐	offeringallowedfeaturetype	sensors		140	Browse
☐	offeringallowedobservationtype	sensors		140	Browse
☐	offeringhasrelatedfeature	sensors		0	Browse
☐	parameter	sensors		0	Browse
☐	procedure	sensors		140	Browse

Fig. 2.11: *Figure - SOS server database being filled: 140 Sensors (Stations) about 100000 Observations inserted*

- `sensor-create` - to create new sensor resources –> map from `stations` table
- `observation-create` - to create observation resources –> map from `measurements` table

Design:

- use Stetl: input Postgres Query, output SOS-REST module
- similar to ETL step 2
- track progress in `etl_progress` table
- new Stetl output, similar to WFS-T and deegree-publisher
- use Python XML templates for the requests
- problem: make SML, Sensor per Station, or Sensor per Station-Component ?

CHAPTER 3

Web Services

This chapter describes how OGC OWS web services are realized on top of the converted/transformed data as described in the data chapter. In particular:

- WFS and WMS-Time services
- OWS SOS service

3.1 Architecture

Figure 1 sketches the overall SOSPilot architecture with emphasis on the flow of data (arrows). Circles depict harvesting/ETL processes. Server-instances are in rectangles. Datastores the "DB"-icons.

3.2 WFS and WMS Services

This describes the realization of WFS and WMS(-Time) in GeoServer. GeoServer can directly generate OWS services from a Postgres/PostGIS datasource as one Layer per Table (or View). For our purpose the source tables are the `stations` table and `measurements` tables in the Postgres schema `rivm-lml`. See the data chapter for info how these tables are continuously populated from the raw AQ data via ETL Step 1 and Step 2.

3.2.1 Database VIEWs

As WFS and WMS always need a geometry-column, we will JOIN the `stations` table and the `measurements` tables to create 'Postgres VIEWs'. A VIEW is basically a query but presented as a database table. Data in the VIEW is always current with the original tables (or VIEWs, as we may derive VIEWs from VIEWs). This way data selections can be provided permanently and tailored to the particular OWS service.

Having all data (stations, measurements) stored PostgreSQL tables gives rise to many possibilities for selection from these two tables. The `stations` table looks as follows.

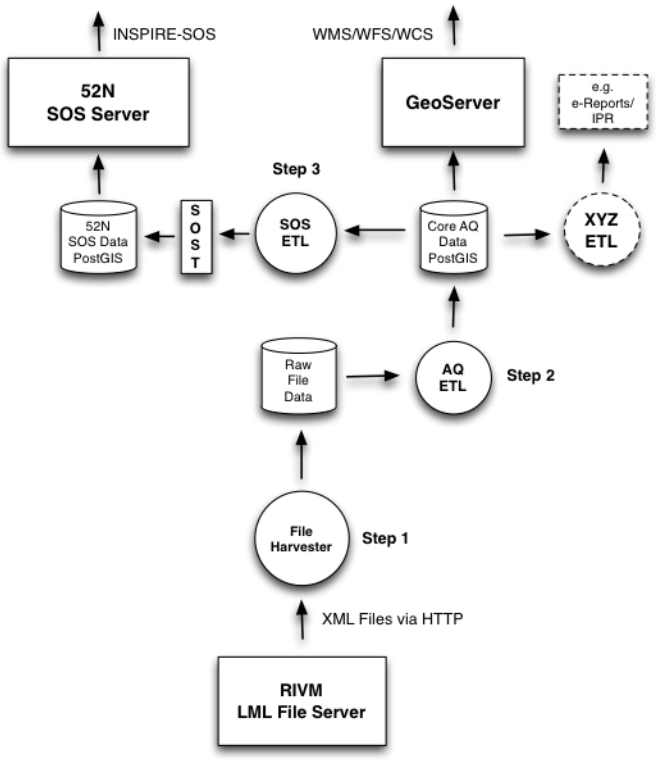
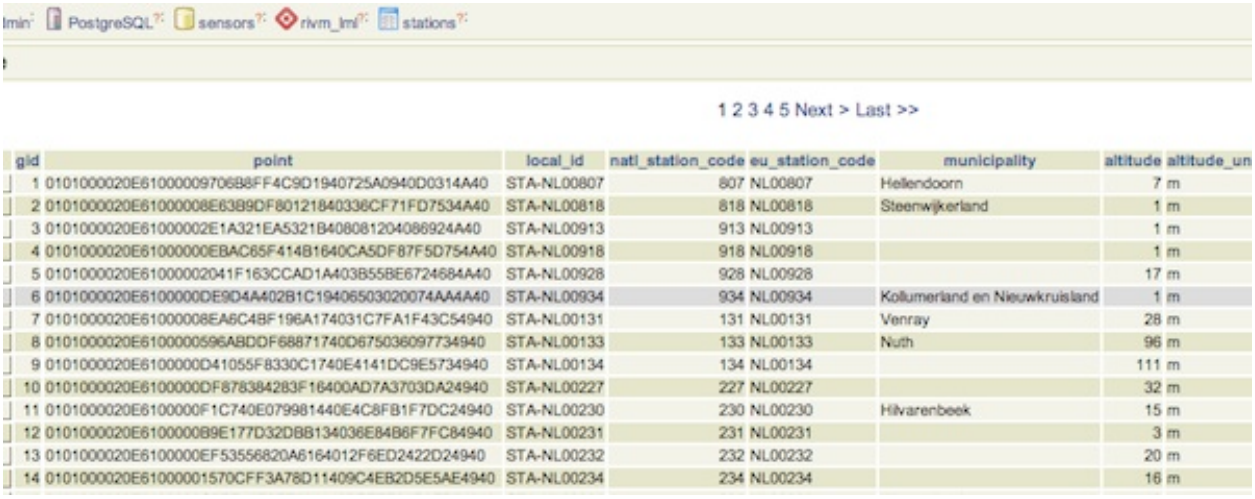


Fig. 3.1: *Figure 1 - Overall Architecture*

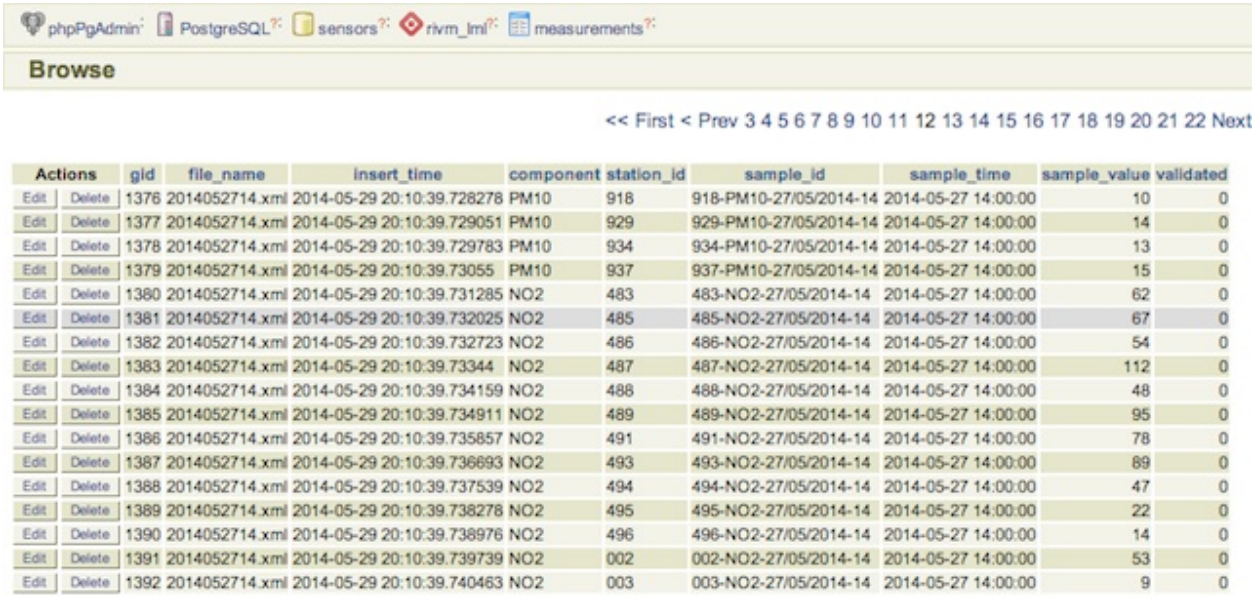


1 2 3 4 5 Next > Last >>

gid	point	local_id	natl_station_code	eu_station_code	municipality	altitude	altitude_un
1	0101000020E6100000970688FF4C9D1940725A0940D0314A40	STA-NL00807	807	NL00807	Hellendoorn	7	m
2	0101000020E61000008E63B9DF80121840336CF71FD7534A40	STA-NL00818	818	NL00818	Steenwijkerland	1	m
3	0101000020E61000002E1A321EA5321B4080812040869 24A40	STA-NL00913	913	NL00913		1	m
4	0101000020E61000000EBAC65F414B1640CA5DF87F5D754A40	STA-NL00918	918	NL00918		1	m
5	0101000020E61000002041F163CCAD1A403B55BE6724684A40	STA-NL00928	928	NL00928		17	m
6	0101000020E6100000DE9D4A402B1C19406503020074AA4A40	STA-NL00934	934	NL00934	Kollumerland en Nieuwkruisland	1	m
7	0101000020E61000008EA6C4BF196A174031C7FA1F43C54940	STA-NL00131	131	NL00131	Venray	28	m
8	0101000020E6100000596ABDDF68871740D67503609773494 0	STA-NL00133	133	NL00133	Nuth	96	m
9	0101000020E6100000D41055F8330C1740E4141DC9E5734940	STA-NL00134	134	NL00134		111	m
10	0101000020E6100000DF878384283F16400AD7A3703DA24940	STA-NL00227	227	NL00227		32	m
11	0101000020E6100000F1C740E079981440E4C8FB1F7DC24940	STA-NL00230	230	NL00230	Hilvarenbeek	15	m
12	0101000020E6100000B9E177D32DBB134036E84B6F7FC84940	STA-NL00231	231	NL00231		3	m
13	0101000020E6100000EF53556820A6164012F6ED2422D24940	STA-NL00232	232	NL00232		20	m
14	0101000020E61000001570CFF3A78D11409C4EB2D5E5AE4940	STA-NL00234	234	NL00234		16	m

Fig. 3.2: *Figure - RIVM Eionet Stations Read into Postgres/PostGIS*

The sample-data is stored in the `measurements` table, as below. `station_id` is a foreign key (though not strictly, as new stations may pop-up) into the `stations` table.


phpPgAdmin PostgreSQL sensors rivm_lml measurements

Browse

<< First < Prev 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 Next

Actions	gid	file_name	insert_time	component	station_id	sample_id	sample_time	sample_value	validated
Edit Delete	1376	2014052714.xml	2014-05-29 20:10:39.728278	PM10	918	918-PM10-27/05/2014-14	2014-05-27 14:00:00	10	0
Edit Delete	1377	2014052714.xml	2014-05-29 20:10:39.729051	PM10	929	929-PM10-27/05/2014-14	2014-05-27 14:00:00	14	0
Edit Delete	1378	2014052714.xml	2014-05-29 20:10:39.729783	PM10	934	934-PM10-27/05/2014-14	2014-05-27 14:00:00	13	0
Edit Delete	1379	2014052714.xml	2014-05-29 20:10:39.73055	PM10	937	937-PM10-27/05/2014-14	2014-05-27 14:00:00	15	0
Edit Delete	1380	2014052714.xml	2014-05-29 20:10:39.731285	NO2	483	483-NO2-27/05/2014-14	2014-05-27 14:00:00	62	0
Edit Delete	1381	2014052714.xml	2014-05-29 20:10:39.732025	NO2	485	485-NO2-27/05/2014-14	2014-05-27 14:00:00	67	0
Edit Delete	1382	2014052714.xml	2014-05-29 20:10:39.732723	NO2	486	486-NO2-27/05/2014-14	2014-05-27 14:00:00	54	0
Edit Delete	1383	2014052714.xml	2014-05-29 20:10:39.73344	NO2	487	487-NO2-27/05/2014-14	2014-05-27 14:00:00	112	0
Edit Delete	1384	2014052714.xml	2014-05-29 20:10:39.734159	NO2	488	488-NO2-27/05/2014-14	2014-05-27 14:00:00	48	0
Edit Delete	1385	2014052714.xml	2014-05-29 20:10:39.734911	NO2	489	489-NO2-27/05/2014-14	2014-05-27 14:00:00	95	0
Edit Delete	1386	2014052714.xml	2014-05-29 20:10:39.735857	NO2	491	491-NO2-27/05/2014-14	2014-05-27 14:00:00	78	0
Edit Delete	1387	2014052714.xml	2014-05-29 20:10:39.736693	NO2	493	493-NO2-27/05/2014-14	2014-05-27 14:00:00	89	0
Edit Delete	1388	2014052714.xml	2014-05-29 20:10:39.737539	NO2	494	494-NO2-27/05/2014-14	2014-05-27 14:00:00	47	0
Edit Delete	1389	2014052714.xml	2014-05-29 20:10:39.738278	NO2	495	495-NO2-27/05/2014-14	2014-05-27 14:00:00	22	0
Edit Delete	1390	2014052714.xml	2014-05-29 20:10:39.738976	NO2	496	496-NO2-27/05/2014-14	2014-05-27 14:00:00	14	0
Edit Delete	1391	2014052714.xml	2014-05-29 20:10:39.739739	NO2	002	002-NO2-27/05/2014-14	2014-05-27 14:00:00	53	0
Edit Delete	1392	2014052714.xml	2014-05-29 20:10:39.740463	NO2	003	003-NO2-27/05/2014-14	2014-05-27 14:00:00	9	0

Fig. 3.3: *Figure - LML raw measurements stored in Postgres*

One appearent VIEW is to combine the `measurements` and `stations` tables into a new `measurements_stations` table by means of a JOIN query as follows (`rivm_lml` is the schema name):

```
CREATE VIEW rivm_lml.measurements_stations AS
  SELECT m.gid, m.station_id, s.municipality, m.component, m.sample_time, m.sample_
↪value, s.point, m.validated,
         m.file_name, m.insert_time, m.sample_id,
         s.local_id, s.eu_station_code, s.altitude, s.area_classification,
         s.activity_begin, s.activity_end
         FROM rivm_lml.measurements as m
           INNER JOIN rivm_lml.stations as s ON m.station_id = s.natl_station_code;
```

The data can now be viewed as rows in the `measurements_stations` VIEW, where each sample also has a POINT-geometry:

Still this VIEW is too broad to use for WMS/WFS Layers as we will usually visualize (via WMS) or query (via WFS) a single chemical component. We will use the `measurements-stations` VIEW to derive more specialized, per-component VIEWs Two broad VIEWs are provided first:

- per component last-captured measurements
- per component time-series measurements

These VIEWs are in SQL as below taking two components (NO2 and O3) as example.

```
-- per component last-captured measurements
CREATE VIEW rivm_lml.v_last_measurements_NO2 AS
  SELECT DISTINCT ON (station_id) station_id,
    municipality, sample_time, sample_value, point, validated, gid, sample_id
  FROM rivm_lml.measurements_stations WHERE component = 'NO2' ORDER BY station_id,␣
↪gid DESC;
```

phpPgAdmin PostgreSQL sensors rivm_lml measurements_stations

Browse

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 1

gid	station_id	municipality	component	sample_time	sample_value	point	validated	file_name
1	547	Hilversum	PM10	2014-05-01 20:00:00	37	0101000020E6100000F4177AC4E8B91440CBF8F719171E4A40	0	2014050120.xml
2	549	Laren	PM10	2014-05-01 20:00:00	35	0101000020E6100000F81C588E90F1144037A6272CF1204A40	0	2014050120.xml
3	247	Veldhoven	PM10	2014-05-01 20:00:00	34	0101000020E610000089E45400C6921540747DFEFF28B44940	0	2014050120.xml
5	244	Gemert-Bakel	PM10	2014-05-01 20:00:00	47	0101000020E61000001E19ABCDFF3B174094F6065F98C44940	0	2014050120.xml
6	246	Moerdijk	PM10	2014-05-01 20:00:00	42	0101000020E6100000485146E0380E124058B009408ED34940	0	2014050120.xml
7	319	Vlissingen	PM10	2014-05-01 20:00:00	53	0101000020E610000023111AC1C6F50D40C0E8F2E670BB4940	0	2014050120.xml
8	131	Venray	PM10	2014-05-01 20:00:00	43	0101000020E61000008EA6C4BF196A174031C7FA1F43C54940	0	2014050120.xml
9	133	Nuth	PM10	2014-05-01 20:00:00	38	0101000020E6100000596ABDDF68871740D675036097734940	0	2014050120.xml
10	136	Heerlen	PM10	2014-05-01 20:00:00	39	0101000020E6100000B18717C06BE2174090B402E0AB714940	0	2014050120.xml
11	138		PM10	2014-05-01 20:00:00	19	0101000020E610000084D6C39789F21740348463963D734940	0	2014050120.xml
12	230	Hilvarenbeek	PM10	2014-05-01 20:00:00	48	0101000020E6100000F1C740E079981440E4C8FB1F7DC24940	0	2014050120.xml
13	235	Woensdrecht	PM10	2014-05-01 20:00:00	39	0101000020E6100000C7FE47E0EE701140BFEEF81FAEB74940	0	2014050120.xml
14	236	Eindhoven	PM10	2014-05-01 20:00:00	31	0101000020E6100000CBF14C6021E41540F6B704E01FBC4940	0	2014050120.xml
15	237		PM10	2014-05-01 20:00:00	37	0101000020E61000008104C58F31B7154000C85EEFFEB84940	0	2014050120.xml
16	240		PM10	2014-05-01 20:00:00	49	0101000020E610000007E7D7FFCE4C13406E280860FBCB4940	0	2014050120.xml
17	241	Breda	PM10	2014-05-01 20:00:00	41	0101000020E6100000484FD35F961F1340B30101003ACD4940	0	2014050120.xml
18	318	Terneuzen	PM10	2014-05-01 20:00:00	32	0101000020E6100000CD5299FFE3FE0D405009FC9FCBA54940	0	2014050120.xml
19	418	Rotterdam	PM10	2014-05-01 20:00:00	32	0101000020E6100000B3EAD6DFD0EB1140DE9A09C01EF54940	0	2014050120.xml
20	433	Vlaardingen	PM10	2014-05-01 20:00:00	32	0101000020E6100000BBB9B6BFCC4E1140BCE3FB7FA8F44940	0	2014050120.xml
21	437		PM10	2014-05-01 20:00:00	36	0101000020E6100000A3203660A9CD11406826F7BFCCE44940	0	2014050120.xml
22	442	Dordrecht	PM10	2014-05-01 20:00:00	29	0101000020E61000002C1A420036D51240CCAB00007CE64940	0	2014050120.xml
23	444	Noordwijk	PM10	2014-05-01 20:00:00	25	0101000020E610000090C8EB7F880A124010E0FC9F26264A40	0	2014050120.xml
24	445		PM10	2014-05-01 20:00:00	33	0101000020E6100000C1CAA145B64311409A9999999094A40	0	2014050120.xml
25	446	's-Gravenhage	PM10	2014-05-01 20:00:00	27	0101000020E61000008A4A55A037711140D15BFA9F15054A40	0	2014050120.xml

Fig. 3.4: *Figure - LML Postgres VIEW of combined measurements and stations*

```
CREATE VIEW rivm_lml.v_last_measurements_O3 AS
  SELECT DISTINCT ON (station_id) station_id,
    municipality, sample_time, sample_value, point, validated, gid, sample_id
  FROM rivm_lml.measurements_stations WHERE component = 'O3' ORDER BY station_id, gid␣
↪DESC;

  .
  .

-- per component time-series measurements
CREATE VIEW rivm_lml.v_measurements_NO2 AS
  SELECT  station_id,
    municipality, sample_time, sample_value, point, validated, gid, sample_id
  FROM rivm_lml.measurements_stations WHERE component = 'NO2';

CREATE VIEW rivm_lml.v_measurements_O3 AS
  SELECT  station_id,
    municipality, sample_time, sample_value, point, validated, gid, sample_id
  FROM rivm_lml.measurements_stations WHERE component = 'O3';
  .
  .
```

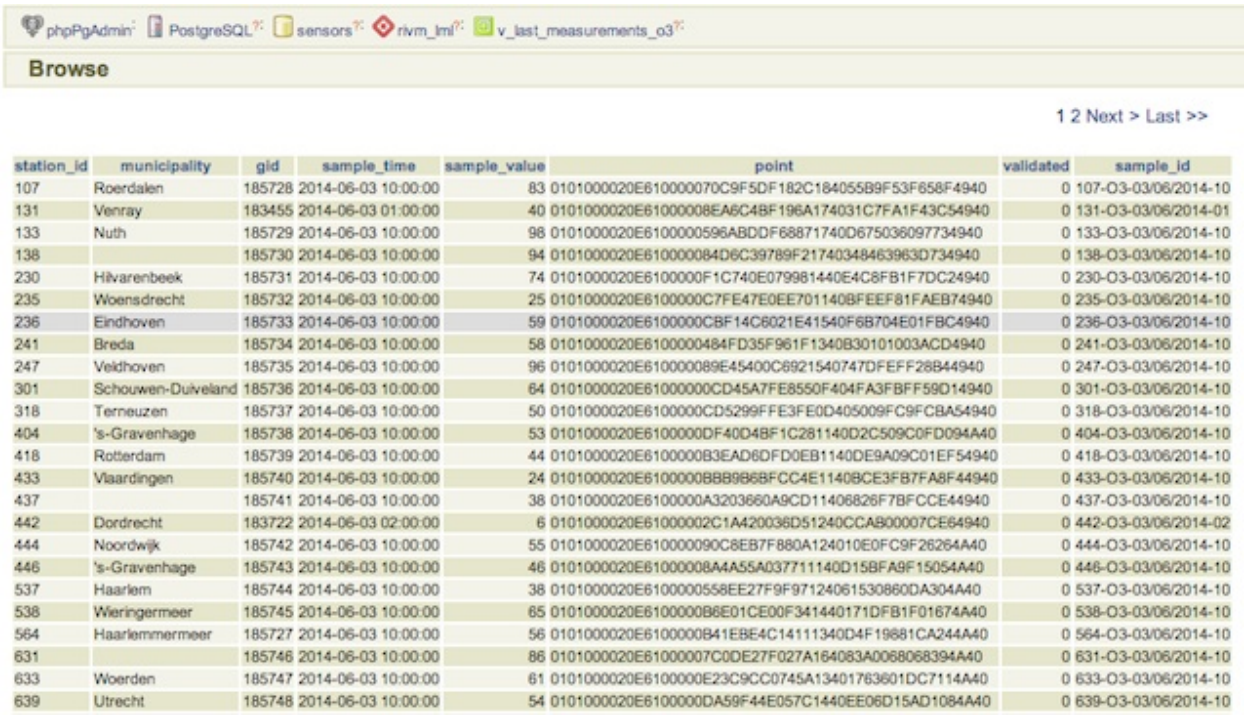
Data from these VIEWs can now be viewed as rows like in this table:

Additional VIEWs

Additional VIEWs for the future can be thought, like:

	View	Owner
☐	measurements_stations	sensors
☐	v_last_measurements_co	sensors
☐	v_last_measurements_nh3	sensors
☐	v_last_measurements_no	sensors
☐	v_last_measurements_no2	sensors
☐	v_last_measurements_o3	sensors
☐	v_last_measurements_pm10	sensors
☐	v_last_measurements_so2	sensors
☐	v_measurements_co	sensors
☐	v_measurements_nh3	sensors
☐	v_measurements_no	sensors
☐	v_measurements_no2	sensors
☐	v_measurements_o3	sensors
☐	v_measurements_pm10	sensors
☐	v_measurements_so2	sensors

Fig. 3.5: *Figure - All database views*



phpPgAdmin PostgreSQL sensors rivm_lml v_last_measurements_o3

Browse

1 2 Next > Last >>

station_id	municipality	gid	sample_time	sample_value	point	validated	sample_id
107	Roerdalen	185728	2014-06-03 10:00:00	83	0101000020E610000070C9F5DF182C184055B9F53F658F4940	0	107-O3-03/06/2014-10
131	Venray	183455	2014-06-03 01:00:00	40	0101000020E61000008EA6C4BF196A174031C7FA1F43C54940	0	131-O3-03/06/2014-01
133	Nuth	185729	2014-06-03 10:00:00	98	0101000020E6100000596ABDDF68871740D675036097734940	0	133-O3-03/06/2014-10
138		185730	2014-06-03 10:00:00	94	0101000020E610000084D6C39789F21740348463963D734940	0	138-O3-03/06/2014-10
230	Hilvarenbeek	185731	2014-06-03 10:00:00	74	0101000020E6100000F1C740E079981440E4C8FB1F7DC24940	0	230-O3-03/06/2014-10
235	Woensdrecht	185732	2014-06-03 10:00:00	25	0101000020E6100000C7FE47E0EE701140BFEEF81FAEB74940	0	235-O3-03/06/2014-10
236	Eindhoven	185733	2014-06-03 10:00:00	59	0101000020E6100000CBF14C6021E41540F6B704E01FBC4940	0	236-O3-03/06/2014-10
241	Breda	185734	2014-06-03 10:00:00	58	0101000020E6100000484FD35F961F1340B30101003ACD4940	0	241-O3-03/06/2014-10
247	Veldhoven	185735	2014-06-03 10:00:00	96	0101000020E610000089E45400C6921540747DFEFF28B44940	0	247-O3-03/06/2014-10
301	Schouwen-Duiveland	185736	2014-06-03 10:00:00	64	0101000020E61000000CD45A7FE8550F404FA3FBFF59D14940	0	301-O3-03/06/2014-10
318	Terneuzen	185737	2014-06-03 10:00:00	50	0101000020E6100000CD5299FFE3FE0D405009FC9FCBA54940	0	318-O3-03/06/2014-10
404	's-Gravenhage	185738	2014-06-03 10:00:00	53	0101000020E6100000DF40D4BF1C281140D2C509C0FD094A40	0	404-O3-03/06/2014-10
418	Rotterdam	185739	2014-06-03 10:00:00	44	0101000020E6100000B3EAD6DFD0EB1140DE9A09C01EF54940	0	418-O3-03/06/2014-10
433	Vlaardingen	185740	2014-06-03 10:00:00	24	0101000020E6100000BBB9B6BFCC4E1140BCE3FB7FA8F44940	0	433-O3-03/06/2014-10
437		185741	2014-06-03 10:00:00	38	0101000020E6100000A3203660A9CD11406826F7BFCCE44940	0	437-O3-03/06/2014-10
442	Dordrecht	183722	2014-06-03 02:00:00	6	0101000020E61000002C1A420036D51240CCAB00007CE64940	0	442-O3-03/06/2014-02
444	Noordwijk	185742	2014-06-03 10:00:00	55	0101000020E610000090C8EB7F880A124010E0FC9F26264A40	0	444-O3-03/06/2014-10
446	's-Gravenhage	185743	2014-06-03 10:00:00	46	0101000020E61000008A4A55A037711140D15BFA9F15054A40	0	446-O3-03/06/2014-10
537	Haarlem	185744	2014-06-03 10:00:00	38	0101000020E6100000558EE27F9F97124061530860DA304A40	0	537-O3-03/06/2014-10
538	Wieringermeer	185745	2014-06-03 10:00:00	65	0101000020E6100000B6E01CE00F341440171DFB1F01674A40	0	538-O3-03/06/2014-10
564	Haarlemmermeer	185727	2014-06-03 10:00:00	56	0101000020E6100000B41EBE4C14111340D4F19681CA244A40	0	564-O3-03/06/2014-10
631		185746	2014-06-03 10:00:00	86	0101000020E61000007C0DE27F027A164083A0068068394A40	0	631-O3-03/06/2014-10
633	Woerden	185747	2014-06-03 10:00:00	61	0101000020E6100000E23C9CC0745A13401763601DC7114A40	0	633-O3-03/06/2014-10
639	Utrecht	185748	2014-06-03 10:00:00	54	0101000020E6100000DA59F44E057C1440EE06D15AD1084A40	0	639-O3-03/06/2014-10

Fig. 3.6: *Figure - LML Postgres VIEW of last measured values at each station for Ozone*

- averages
- peak values
- even Voronoi-data can be derived, though that may be expensive: http://smathermather.wordpress.com/2013/12/21/voronoi-in-postgis

These VIEWs can readily applied for WMS with legenda's like here: http://www.eea.europa.eu/data-and-maps/figures/rural-concentration-map-of-the-ozone-indicator-aot40-for-crops-year-3

GeoServer Service Creation

From the above database VIEWs the WMS/WFS Layers can be created in the GeoServer Admin GUI. The main items to develop specifically are the Styled Layer Descriptors (SLDs). The choice is to provide colouring schemes for ranges of values or just labels showing the values. The latter has been chosen initially.

The challenge is to do something interesting with the Time aspect (field `sample_time`) as all features would be positions at the same (station) point coordinates.

For WMS we can use WMS-Time, for WFS we may provide search forms with queries. Other visualizations may be interesting like Voronoi diagrams: http://smathermather.wordpress.com/2013/12/21/voronoi-in-postgis/.

All OWS services are available from this URL: http://sensors.geonovum.nl/gs/sensors/ows

The WFS capabilities: http://sensors.geonovum.nl/gs/wfs?request=GetCapabilities

The WMS Capabilities: http://sensors.geonovum.nl/gs/wms?request=GetCapabilities

Within the GeoServer Admin GUI, a PostGIS datastore is added, with the schema `rivm-lml`. From there on various wizards and forms will lead to Layer creation and configuration.

3.2.2 WMS TIME Dimension

The OGC has defined additional WMS 1.3 practises for working with Dimensions, and in particular with Time and Elevation: https://portal.opengeospatial.org/files/?artifact_id=56394

GeoServer supports these extensions via a 'Dimensions' tab in each Layer configuration.

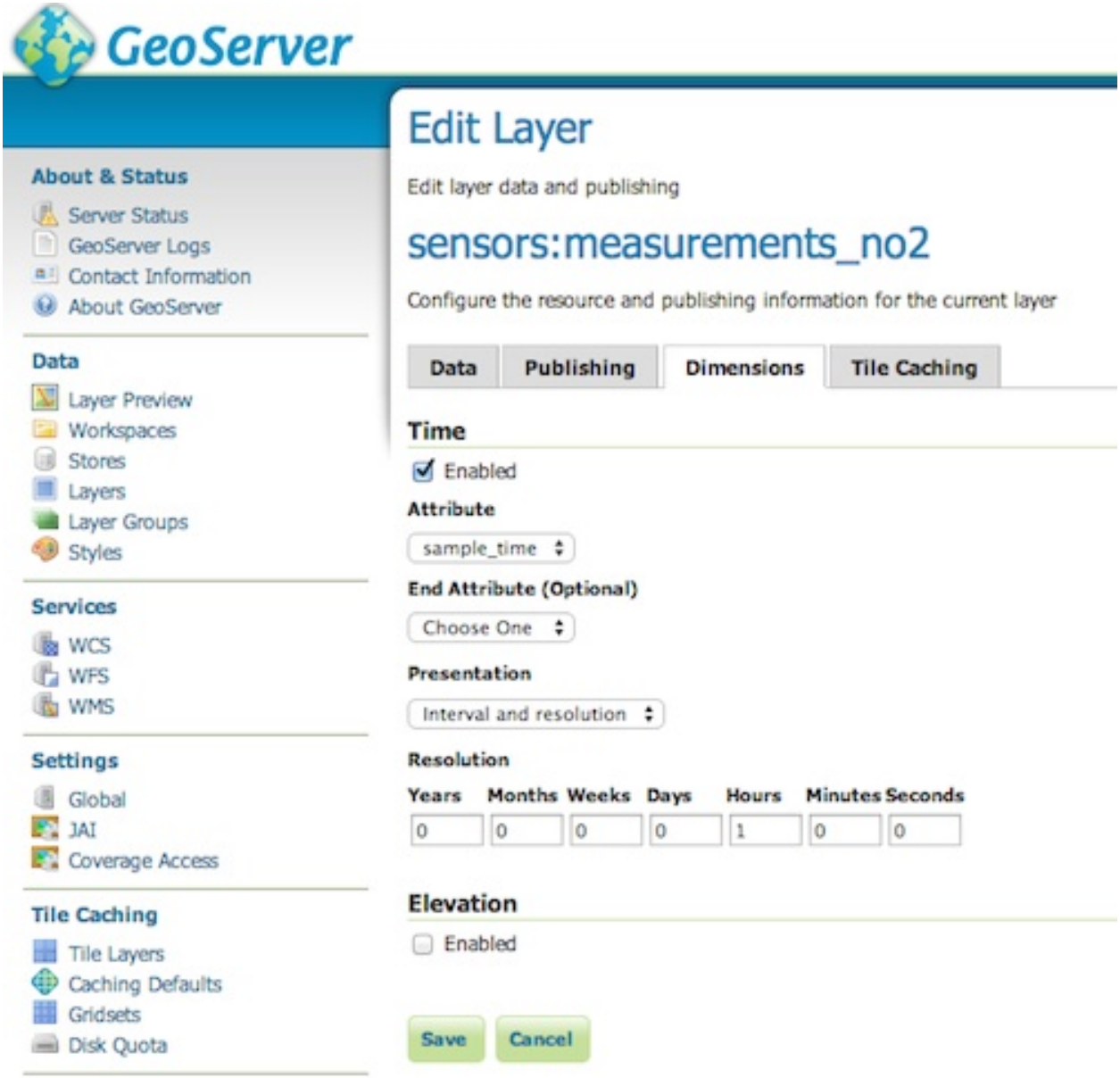


Fig. 3.7: *Figure - WMS Configuration for Time Dimension*

The `sample_time` date-field is selected, the data is presented as 'Interval and resolution' with a resolution of 1 hour as this is the standard interval for LML measurements.

The Capabilities also show for the `<component>_measurements` Layers the Time dimension. See figure below.

```
▼<Layer queryable="1" opaque="0">
   <Name>sensors:measurements_no2</Name>
   <Title>measurements_no2</Title>
   <Abstract/>
  ▼<KeywordList>
     <Keyword>v_measurements_no2</Keyword>
     <Keyword>features</Keyword>
   </KeywordList>
   <CRS>EPSG:4326</CRS>
   <CRS>CRS:84</CRS>
  ▼<EX_GeographicBoundingBox>
     <westBoundLongitude>3.74945831</westBoundLongitude>
     <eastBoundLongitude>6.9333334</eastBoundLongitude>
     <southBoundLatitude>50.88805771</southBoundLatitude>
     <northBoundLatitude>53.33166504</northBoundLatitude>
   </EX_GeographicBoundingBox>
   <BoundingBox CRS="CRS:84" minx="3.74945831" miny="50.88805771" maxx="6.9333334" maxy="53.33166504"/>
   <BoundingBox CRS="EPSG:4326" minx="50.88805771" miny="3.74945831" maxx="53.33166504" maxy="6.9333334"/>
   <BoundingBox CRS="EPSG:25831" minx="549909.7434493984" miny="5637644.127521528" maxx="776619.7438439372" r
   <BoundingBox CRS="EPSG:25832" minx="130791.40364189784" miny="5639411.1180428155" maxx="362378.946956787
   <BoundingBox CRS="EPSG:3034" minx="2684856.4576507737" miny="3575797.904242327" maxx="2960183.096049659" r
   <BoundingBox CRS="EPSG:3035" minx="3090811.014900814" miny="3881777.003332834" maxx="3376111.112180281" ma
   <BoundingBox CRS="EPSG:28992" minx="39760.768195112905" miny="322030.9177588781" maxx="263794.0422425462"
   <BoundingBox CRS="EPSG:900913" minx="417387.78981980804" miny="6601516.264053305" maxx="771815.1435879963
   <BoundingBox CRS="EPSG:3857" minx="417387.78981980804" miny="6601516.264053305" maxx="771815.1435879963" r
   <BoundingBox CRS="EPSG:4258" minx="50.88805775024979" miny="3.7494583099999996" maxx="53.33166507777328" r
  ▼<Dimension name="time" default="current" units="ISO8601">
     2014-05-01T18:00:00.000Z/2014-06-15T16:00:00.000Z/PT1H
   </Dimension>
  ▼<Style>
     <Name>last_measurements_no2</Name>
     <Title>Last RIVM measurements_no2</Title>
     <Abstract>Last RIVM measurements_no2 style</Abstract>
    ▼<LegendURL width="20" height="20">
       <Format>image/png</Format>
       <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple" xlink:href="http://sens
       service=WMS&request=GetLegendGraphic&format=image%2Fpng&width=20&height=20&layer=measurements_no2"/>
     </LegendURL>
   </Style>
 </Layer>
```

Fig. 3.8: *Figure - WMS Capabilities Layer with Time Dimension*

3.2.3 Stations Layer

This is quite trivial: a small flat table with station info and a Point geometry. The WMS layer may be added to a viewer like the KadViewer: http://kadviewer.kademo.nl or any other GIS package.

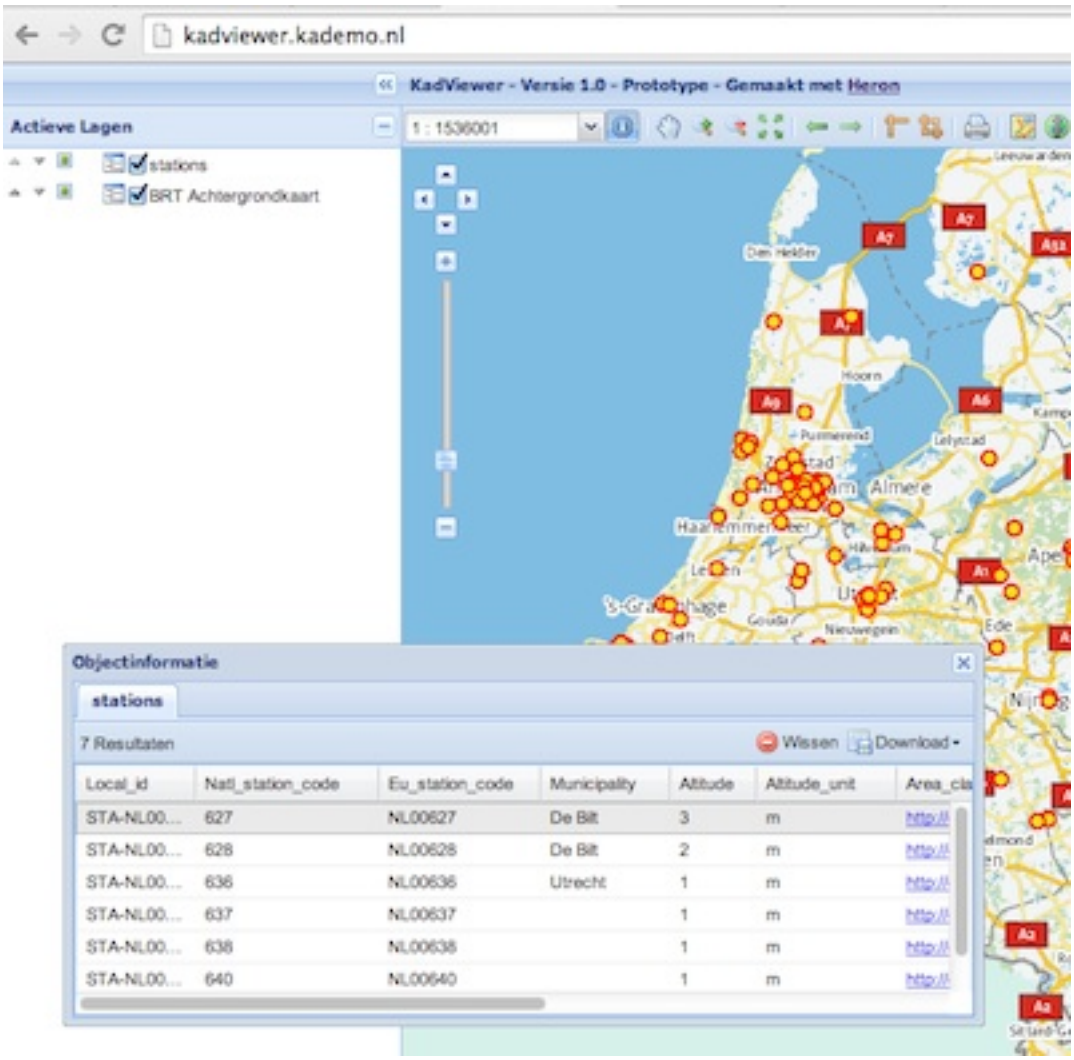


Fig. 3.9: *Figure - GeoServer stations WMS Layer with FeatureInfo*

3.2.4 Measurements Layers

These layers are created from the individual per-component VIEWs described above. For each component there are `last_measurements_<component>` and a `measurements_<component>` Layers.

3.2.5 Styled Layer Descriptors

A simplified SLD approach is taken, just circles with a text label for values. More advanced SLDs may be added later. Here is an example for NO2, both used for the `measurements_no2` and `last_measurements_no2` layers.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<StyledLayerDescriptor version="1.0.0"
                       xsi:schemaLocation="http://www.opengis.net/sld␣
↪StyledLayerDescriptor.xsd"
                       xmlns="http://www.opengis.net/sld"
                       xmlns:ogc="http://www.opengis.net/ogc"
                       xmlns:xlink="http://www.w3.org/1999/xlink"
                       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <!-- a Named Layer is the basic building block of an SLD document -->
    <NamedLayer>
        <Name>last_measurements_no2</Name>
        <UserStyle>
            <!-- Styles can have names, titles and abstracts -->
            <Title>Last RIVM measurements_no2</Title>
            <Abstract>Last RIVM measurements_no2 style</Abstract>
            <IsDefault>1</IsDefault>
            <!-- FeatureTypeStyles describe how to render different features -->
            <!-- A FeatureTypeStyle for rendering points -->
            <FeatureTypeStyle>
                <Rule>
                    <PointSymbolizer>
                        <Graphic>
                            <Mark>
                                <WellKnownName>circle</WellKnownName>
                                <Fill>
                                    <CssParameter name="fill">#8b008b</CssParameter>
                                    <CssParameter name="fill-opacity">1.0</
↪CssParameter>
                                </Fill>
                                <Stroke>
                                    <CssParameter name="stroke">#ee82ee</CssParameter>
                                    <CssParameter name="stroke-width">1</CssParameter>
                                </Stroke>
                            </Mark>
                            <Size>30</Size>
                        </Graphic>
                    </PointSymbolizer>

                    <TextSymbolizer>
                        <Label>
                            <ogc:PropertyName>sample_value</ogc:PropertyName>
                        </Label>
                        <Font>
                            <CssParameter name="font-family">
                                <ogc:Literal>Lucida Sans Regular</ogc:Literal>
                            </CssParameter>

                            <CssParameter name="font-size">
                                <ogc:Literal>10</ogc:Literal>
                            </CssParameter>
                            <CssParameter name="font-weight">
                                <ogc:Literal>bold</ogc:Literal>
                            </CssParameter>
                        </Font>
                        <LabelPlacement>
                            <PointPlacement>
                                <AnchorPoint>
```

```
                                    <AnchorPointX>0.5</AnchorPointX>
                                    <AnchorPointY>0.5</AnchorPointY>
                                </AnchorPoint>
                            </PointPlacement>
                        </LabelPlacement>
                        <Fill>
                            <CssParameter name="fill">#ffffff</CssParameter>
                        </Fill>
                    </TextSymbolizer>
                </Rule>
            </FeatureTypeStyle>
        </UserStyle>
    </NamedLayer>
</StyledLayerDescriptor>
```

These layers can be rendered in any WMS viewer, but in particular viewers that support the WMS-Time parameter on the client, for example the HeronViewer developed for this project. Heron is a web mapping client framework that builds on OpenLayers . OL supports WMS Dimensions in general.

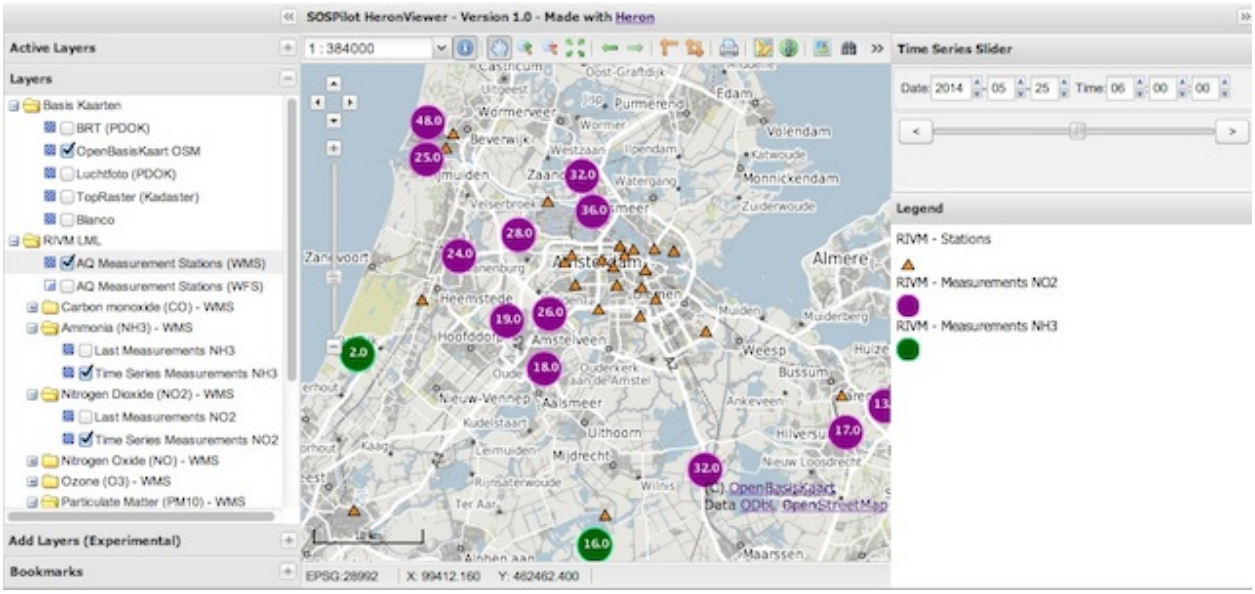


Fig. 3.10: *Figure - Heron Viewer showing WMS-T Measurements Layers*

3.2.6 WFS Layers

WFS Layers are created automatically for each WMS Layer in GeoServer. A WFS query can be as follows:

3.3 SOS Services

"The OGC Sensor Observation Service aggregates readings from live, in-situ and remote sensors. The service provides an interface to make sensors and sensor data archives accessible via an interoperable web based interface."

The chapter on server administration describes how the 52 North SOS server is deployed. This is called here the 'SOS Server'.

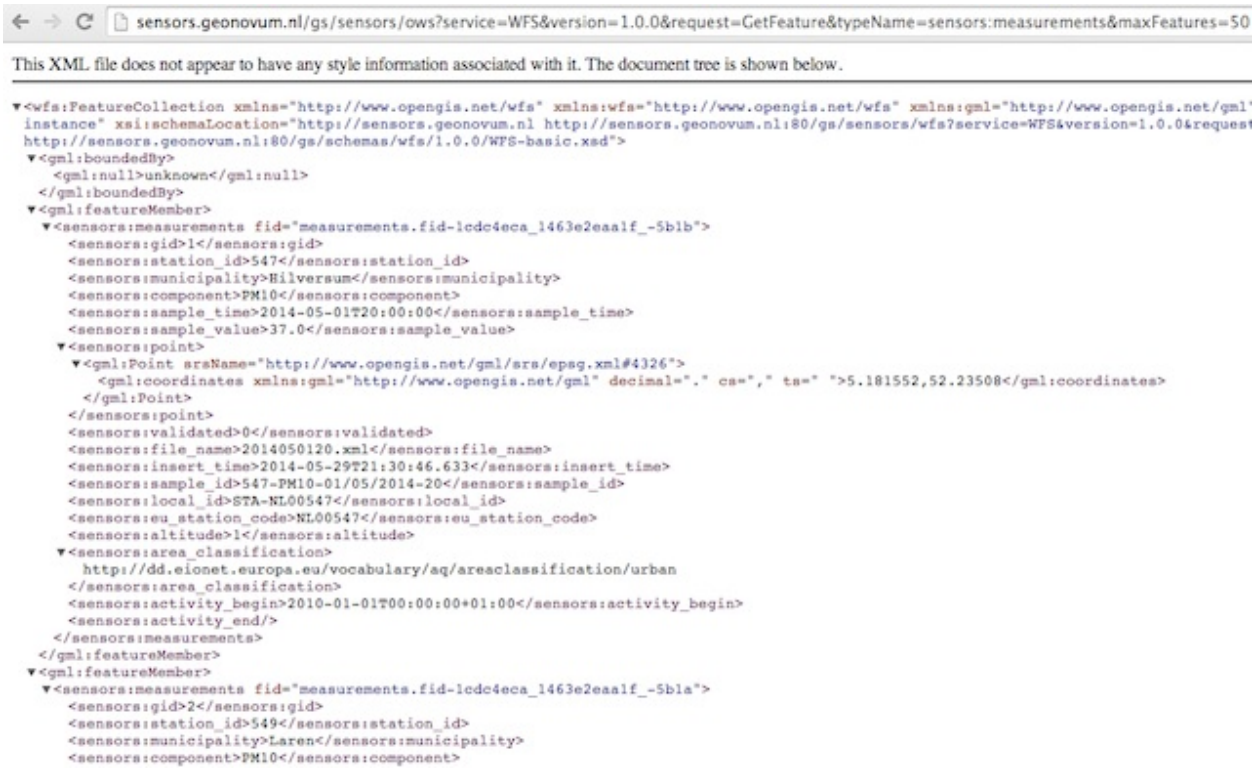


Fig. 3.11: *Figure - GeoServer measurements Layer WFS query*

Different as from GeoServer the 'SOS Server' comes with its own database schema and store. Via ETL Step 3 as described in the data chapter, the tables are populated by publishing both sensor and observation data via SOS-Transactions to the SOS Server. There is no need to, like with GeoServer, to configure SOS services as these are readily available.

A later option is to directly couple the Core AQ tables to the SOS server via a mapping config.

The SOS server is available at this endpoint: http://sensors.geonovum.nl/sos Note that this server also supports the REST API as specified in: http://sensorweb.demo.52north.org/sensorwebclient-webapp-stable/api-doc/index.html. The REST API is at: http://sensors.geonovum.nl/sos/api/v1/

On June 13, 2014, a first test was performed integrating Air Quality SOSs from Belgium, Holland and Sweden via the 52 North HTML5 client. This gave a promising result as depicted below.

Fig. 3.12: *Figure - GeoServer measurements Layer WFS query*

From: Simon Jirka ..... during the last days, the SOS servers form the Netherlands and also from Sweden went online. We have used this opportunity to connect our client to all three servers and to load Ozone data. Attached you find a screenshot of the result. The data from Sweden is coloured in green, the time series from Belgium is blue and the data from the Netherlands has red colour. Data from the SOS operated by the EEA should follow next week, as well.

CHAPTER 4

Clients

This chapter describes how various client applications are realized on top of the web services In particular:

- WFS and WMS-Time clients
- OWS SOS clients

4.1 HeronViewer

Figure 1 depicts a screenshot of the HeronViewer, found at http://sensors.geonovum.nl/heronviewer .

The HeronViewer is built using the Heron Web Mapping client framework. This is mainly a matter of providing a configuration of widgets and layer settings.

The main feature of this viewer is that it interacts with the various WMS Layers (see Web Services), using the OGC standardized WMS Time Dimension, also know as WMS-T.

This viewer uses all standard Heron components, except a for a `TimeRangePanel`, the slider that enables the user to go through time. Via WMS-T component measurement values are displayed for that paricular date and time.

4.2 SOS Clients

There are several possible SOS clients.

4.2.1 JS-SensorWeb-Client

This client is known as the "pure HTML5 Client". The GitHub is at https://github.com/52North/js-sensorweb-client and a nice 52North demo at http://sensorweb.demo.52north.org/jsClient-0.2.0.

This app uses the REST interface of the 52North SOS server, so no extra .war is required. The REST API is at: http://sensors.geonovum.nl/sos/api/v1/

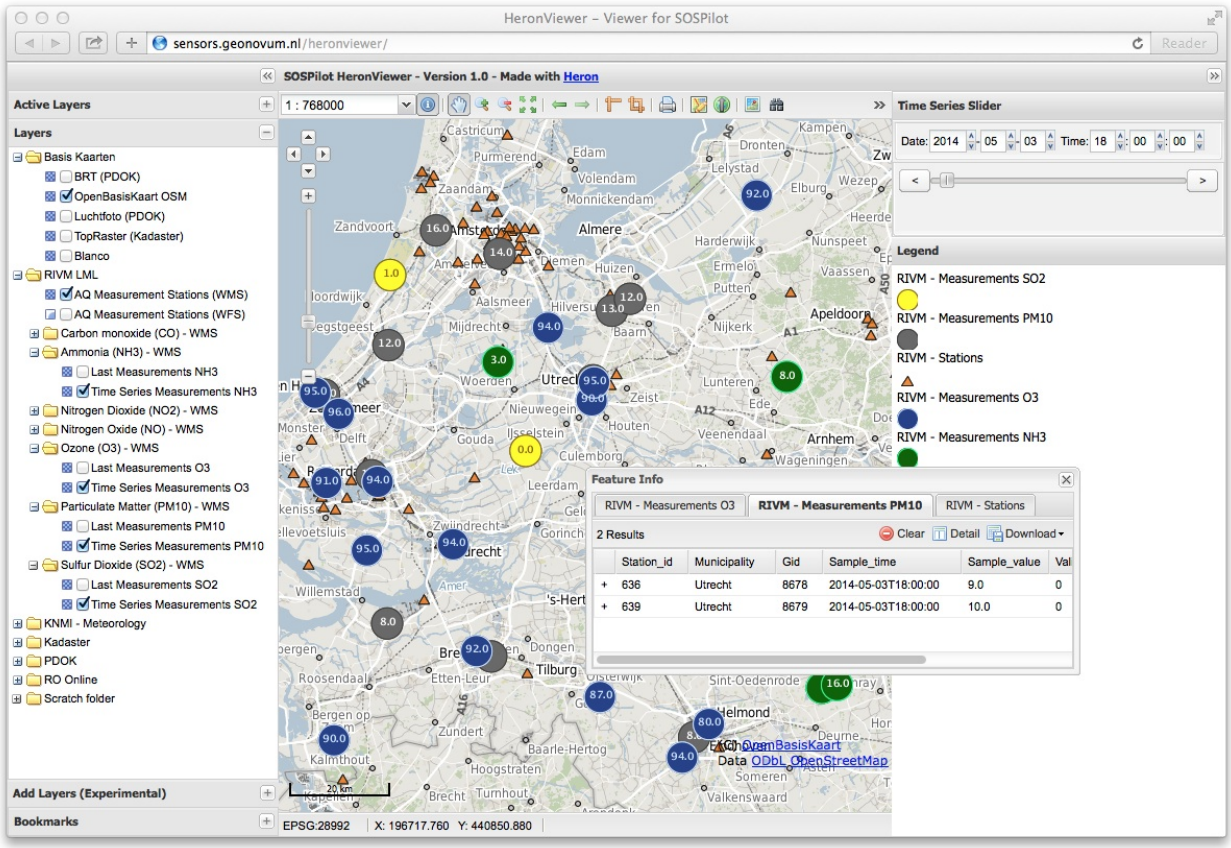


Fig. 4.1: *Figure 1 - HeronViewer showing components with time series values*

We install this client as a pure HTML5/JavaScript app (no .war) at: at URL http://sensors.geonovum.nl/jsclient

Installation from GitHub and local configuration as follows

```
$ cd /opt/52north/js-sensorweb-client
$  git clone https://github.com/52North/js-sensorweb-client git
$ cd git
# Configure Geonovum/RIVM SOS REST service instances within ./WebContent/js/models/
↪settings.js
  .
  .
var Settings = {

    // The entries in this list will be removed from the provider list offered to the␣
↪user
    providerBlackList : [
    {
        serviceID : 'srv_6d9ccea8d609ecb74d4a512922bb7cee', // ircel
        apiUrl : 'http://sensorweb.demo.52north.org/sensorwebclient-webapp-stable/api/
↪v1/'
    },
    {
        serviceID : 'srv_7cabc8c30a85fab035c95882df6db343', // BfG sos
        apiUrl : 'http://sensorweb.demo.52north.org/sensorwebclient-webapp-stable/api/
↪v1/'
    }
    ],

    // A list of timeseries-API urls and an appropriate identifier to create internal␣
↪timeseries ids
    restApiUrls : {
//          'http://192.168.1.135:8080/sensorwebclient-webapp/api/v1/' : 'localhost'
//          'http://localhost:8090/sensorwebclient-webapp-3.3.0-SNAPSHOT/api/v1/' :
↪'localhost'
//          'http://sensorweb.demo.52north.org/sensorwebclient-webapp-stable/api/v1/'␣
↪: '52nSensorweb',
        'http://sosrest.irceline.be/api/v1/' : 'irceline',
                'http://sensors.geonovum.nl/sos/api/v1/' : 'rivm'
//          'http://www.fluggs.de/sos2/api/v1/' : 'fluggs'
    },

    // default selected provider
//  defaultProvider : {
//          serviceID : 'srv_738111ed219f738cfc85be0c8d87843c',
//          apiUrl : 'http://sensorweb.demo.52north.org/sensorwebclient-webapp-stable/
↪api/v1/'
//  },

    defaultProvider : {
        serviceID : 'srv_738111ed219f738cfc85be0c8d87843a',
        apiUrl : 'http://sensors.geonovum.nl/sos/api/v1/'
    },

    // zoom level in the map, used for user location and station position
    zoom : 13,
  .
  .
# Build with Maven
$ mvn clean install
```

```
# jsclient is at /opt/52north/js-sensorweb-client/git/target/jsClient-0.2.0
# deploy in Apache
$ cp -r target/jsClient-0.2.0 /var/www/sensors.geonovum.nl/site/jsclient
```

Below is the result.

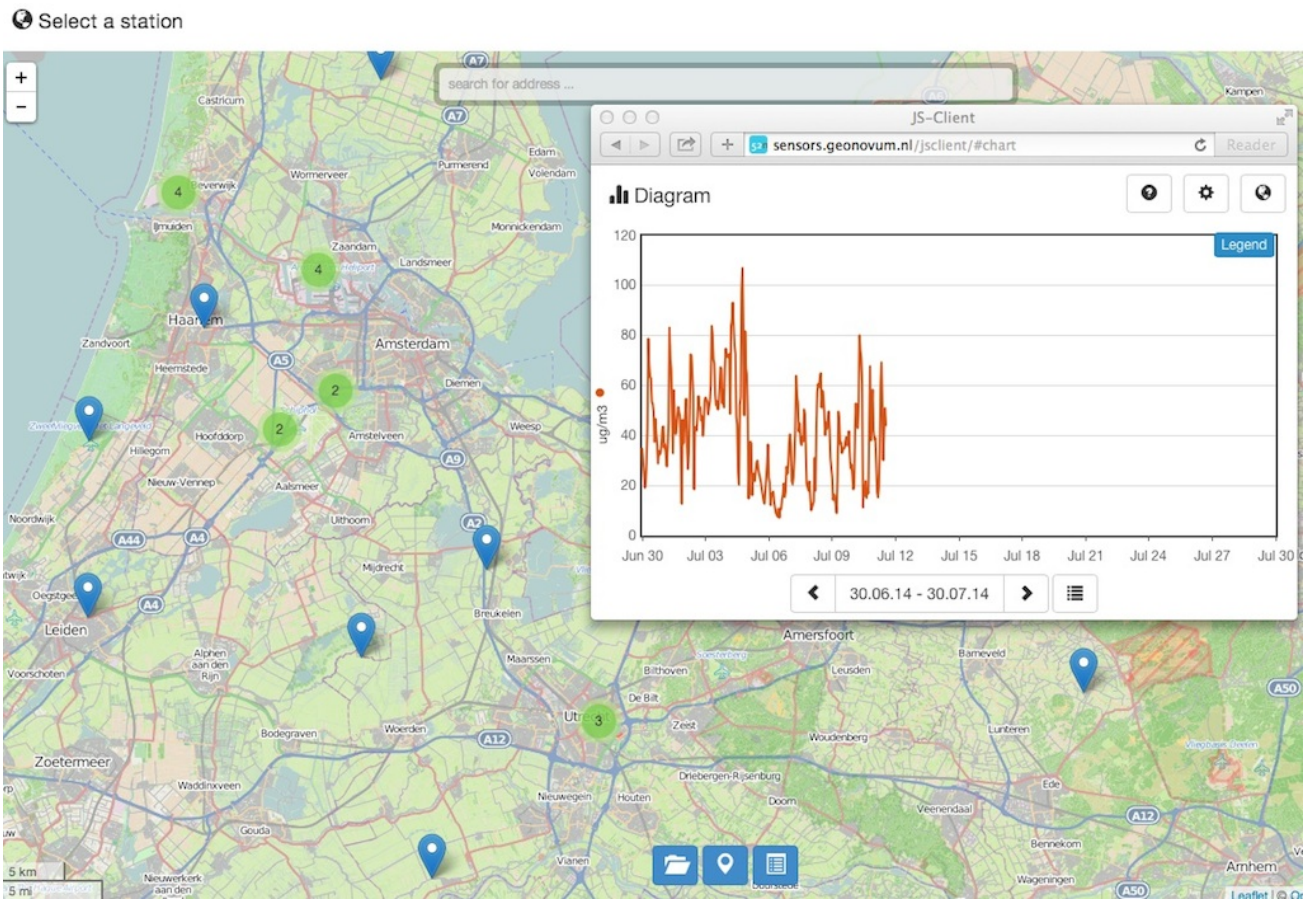


Fig. 4.2: *Figure 2 - HTML5 App JSClient at http://sensors.geonovum.nl/jsclient*

4.2.2 SensorWebClient (52N)

Homepage: https://wiki.52north.org/bin/view/SensorWeb/SensorWebClient Installation Guide: https://wiki.52north.org/bin/view/SensorWeb/SensorWebClientInstallationGuide

Install Development line from GitHub

```
$ cd /opt/52north/sensorwebclient
$ git clone https://github.com/ridoo/SensorWebClient.git git
$ cd git
# Configure SOS instances within ./sensorwebclient-webapp/src/main/webapp/ds/sos-
↪instances.data.xml
# Copy ${project_root}/sensorwebclient-build-example.properties to ~/sensorwebclient-
↪build-dev.properties
# Adapt: ~/sensorwebclient-build-dev.properties
$ cd sensorwebclient-webapp
```

```
$ mvn -e clean install -P env-dev
# target war: sensorwebclient-webapp/target/sensorwebclient-webapp-3.3.0-SNAPSHOT.war

# Deploy in Tomcat
# Als root:
$ chown tomcat7:tomcat7
    /opt/52north/sensorwebclient/git/sensorwebclient-webapp/target/sensorwebclient-
↪webapp-3.3.0-SNAPSHOT.war
$ cp /opt/52north/sensorwebclient/git/sensorwebclient-webapp/target/sensorwebclient-
↪webapp-3.3.0-SNAPSHOT.war
   /var/www/sensors.geonovum.nl/webapps/swc.war
```

The client runs at http://sensors.geonovum.nl:8080/swc but shows no stations for The Netherlands. It does for Belgium. See also this issue: https://github.com/Geonovum/sospilot/issues/11

4.2.3 SOS-JS Client (52N)

This is a "pure" JavaScript client. It builds on OpenLayers and JQuery.

Homepage: https://github.com/52North/sos-js

SOS.js is a Javascript library to browse, visualise, and access, data from an Open Geospatial Consortium (OGC) Sensor Observation Service (SOS)....This module is built on top of OpenLayers, for low-level SOS request/response handling.

This client has components for protocol handling (via OpenLayers), maps and visualization with plotted graphs and tabular data. There are some examples available.

A live application built by the British Antarctic Survey and can be viewed here: http://basmet.nerc-bas.ac.uk/sos. There is also an advanced viewer: http://add.antarctica.ac.uk/home/add6

We will build a web-app based on the above. This app can be found at: http://sensors.geonovum.nl/sos-js-app We cannot yet select stations by clcking inthe map, but via the offering list we can plot a graph for a chemical component for a station during a timeframe.

One can also select multiple stations for a pollutant and select date intervals int he time segment. See Figure 7 below.

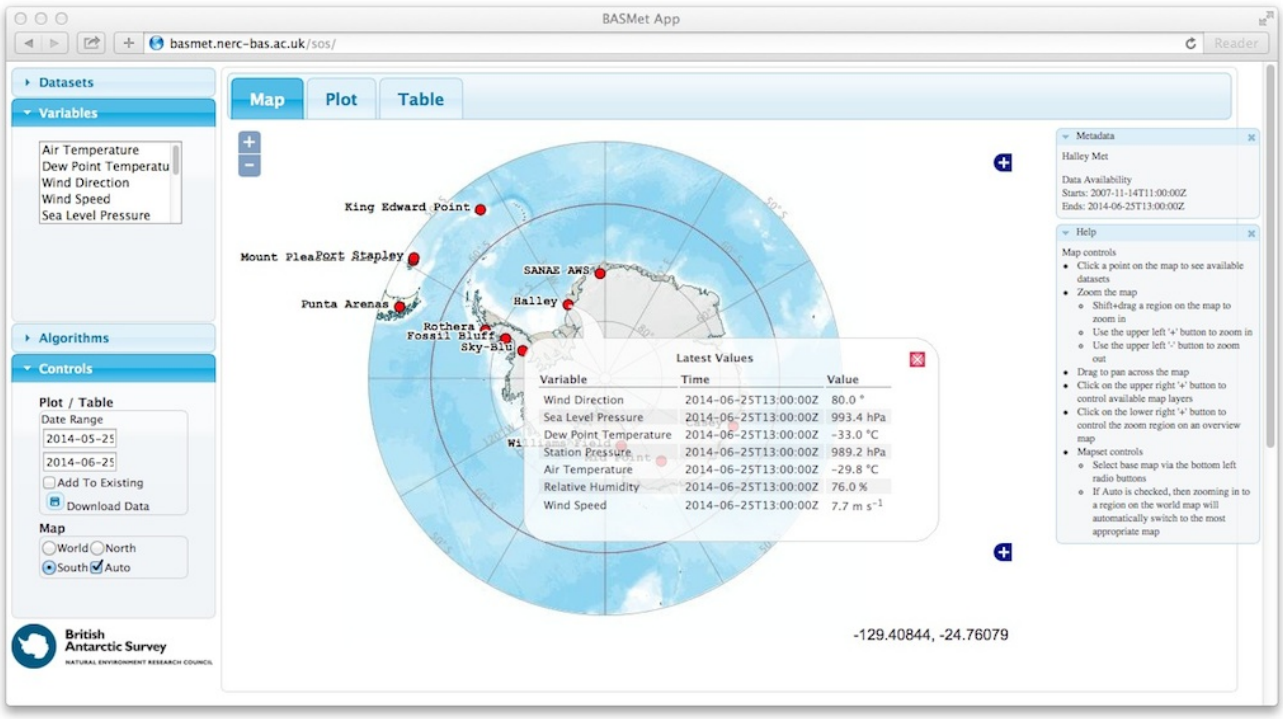


Fig. 4.3: *Figure 3 - app made with SOS-JS by British Antarctic Survey (Map)*

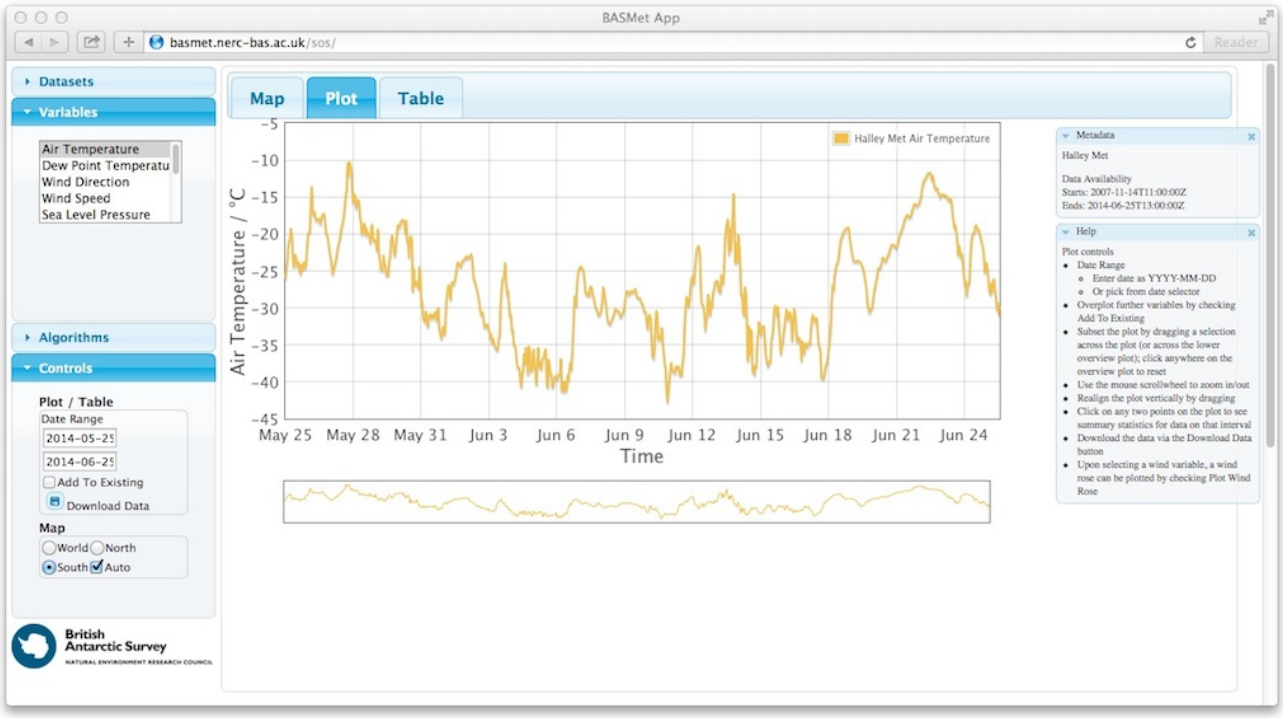


Fig. 4.4: *Figure 4 - app made with SOS-JS by British Antarctic Survey (Plot)*

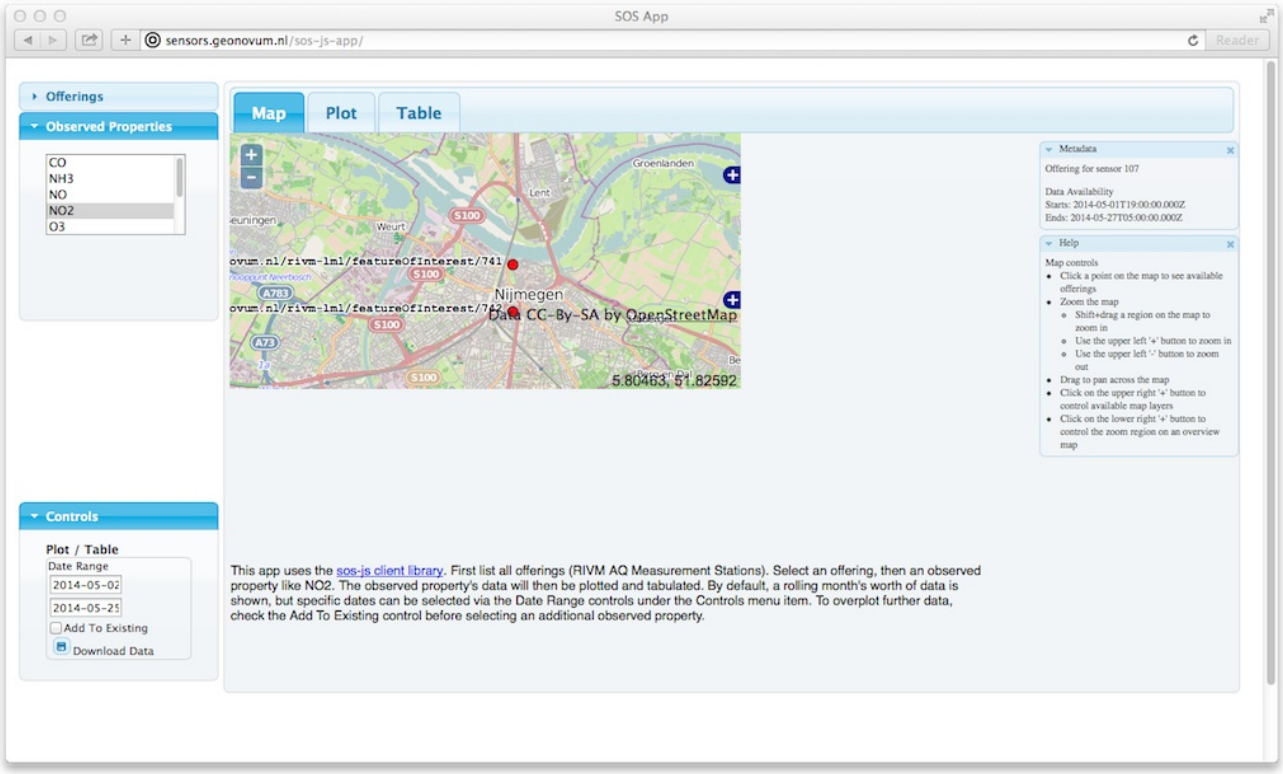


Fig. 4.5: *Figure 5 - app made with SOS-JS for SOSPilot (Map)*

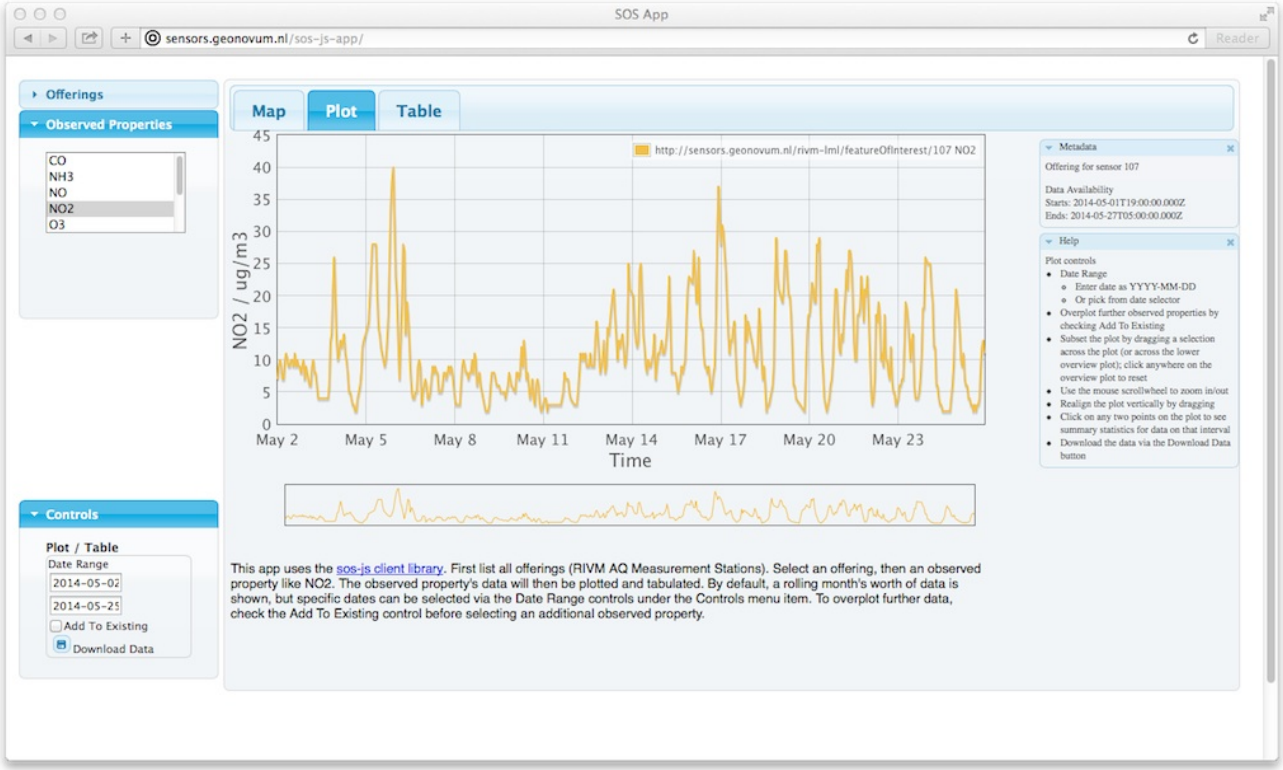


Fig. 4.6: *Figure 6 - app made with SOS-JS for SOSPilot shows NO2 graph for Station Roerdalen,NL00107*

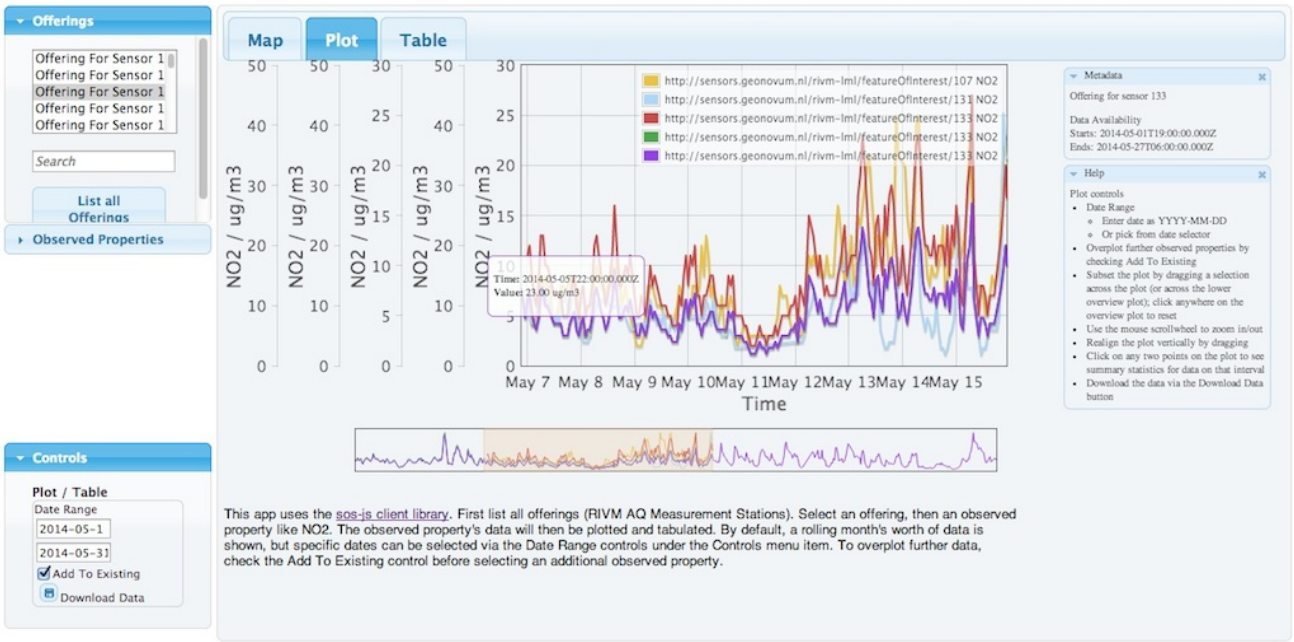


Fig. 4.7: *Figure 7 - app made with SOS-JS for SOSPilot shows NO2 graph for Multiple Stations*

CHAPTER 5

Administration

This chapter describes the operation and maintenance aspects for the SOSPilot platform. For example:

- how to start stop servers
- managing the ETL
- where to find logfiles
- setting Postgres credentials

5.1 Data Management

The data is harvested and transformed in three ETL steps. See chapter on Data Management.

The `cron` job runs under account `sadmin` with this cronfile: https://github.com/Geonovum/sospilot/blob/master/src/rivm-lml/cronfile.txt . LOgfiles are under `/var/log/sospilot`.

Every 30 minutes all three steps are run (for now) via the script `https://github.com/Geonovum/sospilot/blob/master/src/rivm-lml/etl-all.sh`. It is best to stop the cronjob whenever doing any of the resets below. Also be aware that the ETL steps build up history.

Postgres credentials: set these once in the file `https://github.com/Geonovum/sospilot/blob/master/src/rivm-lml/pgcreds.sh` for your local Postgres. All scripts will use these values.

5.1.1 Reset ETL Step 1 - RIVM File Harvester

- Empty (not drop) the table `rivm_lml.lml_files`

The Harvester tracks its progress via the unique file id in `rivm_lml.lml_files`.

5.1.2 Reset ETL Step 1 - Stations

Whenever there is a new set of stations CSV. This needs to be done. Note: also the SOS Sensor data (see below) needs to be updated. This may be problematic/refined. See https://github.com/Geonovum/sospilot/tree/master/data/rivm-lml/stations

- Drop the table `rivm_lml.stations`
- check with * `stations2gml.sh`
- `stations2postgis.sh`

5.1.3 Reset ETL Step 2 - Files to Core AQ

- Reset counter `last_id` to `-1` in table `rivm_lml.etl_progress` for row where worker is `files2measurements`
- Also you will need to empty (not drop) the table `rivm_lml.measurements`

5.1.4 Reset ETL Step 1 + Step 2 - Shortcut

Step 1 + Step 2 can be reset via a single script:

`https://github.com/Geonovum/sospilot/blob/master/src/rivm-lml/reset-rivm-etl.sh`

5.1.5 Reset ETL Step 3 - SOS-T Sensor Publishing

This re-publishes the Stations as Sensors.

- You will need to clear the SOS database (for now, see below)
- run https://github.com/Geonovum/sospilot/blob/master/src/rivm-lml/stations2sensors.sh

5.1.6 Reset ETL Step 3 - SOS-T Observations Publishing

- Reset counter `last_id` to `-1` in table `rivm_lml.etl_progress` for row where worker is `measurements2sos`
- Also you will need to clear the SOS database

5.1.7 Clean SOS data

See data and scripts at https://github.com/Geonovum/sospilot/tree/master/data/sosdb. Using this procedure, no reinstall of the .war file is requried or any other Admin reset (somehow an Admin reset did not work).

As root do

- Stop Tomcat (command `/opt/bin/tcdown`)
- Clean DB: reinstall PG schema using: https://github.com/Geonovum/sospilot/blob/master/data/sosdb/empty-dm-dump.sql
- Remove `/var/www/sensors.geonovum.nl/webapps/sos/cache.tmp`.
- Start Tomcat (command `/opt/bin/tcup`)

- Republish the Sensors (see Stations to Sensors)
- restart the cron (see above)

5.1.8 Reset ETL Step 3 - Shortcut

Step 3 can be reset via a single script:

`https://github.com/Geonovum/sospilot/blob/master/src/rivm-lml/reset-sos-etl.sh`

This renders a clean/empty SOS. The Sensors (see above) need to be republished.

5.2 Web Services

The Tomcat server runs both GeoServer and the 52N SOS server. Logfiles in `/var/log/tomcat/catalina.out`. Stop/start with shorthand: `/opt/bin/tcstop` and `/opt/bin/tcstart`.

Admin GeoServer: http://sensors.geonovum.nl/gs/web

Admin SOS: http://sensors.geonovum.nl/sos

5.2.1 SOS Server

The original version received was `52N-SOS-INSPIRE-with-RestAPI_20140519.zip`. This version has been patched since to solve some issues. See `server-inrichting` chapter.

Patching basically means: Stop Tomcat, copy a replacement .jar to `/var/www/sensors.geonovum.nl/webapps/sos/WEB-INF/lib` and start Tomcat.

CHAPTER 6

Smart Emission Project

NB The Smart Emission Project now (May 2016) has its own resources: website, GitHub etc. The info below is left here for historic reasons and not accurate anymore!! Please go to: www.smartemission.nl **for all links to docs, GitHub etc**

In september 2015 the SOSPilot platform was extended to handle air quality data from the Smart Emission (Nijmegen) project. The same ETL components and services (WMS, WFS, SOS) as used for RIVM AQ data were reused with some small modifications. Source code for the Smart Emission ETL can be found in GitHub: https://github.com/Geonovum/sospilot/tree/master/src/smartem . Small sample data used for development can be found at https://github.com/Geonovum/sospilot/tree/master/data/smartem

6.1 Background

Read about the Smart Emission project via: Smart Emission (Nijmegen) project. The figures below were taken from the Living Lab presentation, on June 24, 2015: http://www.ru.nl/publish/pages/774337/smartemission_ru_24juni_lc_v5_smallsize.pdf

In the paper Filling the feedback gap of place-related externalities in smart cities the project is described extensively.

"...we present the set-up of the pilot experiment in project "Smart Emission", constructing an experimental citizen-sensor-network in the city of Nijmegen. This project, as part of research program 'Maps 4 Society,' is one of the currently running Smart City projects in the Netherlands. A number of social, technical and governmental innovations are put together in this project: (1) innovative sensing method: new, low-cost sensors are being designed and built in the project and tested in practice, using small sensing-modules that measure air quality indicators, amongst others NO2, CO2, ozone, temperature and noise load. (2) big data: the measured data forms a refined data-flow from sensing points at places where people live and work: thus forming a 'big picture' to build a real-time, in-depth understanding of the local distribution of urban air quality (3) empowering citizens by making visible the 'externality' of urban air quality and feeding this into a bottom-up planning process: the community in the target area get the co-decision-making control over where the sensors are placed, co-interpret the mapped feedback data, discuss and collectively explore possible options for improvement (supported by a Maptable instrument) to get a fair and 'better' distribution of air pollution in the city, balanced against other spatial qualities. ...."

The Sensor (Sensor Jose) used was developed by Intemo with Server-sensor connection by CityGIS. See below.

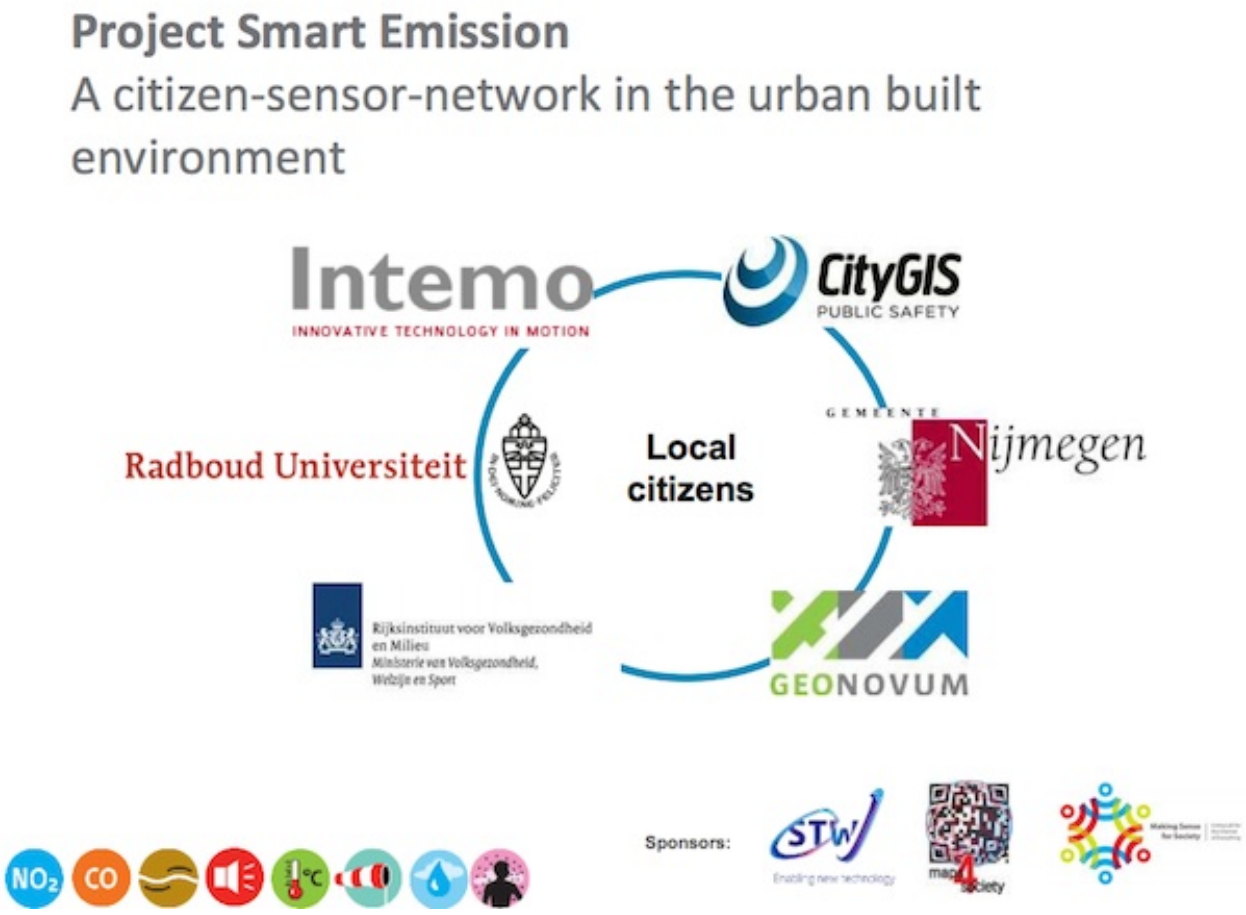


Fig. 6.1: *Smart Emission Project - Participants*

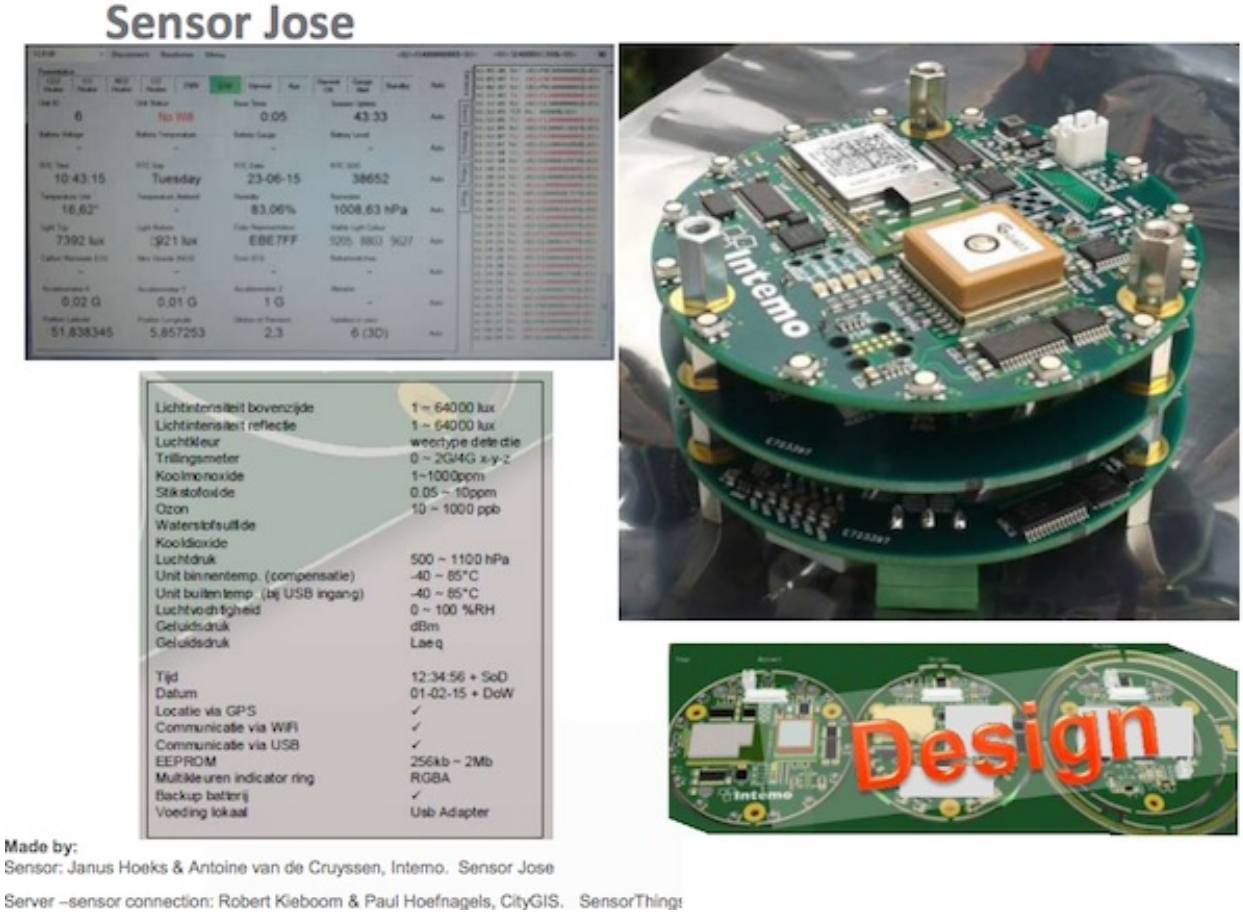


Fig. 6.2: *Smart Emission Project - Sensors*

The data from these sensors was converted and published into standard OGC services: WMS(-Time), WFS and SOS. This is described in the remainder of this chapter. For readers eager to see the results, these are presented in the next section. A snapshot of AQ data was provided through CityGIS via FTP.

6.2 Results

Results can be viewed in basically 3 ways:

- as WMS and WMS-Time Layers via the Heron Viewer: http://sensors.geonovum.nl/heronviewer
- as SOS-data via the SOS Web-Client: http://sensors.geonovum.nl/jsclient
- as raw SOS-data via SOS or easier via the SOS-REST API

Below some guidance for each viewing method.

6.2.1 Heron Viewer

Within the Heron Viewer at http://sensors.geonovum.nl/heronviewer one can view Stations and Measurements. These are available as standard WMS layers organized as follows:

- Stations Layer: geo-locations and info of all stations (triangles)
- Last Measurements Layers: per-component Layers showing the last known measurements for each station
- Measurements History Layers: per-component Layers showing measurements over time for each station

To discern among RIVM LML and Smart Emission data, the latter Stations are rendered as pink-purple icons. The related measurements data (circles) have a pink-purple border.

Stations Layer

The figure below shows all stations from RIVM LML (orange active stations, grey inactive) and Smart Emission (pink-purple triangles).

Clicking on a Station provides more detailed info via WMS `GetFeatureInfo` in a pop-up window.

Last Measurements Layers

In the viewer the latest measurements per station can be shown. NB the Smart Emission data may not be current. LML data is current, i.e. from the current last hour.

Use the map/folder on the left called “Chemische Componenten (Current)” to open a chemical component. For each component there are one or two (NO2, CO and O3) layers that can be enabled. Click on a circle to see more detail.

Measurements History Layers

This shows measurements through time (using WMS-Time). NB Smart Emission data is mainly from july/august 2015!

Use the map/folder on the left called “Chemische Componenten (Historie)”. For each component there are one or two (NO2, CO and O3)layers that can be enabled. Click on a circle to see more detail. Use the TimeSlider on the upper right to go through date and time. The image above shows O3 at July 27, 2015, 10:00. Pink-purple-bordered circles denote Smart Emission measurements.

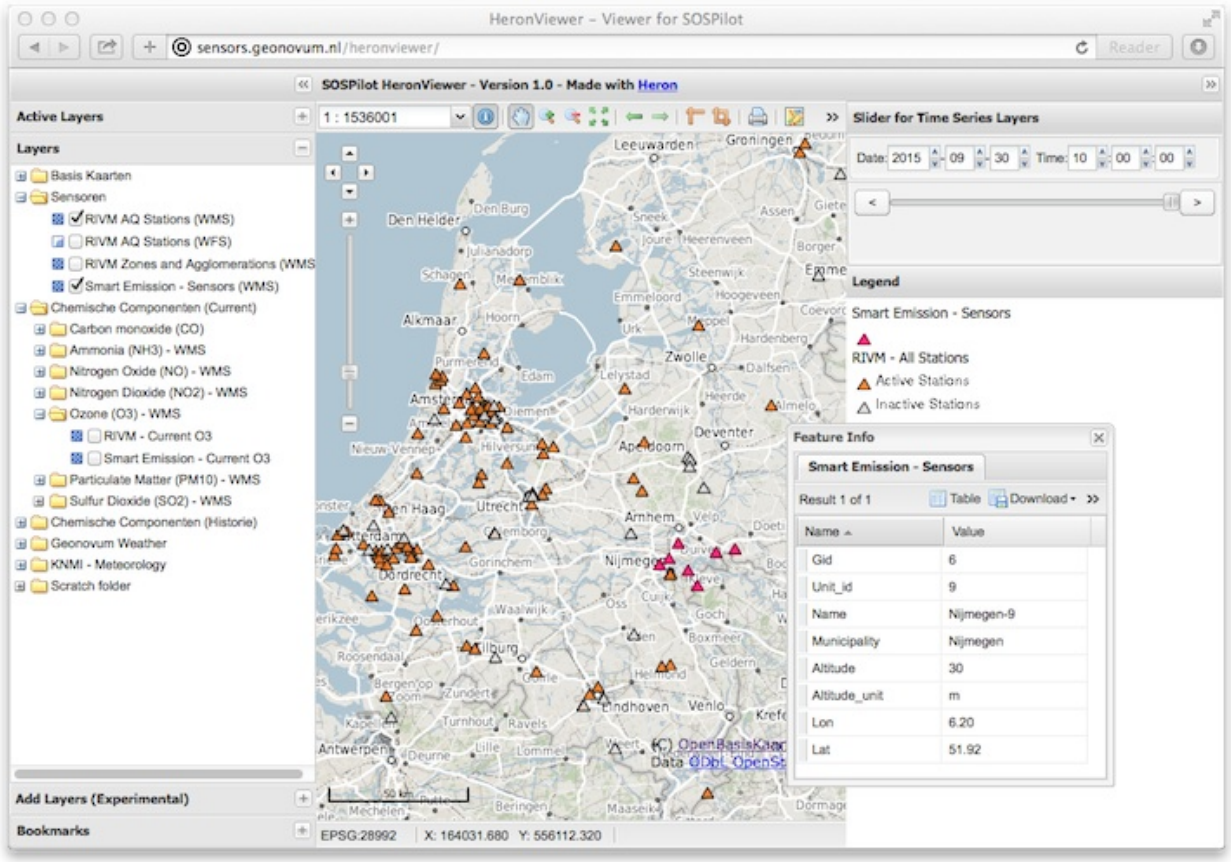


Fig. 6.3: *Smart Emission Stations in Heron Viewer*

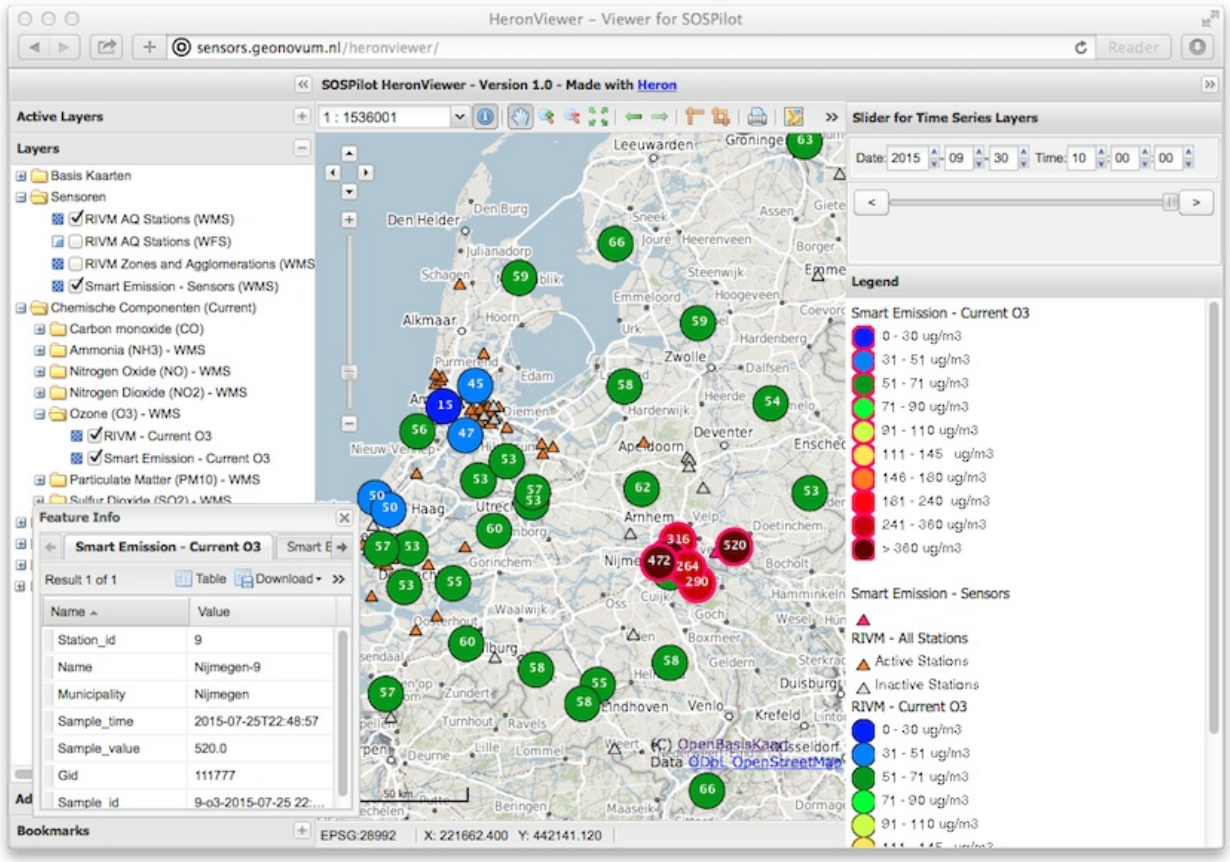


Fig. 6.4: *Heron Viewer showing latest known O3 Measurements*

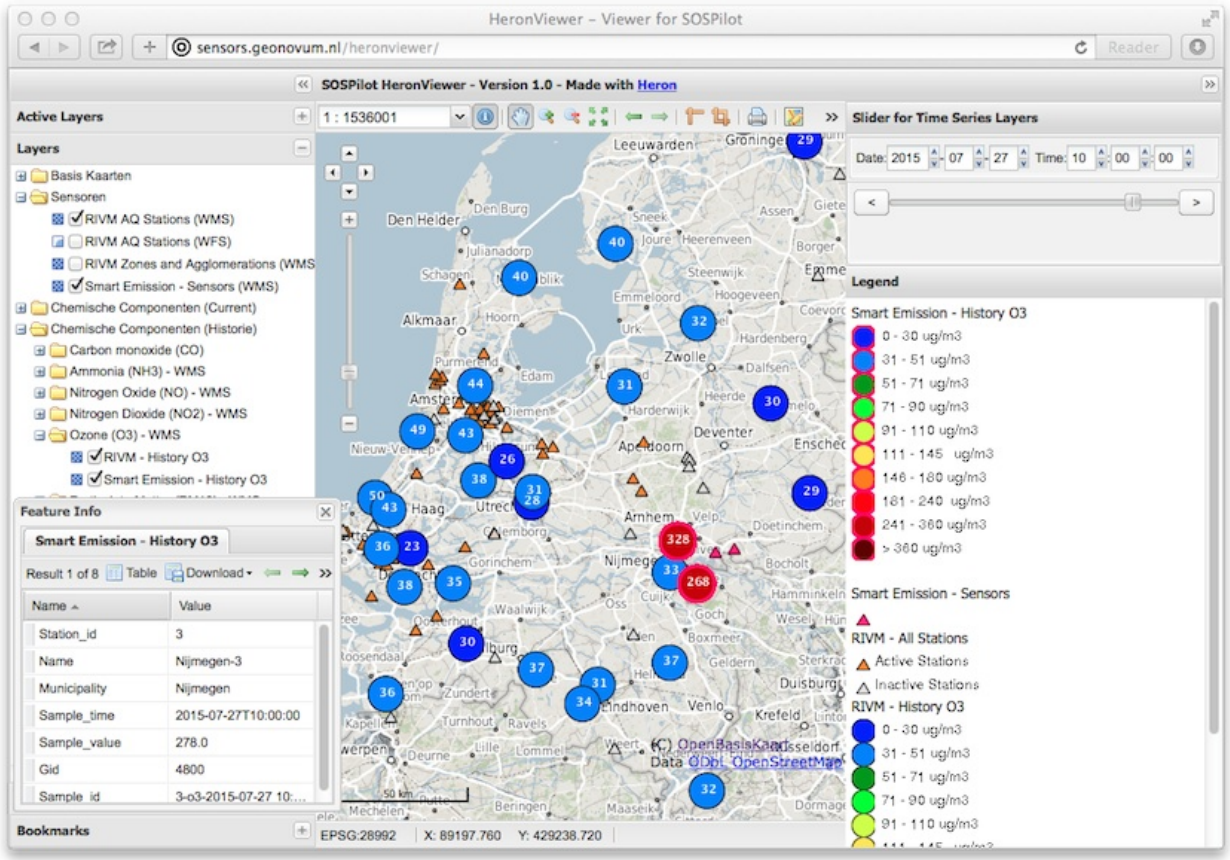


Fig. 6.5: *Heron Viewer showing O3 Measurements over time with timeslider*

6.2.2 SOS Web-Client

The SOS Web Client by 52North: http://sensors.geonovum.nl/jsclient accesses the SOS directly via the map and charts.

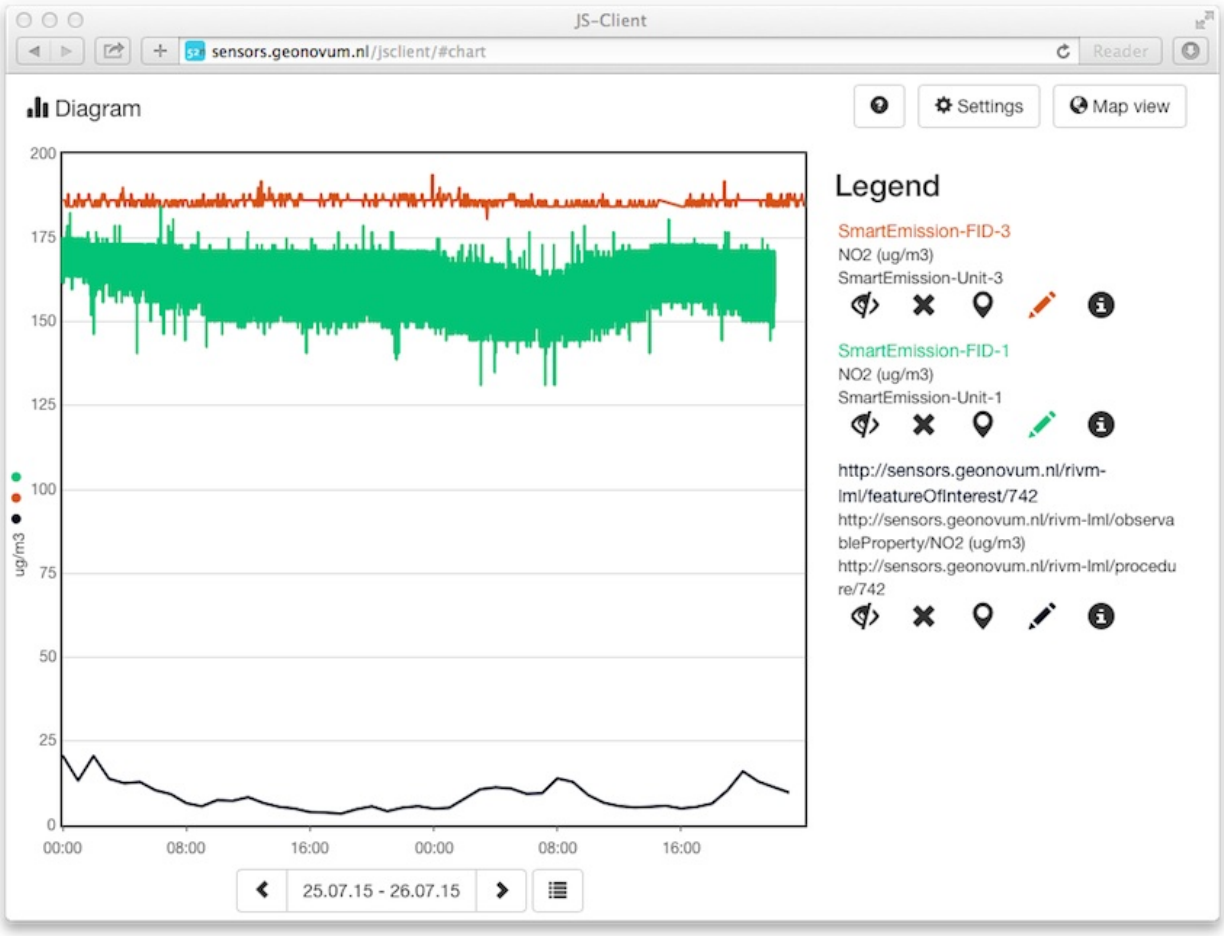


Fig. 6.6: *SOS Web Client showing NO2 Measurements in Chart*

The viewer is quite advanced and takes some time to get used to. It is possible to get charts and other views. Best is to follow the built-in tutorial. Custom charts can be made by selecting stations on the map (Map View) and adding these. The Smart Emission components are called O3, CO and NO2. The RIVM LML ones have longer names starting with `http:`. As can be seen the Smart Emission measurements are significantly higher. Also RIVM LML data is produced as an hourly average while Smart Emission data (now) provides all measurements. This is an item for improvement.

6.2.3 SOS-REST API

Easiest is looking at results via the SOS REST API.

The following are examples:

- http://sensors.geonovum.nl/sos/api/v1/stations REST API, Stations
- http://sensors.geonovum.nl/sos/api/v1/phenomena REST API, Phenomena
- http://sensors.geonovum.nl/sos/api/v1/timeseries REST API, All Time Series List

- http://sensors.geonovum.nl/sos/api/v1/timeseries/260 REST API, Single Time Series MetaData
- http://sensors.geonovum.nl/sos/api/v1/timeseries/100/getData?timespan=PT48H/2014-09-06 REST API, Time Series Data
- http://sensors.geonovum.nl/sos/api/v1/timeseries/260/getData?timespan=2015-07-21TZ/2015-07-28TZ REST API, Time Series Data

The remainder of this chapter describes the technical setup.

6.3 Architecture

Figure 2 sketches the overall SOSPilot architecture with emphasis on the flow of data (arrows). Circles depict harvesting/ETL processes. Server-instances are in rectangles. Datastores the "DB"-icons.

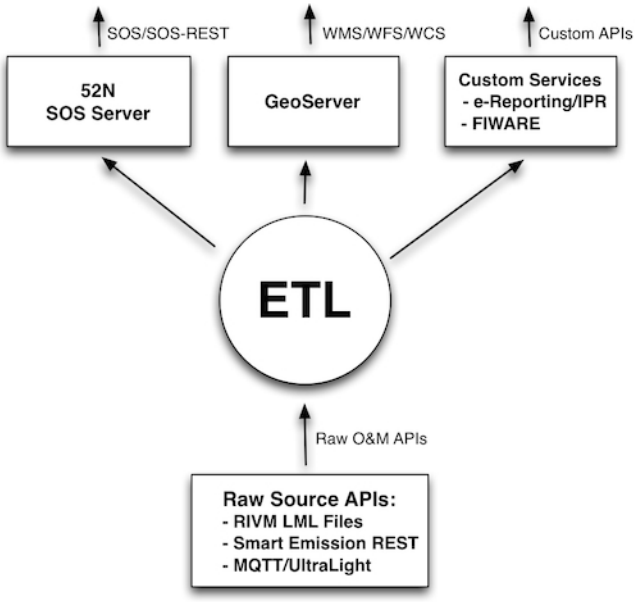


Fig. 6.7: *Figure 2 - Overall Architecture*

Figure 2 sketches the approach for RIVM LML AQ data, but this same approach was used voor Smart Emission. For "RIVM LML File Server" one should read: "Raw Smart Emission Sample Data".

6.4 ETL Design

In this section the ETL is elaborated in more detail as depicted in the figure below. Figure 3 sketches the same approach as earlier used for for RIVM LML AQ data. Step 1 and Step 2 are (partly) different from the RIVM LML ETL. Step 3 is identical to the RIVM LML ETL, an advantage of the multi-step ETL process now pays back! The main difference/extension to RIVM LML ETL processing is that the Smart Emission raw O&M data is not yet validated (e.g. has outliers) and aggregated (e.g. no hourly averages).

The ETL design comprises three main processing steps and three datastores. The three ETL Steps are:

1. O&M Harvester: fetch raw O&M data from CityGIS server via Sensor REST API

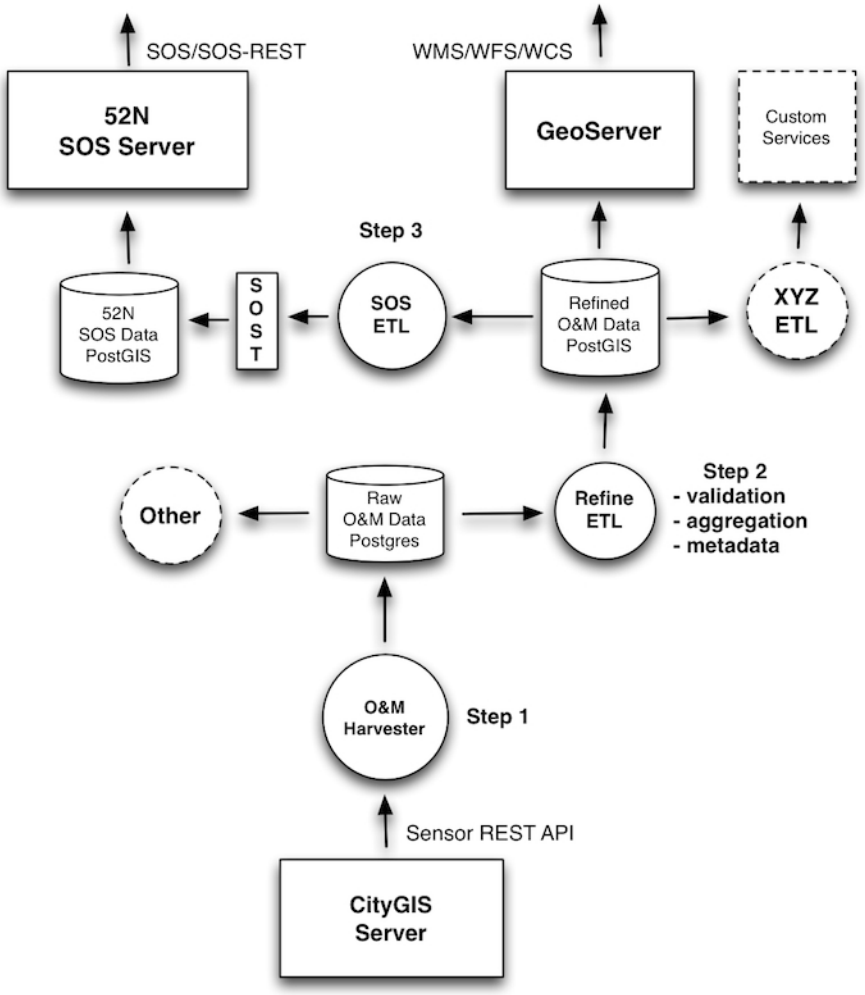


Fig. 6.8: *Figure 3 - Overall Architecture with ETL Steps*

2. Refine ETL: validate and aggregate the raw O&M data and transform to intermediate "Core AQ Data" in PostGIS
3. SOS ETL: transform and publish "Core AQ Data" to the 52N SOS DB via SOS-Transactions (SOS-T)

The detailed dataflow from source to destination is as follows:

1. raw O&M timeseries data is read by the *O&M Harvester* via Sensor REST API
2. *O&M Harvester* reads the timeseries response docs into the Raw O&M Data DB (Raw Measurements)
3. The Core AQ DB contains measurements + stations in regular tables 1-1 with original data, including a Time column
4. The Core AQ DB can be used for OWS (WMS/WFS) services via GeoServer (using VIEW by Measurements/Stations JOIN)
5. The SOS ETL process transforms core AQ data to SOS Observations and publishes Observations using SOS-T `InsertObservation`
6. These three processes run continuously (via cron)
7. Each process always knows its progress and where it needs to resume, even after it has been stopped (by storing a progress/checkpoint info)

These last two ETL processes manage their `last sync-time` using a separate `progress table` within the database. The first, the *O&M Harvester*, only needs to check if a particular timeseries for a specific device has already been stored.

Advantages of this approach:

- backups of source data possible
- incrementally build up of history past the last month
- in case of (design) errors we can always reset the 'progress timestamp(s)' and restart anew
- simpler ETL scripts than "all-in-one", e.g. from "Core AQ DB" to "52N SOS DB" may even be in plain SQL
- migration with changed in 52N SOS DB schema simpler
- prepared for op IPR/INSPIRE ETL (source is Core OM DB)
- OWS server (WMS/WFS evt WCS) can directly use op Core OM DB (possibly via Measurements/Stations JOIN VIEW evt, see below)

The Open Source ETL tool Stetl, Streaming ETL , is used for most of the transformation steps. Stetl provides standard modules for building an ETL Chain via a configuration file. This ETL Chain is a linkage of Input, Filter and Output modules. Each module is a Python class derived from Stetl base classes. In addition a developer may add custom modules where standard Stetl modules are not available or to specialize processing aspects.

Stetl has been used sucessfully to publish BAG (Dutch Addresses and Buildings) to INSPIRE Addresses via XSLT and WFS-T (to the `deegree WFS server`) but also for transformation of Dutch topography (Top10NL and BGT) to PostGIS. As Stetl is written in Python it is well-integrated with standard ETL and Geo-tools like GDAl/OGR, XSLT and PostGIS.

At runtime Stetl (via the `stetl` command) basically reads the config file, creates all modules and links their inputs and outputs. This also makes for an easy programming model as one only needs to concentrate on a single ETL step.

6.4.1 ETL Step 1. - Harvester

The Smart Emission FTP server provides measurements per sensor (unit) in text files. See figure below. The raw data records per unit are divided over multiple lines. See example below:

```
07/24/2015 07:25:41,P.UnitSerialnumber,1   # start record
07/24/2015 07:25:41,S.Longitude,5914103
07/24/2015 07:25:41,S.Latitude,53949942
07/24/2015 07:25:41,S.SatInfo,90889
07/24/2015 07:25:41,S.O3,163
07/24/2015 07:25:41,S.BottomSwitches,0
07/24/2015 07:25:41,S.RGBColor,16771990
07/24/2015 07:25:41,S.LightsensorBlue,92
07/24/2015 07:25:41,S.LightsensorGreen,144
07/24/2015 07:25:41,S.LightsensorRed,156
07/24/2015 07:25:41,S.AcceleroZ,753
07/24/2015 07:25:41,S.AcceleroY,516
07/24/2015 07:25:41,S.AcceleroX,510
07/24/2015 07:25:41,S.NO2,90
07/24/2015 07:25:41,S.CO,31755
07/24/2015 07:25:41,S.Altimeter,118
07/24/2015 07:25:41,S.Barometer,101101
07/24/2015 07:25:41,S.LightsensorBottom,26
07/24/2015 07:25:41,S.LightsensorTop,225
07/24/2015 07:25:41,S.Humidity,48618
07/24/2015 07:25:41,S.TemperatureAmbient,299425
07/24/2015 07:25:41,S.TemperatureUnit,305400
07/24/2015 07:25:41,S.SecondOfDay,33983
07/24/2015 07:25:41,S.RtcDate,1012101
07/24/2015 07:25:41,S.RtcTime,596503
07/24/2015 07:25:41,P.SessionUptime,60781
07/24/2015 07:25:41,P.BaseTimer,9
07/24/2015 07:25:41,P.ErrorStatus,0
07/24/2015 07:25:41,P.Powerstate,79
07/24/2015 07:25:51,P.UnitSerialnumber,1  # start record
07/24/2015 07:25:51,S.Longitude,5914103
07/24/2015 07:25:51,S.Latitude,53949942
07/24/2015 07:25:51,S.SatInfo,90889
07/24/2015 07:25:51,S.O3,157
07/24/2015 07:25:51,S.BottomSwitches,0
```

Each record starts on a line that contains `P.UnitSerialnumber` and runs to the next line containing `P.UnitSerialnumber` or the end-of-file is reached. Each record contains zero to three chemical component values named: `S.CO` (Carbon Monoxide), `S.NO2` (Nitrogen Dioxide) or `S.O3` (Ozone), and further fields such as location (`S.Latitude`, `S.Longitude`) and weather data (Temperature, Pressure). All fields have the same timestamp, e.g. `07/24/2015 07:25:41`. This value is taken as the timestamp of the record.

According to CityGIS the units are defined as follows.

```
S.TemperatureUnit            milliKelvin
S.TemperatureAmbient milliKelvin
S.Humidity                          %mRH
S.LightsensorTop             Lux
S.LightsensorBottom          Lux
S.Barometer                         Pascal
S.Altimeter                         Meter
S.CO                                ppb
S.NO2                                       ppb
S.AcceleroX                         2 ~ +2G (0x200 = midscale)
S.AcceleroY                         2 ~ +2G (0x200 = midscale)
S.AcceleroZ                         2 ~ +2G (0x200 = midscale)
S.LightsensorRed             Lux
S.LightsensorGreen           Lux
```

```
S.LightsensorBlue            Lux
S.RGBColor                           8 bit R, 8 bit G, 8 bit B
S.BottomSwitches             ?
S.O3                                 ppb
S.CO2                                        ppb
v3: S.ExternalTemp           milliKelvin
v3: S.COResistance           Ohm
v3: S.No2Resistance          Ohm
v3: S.O3Resistance           Ohm
S.AudioMinus5                        Octave -5 in dB(A)
S.AudioMinus4                        Octave -4 in dB(A)
S.AudioMinus3                        Octave -3 in dB(A)
S.AudioMinus2                        Octave -2 in dB(A)
S.AudioMinus1                        Octave -1 in dB(A)
S.Audio0                             Octave 0 in dB(A)
S.AudioPlus1                 Octave +1 in dB(A)
S.AudioPlus2                 Octave +2 in dB(A)
S.AudioPlus3                 Octave +3 in dB(A)
S.AudioPlus4                 Octave +4 in dB(A)
S.AudioPlus5                 Octave +5 in dB(A)
S.AudioPlus6                 Octave +6 in dB(A)
S.AudioPlus7                 Octave +7 in dB(A)
S.AudioPlus8                 Octave +8 in dB(A)
S.AudioPlus9                 Octave +9 in dB(A)
S.AudioPlus10                        Octave +10 in dB(A)
S.SatInfo
S.Latitude                           nibbles: n1:0=East/North, 8=West/South; n2&n3:␣
↪whole degrees (0-180); n4-n8: degree fraction (max 999999)
S.Longitude                          nibbles: n1:0=East/North, 8=West/South; n2&n3:␣
↪whole degrees (0-180); n4-n8: degree fraction (max 999999)

P.Powerstate                                 Power State
P.BatteryVoltage                             Battery Voltage (milliVolts)
P.BatteryTemperature                 Battery Temperature (milliKelvin)
P.BatteryGauge                                       Get Battery Gauge, BFFF =␣
↪Battery full, 1FFF = Battery fail, 0000 = No Battery Installed
P.MuxStatus                                          Mux Status (0-7=channel,
↪F=inhibited)
P.ErrorStatus                                        Error Status (0=OK)
P.BaseTimer                                          BaseTimer (seconds)
P.SessionUptime                                      Session Uptime (seconds)
P.TotalUptime                                        Total Uptime (minutes)
v3: P.COHeaterMode                           CO heater mode
P.COHeater                                           Powerstate CO heater (0/1)
P.NO2Heater                                          Powerstate NO2 heater (0/1)
P.O3Heater                                           Powerstate O3 heater (0/1)
v3: P.CO2Heater                                      Powerstate CO2 heater (0/1)
P.UnitSerialnumber                           Serialnumber of unit
P.TemporarilyEnableDebugLeds Debug leds (0/1)
P.TemporarilyEnableBaseTimer Enable BaseTime (0/1)
P.ControllerReset                            WIFI reset
P.FirmwareUpdate                             Firmware update, reboot to bootloader

Unknown at this moment (decimal):
P.11
P.16
P.17
P.18
```

Conversion for latitude/longitude:

```
/*
            8 nibbles:
            MSB                 LSB
            n1 n2 n3 n4 n5 n6 n7 n8
            n1: 0 of 8, 0=East/North, 8=West/South
            n2 en n3: whole degrees (0-180)
            n4-n8: fraction of degrees (max 999999)
*/
private convert(input: number): number {
    var sign = input >> 28 ? -1 : +1;
    var deg = (input >> 20) & 255;
    var dec = input & 1048575;

    return (deg + dec / 1000000) * sign;
}
```

As stated above: this step, acquiring/harvesting files, was initially done via FTP.

6.4.2 ETL Step 2 - Raw Measurements

This step produces raw AQ measurements, "AQ ETL" in Figure 2, from raw source (file) data harvested in Step 1. The results of this step can be accessed via WMS and WFS, directly in the project Heron viewer: http://sensors.geonovum.nl/heronviewer

Two tables: `stations` and `measurements`. This is a 1:1 transformation from the raw text. The `measurements` refers to the `stations` by a FK `unit_id`.

Stations

Station info has been assembled in a CSV file: https://github.com/Geonovum/sospilot/tree/master/src/smartem/stations.csv

```
UnitId,Name,Municipality,Lat,Lon,Altitude,AltitudeUnit
1,Nijmegen-1,Nijmegen,51.94,5.90,30,m
3,Nijmegen-3,Nijmegen,51.80,6.00,30,m
5,Nijmegen-5,Nijmegen,51.85,5.95,30,m
7,Nijmegen-7,Nijmegen,51.91,6.10,30,m
8,Nijmegen-8,Nijmegen,51.87,5.80,30,m
9,Nijmegen-9,Nijmegen,51.92,6.20,30,m
10,Nijmegen-10,Nijmegen,51.89,5.85,30,m
```

This info was deducted from the raw measurements files. NB: the Lat,Lon values were inaccurate. This is still under investigation. **For the sake of the project Lat,Lon values have been randomly altered here!**. This will need to be corrected at a later stage.

Test by viewing in http://sensors.geonovum.nl/heronviewer See result (pink-purple triangles). Clicking on a station provides more detailed info via WMS `GetFeatureInfo`.

Measurements

Reading raw measurements from the files is done with a `Stetl` process. A specific Stetl Input module was developed to effect reading and parsing the files and tracking the last id of the file processed. https://github.com/Geonovum/

gid	point	unit_id	name	municipality	altitude	altitude_unit	lon	lat
1	0101000020E61000009A99999999991740B81E85EB51F84940	1	Nijmegen-1	Nijmegen	30	m	5.90	51.94
2	0101000020E6100000000000000000184066666666666E64940	3	Nijmegen-3	Nijmegen	30	m	6.00	51.80
3	0101000020E6100000CDCCCCCCCCCC1740CDCCCCCCCCEC4940	5	Nijmegen-5	Nijmegen	30	m	5.95	51.85
4	0101000020E6100000666666666666184014AE47E17AF44940	7	Nijmegen-7	Nijmegen	30	m	6.10	51.91
5	0101000020E6100000333333333333317408FC2F5285CEF4940	8	Nijmegen-8	Nijmegen	30	m	5.80	51.87
6	0101000020E6100000CDCCCCCCCCCC1840F6285C8FC2F54940	9	Nijmegen-9	Nijmegen	30	m	6.20	51.92
7	0101000020E6100000666666666666174052B81E85EBF14940	10	Nijmegen-10	Nijmegen	30	m	5.85	51.89

Fig. 6.9: *Stations Read into Postgres/PostGIS*

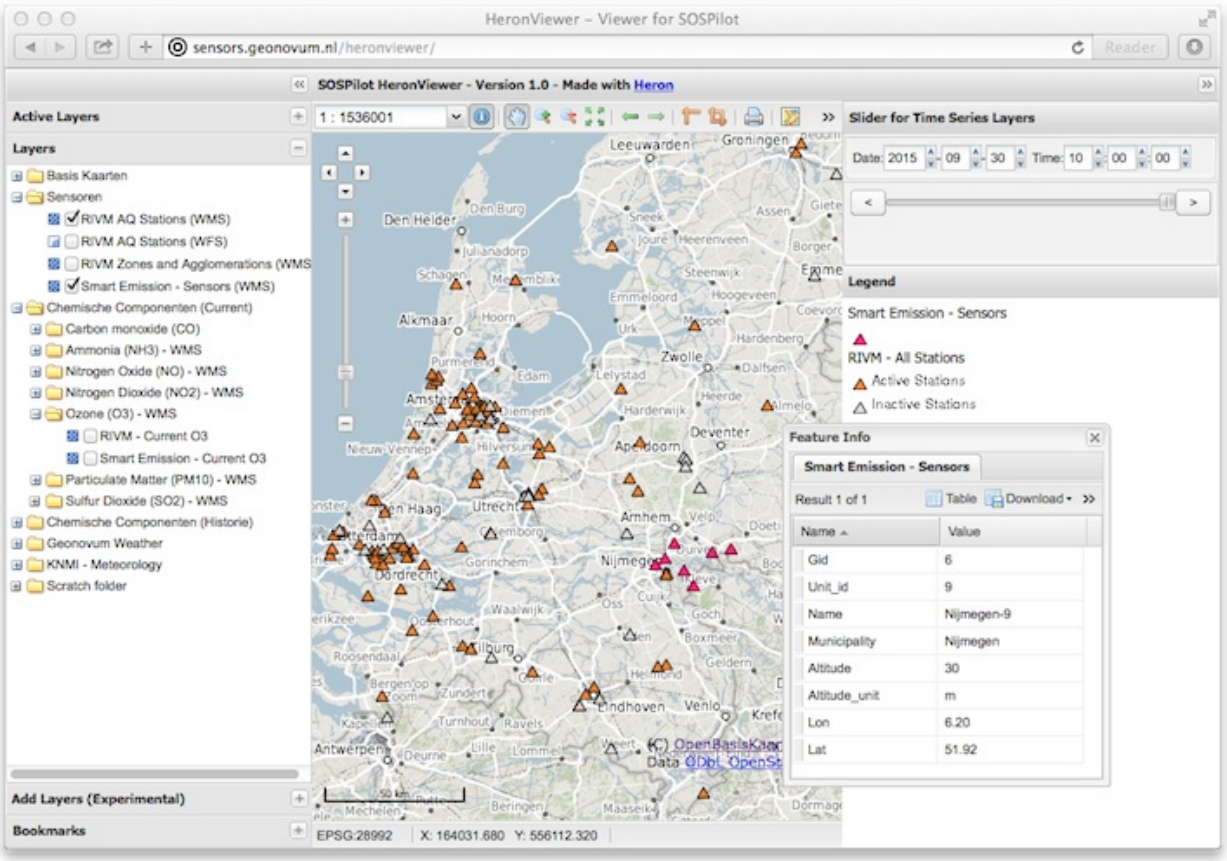


Fig. 6.10: *Smart Emission Stations in Heron Viewer*

sospilot/blob/master/src/smartem/raw2measurements.py These are two Filters: the class `Raw2RecordFilter` converts raw lines from the file to raw records. The class `Record2MeasurementsFilter` converts these records to records to be inserted into the `measurements` table. Other components used are standard Stetl.

Unit Conversion: as seen above the units for chemical components are in `ppb` (Parts-Per-Billion). For AQ data the usual unit is ug/m3 (Microgram per cubic meter). The conversion from ppb to ug/m3 is well-known and is dependent on molecular weight, temperature and pressure. See more detail here: http://www.apis.ac.uk/unit-conversion. Some investigation:

```
# Zie http://www.apis.ac.uk/unit-conversion
# ug/m3 = PPB * moleculair gewicht/moleculair volume
# waar molec vol = 22.41 * T/273 * 1013/P
#
# Typical values:
# Nitrogen dioxide 1 ppb = 1.91 ug/m3  bij 10C 1.98, bij 30C 1.85 --> 1.9
# Ozone 1 ppb = 2.0 ug/m3  bij 10C 2.1, bij 30C 1.93 --> 2.0
# Carbon monoxide 1 ppb = 1.16 ug/m3 bij 10C 1.2, bij 30C 1.1 --> 1.15
#
# Benzene 1 ppb = 3.24 ug/m3
# Sulphur dioxide 1 ppb = 2.66 ug/m3
#
```

For now a crude approximation as the measurements themselves are also not very accurate (another issue). In raw2measurements.py:

```
record['sample_value'] = Record2MeasurementsFilter.ppb_to_ugm3_factor[component_name]␣
↪* ppb_val
```

with `Record2MeasurementsFilter.ppb_to_ugm3_factor`:

```
# For now a crude conversion (1 atm, 20C)
ppb_to_ugm3_factor = {'o3': 2.0, 'no2': 1.9, 'co': 1.15}
```

The entire Stetl process is defined in https://github.com/Geonovum/sospilot/blob/master/src/smartem/files2measurements.cfg

The invokation of that Stetl process is via shell script: https://github.com/Geonovum/sospilot/blob/master/src/smartem/files2measurements.sh

The data is stored in the `measurements` table, as below. `station_id` is a foreign key into the `stations` table corresponding to a `unit_id`.

Using a Postgres VIEW the two tables can be combined via an `INNER JOIN` to provide measurements with location. This VIEW can be used as a WMS/WFS data source in GeoServer.

The VIEW is defined in https://github.com/Geonovum/sospilot/blob/master/src/smartem/db/db-schema.sql:

```
CREATE VIEW smartem.measurements_stations AS
   SELECT m.gid, m.station_id, s.name, s.municipality, m.component, m.sample_time, m.
↪sample_value,
   m.sample_value_ppb, s.point, s.lon, s.lat,m.insert_time, m.sample_id,s.unit_id, s.
↪altitude
          FROM smartem.measurements as m
            INNER JOIN smartem.stations as s ON m.station_id = s.unit_id;
```

Other detailed VIEWs provide virtual tables like Last Measurements and Measurements per component (see the DB schema and the Heron viewer).

gid	insert_time	component	station_id	sample_id	sample_time	sample_value_ppb	sample_value
211	2015-09-30 11:21:38.892321	O3	1	1-o3-2015-07-24 07:37:30	2015-07-24 07:37:30	155	310
212	2015-09-30 11:21:38.892872	NO2	1	1-no2-2015-07-24 07:37:30	2015-07-24 07:37:30	91	172.9
213	2015-09-30 11:21:38.893456	CO	1	1-co-2015-07-24 07:37:30	2015-07-24 07:37:30	31473	36193.9
214	2015-09-30 11:21:38.895276	O3	1	1-o3-2015-07-24 07:37:41	2015-07-24 07:37:41	160	320
215	2015-09-30 11:21:38.895827	NO2	1	1-no2-2015-07-24 07:37:41	2015-07-24 07:37:41	85	161.5
216	2015-09-30 11:21:38.896552	CO	1	1-co-2015-07-24 07:37:41	2015-07-24 07:37:41	31699	36453.9
217	2015-09-30 11:21:38.898386	O3	1	1-o3-2015-07-24 07:37:52	2015-07-24 07:37:52	163	326
218	2015-09-30 11:21:38.89953	NO2	1	1-no2-2015-07-24 07:37:52	2015-07-24 07:37:52	91	172.9
219	2015-09-30 11:21:38.900312	CO	1	1-co-2015-07-24 07:37:52	2015-07-24 07:37:52	32217	37049.6
220	2015-09-30 11:21:38.902308	O3	1	1-o3-2015-07-24 07:38:03	2015-07-24 07:38:03	158	316
221	2015-09-30 11:21:38.902881	NO2	1	1-no2-2015-07-24 07:38:03	2015-07-24 07:38:03	90	171
222	2015-09-30 11:21:38.903954	CO	1	1-co-2015-07-24 07:38:03	2015-07-24 07:38:03	32482	37354.3
223	2015-09-30 11:21:38.906028	O3	1	1-o3-2015-07-24 07:38:14	2015-07-24 07:38:14	158	316
224	2015-09-30 11:21:38.906631	NO2	1	1-no2-2015-07-24 07:38:14	2015-07-24 07:38:14	90	171
225	2015-09-30 11:21:38.907218	CO	1	1-co-2015-07-24 07:38:14	2015-07-24 07:38:14	32295	37139.2
226	2015-09-30 11:21:38.909016	O3	1	1-o3-2015-07-24 07:38:25	2015-07-24 07:38:25	157	314
227	2015-09-30 11:21:38.909609	NO2	1	1-no2-2015-07-24 07:38:25	2015-07-24 07:38:25	92	174.8
228	2015-09-30 11:21:38.910141	CO	1	1-co-2015-07-24 07:38:25	2015-07-24 07:38:25	31746	36507.9
229	2015-09-30 11:21:38.911906	O3	1	1-o3-2015-07-24 07:38:36	2015-07-24 07:38:36	161	322
230	2015-09-30 11:21:38.912471	NO2	1	1-no2-2015-07-24 07:38:36	2015-07-24 07:38:36	91	172.9
231	2015-09-30 11:21:38.913065	CO	1	1-co-2015-07-24 07:38:36	2015-07-24 07:38:36	32479	37350.9
232	2015-09-30 11:21:38.914854	O3	1	1-o3-2015-07-24 07:38:47	2015-07-24 07:38:47	157	314
233	2015-09-30 11:21:38.915412	NO2	1	1-no2-2015-07-24 07:38:47	2015-07-24 07:38:47	90	171
234	2015-09-30 11:21:38.916007	CO	1	1-co-2015-07-24 07:38:47	2015-07-24 07:38:47	31894	36678.1
235	2015-09-30 11:21:38.917828	O3	1	1-o3-2015-07-24 07:38:58	2015-07-24 07:38:58	160	320
236	2015-09-30 11:21:38.918389	NO2	1	1-no2-2015-07-24 07:38:58	2015-07-24 07:38:58	92	174.8
237	2015-09-30 11:21:38.918988	CO	1	1-co-2015-07-24 07:38:58	2015-07-24 07:38:58	32227	37061.1
238	2015-09-30 11:21:38.920986	O3	1	1-o3-2015-07-24 07:39:09	2015-07-24 07:39:09	163	326
239	2015-09-30 11:21:38.921631	NO2	1	1-no2-2015-07-24 07:39:09	2015-07-24 07:39:09	90	171
240	2015-09-30 11:21:38.922217	CO	1	1-co-2015-07-24 07:39:09	2015-07-24 07:39:09	31683	36435.4

Fig. 6.11: *Smart Emission raw measurements stored in Postgres*

gid	station_id	name	municipality	component	sample_time	sample_value	sample_value_ppb	point	lon	lat	insert_time	sample_id	unit_id	altitude
331	3	Nijmegen-3	Nijmegen	O3	2015-07-18 08:39:05	254	127	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.021936	3-o3-2015-07-18 08:39:05	3	30
332	3	Nijmegen-3	Nijmegen	NO2	2015-07-18 08:39:05	174.8	92	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.028652	3-no2-2015-07-18 08:39:05	3	30
333	3	Nijmegen-3	Nijmegen	O3	2015-07-18 08:39:11	264	132	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.039546	3-o3-2015-07-18 08:39:11	3	30
334	3	Nijmegen-3	Nijmegen	NO2	2015-07-18 08:39:11	174.8	92	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.046312	3-no2-2015-07-18 08:39:11	3	30
335	3	Nijmegen-3	Nijmegen	O3	2015-07-18 08:43:13	266	133	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.059871	3-o3-2015-07-18 08:43:13	3	30
336	3	Nijmegen-3	Nijmegen	NO2	2015-07-18 08:43:13	174.8	92	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.066661	3-no2-2015-07-18 08:43:13	3	30
337	3	Nijmegen-3	Nijmegen	O3	2015-07-18 08:58:57	256	128	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.079104	3-o3-2015-07-18 08:58:57	3	30
338	3	Nijmegen-3	Nijmegen	NO2	2015-07-18 08:58:57	174.8	92	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.085518	3-no2-2015-07-18 08:58:57	3	30
339	3	Nijmegen-3	Nijmegen	O3	2015-07-18 08:59:15	254	127	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.096455	3-o3-2015-07-18 08:59:15	3	30
340	3	Nijmegen-3	Nijmegen	NO2	2015-07-18 08:59:15	174.8	92	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.103282	3-no2-2015-07-18 08:59:15	3	30
341	3	Nijmegen-3	Nijmegen	O3	2015-07-18 09:03:15	264	132	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.109855	3-o3-2015-07-18 09:03:15	3	30
342	3	Nijmegen-3	Nijmegen	NO2	2015-07-18 09:03:15	174.8	92	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.11144	3-no2-2015-07-18 09:03:15	3	30
343	3	Nijmegen-3	Nijmegen	O3	2015-07-18 09:07:15	262	131	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.118779	3-o3-2015-07-18 09:07:15	3	30
344	3	Nijmegen-3	Nijmegen	NO2	2015-07-18 09:07:15	172.9	91	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.119432	3-no2-2015-07-18 09:07:15	3	30
345	3	Nijmegen-3	Nijmegen	O3	2015-07-18 09:11:16	256	128	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.125517	3-o3-2015-07-18 09:11:16	3	30
346	3	Nijmegen-3	Nijmegen	NO2	2015-07-18 09:11:16	174.8	92	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.126167	3-no2-2015-07-18 09:11:16	3	30
347	3	Nijmegen-3	Nijmegen	O3	2015-07-18 09:15:16	264	132	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.132039	3-o3-2015-07-18 09:15:16	3	30
348	3	Nijmegen-3	Nijmegen	NO2	2015-07-18 09:15:16	172.9	91	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.132622	3-no2-2015-07-18 09:15:16	3	30
349	3	Nijmegen-3	Nijmegen	O3	2015-07-18 09:19:16	250	125	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.13849	3-o3-2015-07-18 09:19:16	3	30
350	3	Nijmegen-3	Nijmegen	NO2	2015-07-18 09:19:16	172.9	91	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.13928	3-no2-2015-07-18 09:19:16	3	30
351	3	Nijmegen-3	Nijmegen	O3	2015-07-18 09:23:16	258	129	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.145109	3-o3-2015-07-18 09:23:16	3	30
352	3	Nijmegen-3	Nijmegen	NO2	2015-07-18 09:23:16	174.8	92	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.145667	3-no2-2015-07-18 09:23:16	3	30
353	3	Nijmegen-3	Nijmegen	O3	2015-07-18 09:27:17	270	135	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.151815	3-o3-2015-07-18 09:27:17	3	30
354	3	Nijmegen-3	Nijmegen	NO2	2015-07-18 09:27:17	172.9	91	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.152642	3-no2-2015-07-18 09:27:17	3	30
355	3	Nijmegen-3	Nijmegen	O3	2015-07-18 09:31:17	254	127	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.159646	3-o3-2015-07-18 09:31:17	3	30
356	3	Nijmegen-3	Nijmegen	NO2	2015-07-18 09:31:17	172.9	91	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.16036	3-no2-2015-07-18 09:31:17	3	30
357	3	Nijmegen-3	Nijmegen	O3	2015-07-18 09:35:17	274	137	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.166174	3-o3-2015-07-18 09:35:17	3	30
358	3	Nijmegen-3	Nijmegen	NO2	2015-07-18 09:35:17	172.9	91	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.166742	3-no2-2015-07-18 09:35:17	3	30
359	3	Nijmegen-3	Nijmegen	O3	2015-07-18 09:39:17	258	129	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.172598	3-o3-2015-07-18 09:39:17	3	30
360	3	Nijmegen-3	Nijmegen	NO2	2015-07-18 09:39:17	172.9	91	0101000020E6100000000000000000001840666666666E64940	6.00	51.80	2015-09-30 11:21:39.173232	3-no2-2015-07-18 09:39:17	3	30

Fig. 6.12: *Postgres VIEW combining measurements and stations (units)*

6.4.3 ETL Step 3 - SOS Publication

In this step the Raw Measurements data (see Step 2) is transformed to "SOS Ready Data", i.e. data that can be handled by the 52North SOS server. This is done via SOS Transaction (SOS-T) services using `Stetl`.

SOS Publication - Stetl Strategy

As Stetl only supports WFS-T, not yet SOS, a SOS Output module `sosoutput.py` was developed derived from the standard `httpoutput.py` module. See https://github.com/Geonovum/sospilot/blob/master/src/smartem/sosoutput.py (this version was slightly adapted from the version used for RIVM LML).

Most importantly, the raw Smart Emission Measurements data from Step 2 needs to be transformed to OWS Observations & Measurements (O&M) data. This is done via `substitutable templates`, like the Stetl config itself also applies. This means we develop files with SOS Requests in which all variable parts get a symbolic value like `{sample_value}`. These templates can be found under https://github.com/Geonovum/sospilot/tree/master/src/smartem/sostemplates in particular

- https://github.com/Geonovum/sospilot/blob/master/src/smartem/sostemplates/insert-sensor.json InsertSensor
- https://github.com/Geonovum/sospilot/blob/master/src/smartem/sostemplates/delete-sensor.json DeleteSensor
- https://github.com/Geonovum/sospilot/blob/master/src/smartem/sostemplates/procedure-desc.xml Sensor ML
- https://github.com/Geonovum/sospilot/blob/master/src/smartem/sostemplates/insert-observation.json InsertObservation

Note that we use JSON for the requests, as this is simpler than XML. The Sensor ML is embedded in the `insert-sensor` JSON request.

SOS Publication - Sensors

This step needs to be performed only once, or when any of the original Station data (CSV) changes.

The Stetl config https://github.com/Geonovum/sospilot/blob/master/src/smartem/stations2sensors.cfg uses a Standard Stetl module, `inputs.dbinput.PostgresDbInput` for obtaining Record data from a Postgres database.

```
{{
  "request": "InsertSensor",
  "service": "SOS",
  "version": "2.0.0",
  "procedureDescriptionFormat": "http://www.opengis.net/sensorML/1.0.1",
  "procedureDescription": "{procedure-desc.xml}",
   "observableProperty": [
     "CO",
     "NO2",
     "O3"
  ],
  "observationType": [
     "http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement"
  ],
  "featureOfInterestType": "http://www.opengis.net/def/samplingFeatureType/OGC-OM/2.0/
↪SF_SamplingPoint"
}}
```

The SOSTOutput module will expand `{procedure-desc.xml}` with the Sensor ML template from https://github.com/Geonovum/sospilot/blob/master/src/smartem/sostemplates/procedure-desc.xml.

SOS Publication - Observations

The Stetl config https://github.com/Geonovum/sospilot/blob/master/src/smartem/measurements2sos.cfg uses an extended Stetl module (`inputs.dbinput.PostgresDbInput`) for obtaining Record data from a Postgres database: https://github.com/Geonovum/sospilot/blob/master/src/smartem/measurementsdbinput.py. This is required to track progress in the `etl_progress` table similar as in Step 2. The `last_id` is remembered.

The Observation template looks as follows.

```
{{
  "request": "InsertObservation",
  "service": "SOS",
  "version": "2.0.0",
  "offering": "SmartEmission-Offering-{unit_id}",
  "observation": {{
    "identifier": {{
      "value": "{sample_id}",
      "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown"
    }},
    "type": "http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement",
    "procedure": "SmartEmission-Unit-{unit_id}",
    "observedProperty": "{component}",
    "featureOfInterest": {{
      "identifier": {{
        "value": "SmartEmission-FID-{unit_id}",
        "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown"
      }},
      "name": [
        {{
          "value": "{municipality}",
          "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown"
        }}
      ],
      "geometry": {{
        "type": "Point",
        "coordinates": [
          {lat},
          {lon}
        ],
        "crs": {{
          "type": "name",
          "properties": {{
            "name": "EPSG:4326"
          }}
        }}
      }}
    }},
    "phenomenonTime": "{sample_time}",
    "resultTime": "{sample_time}",
    "result": {{
      "uom": "ug/m3",
      "value": {sample_value}
    }}
  }}
}}
```

It is quite trivial in `sosoutput.py` to substitute these values from the `measurements`-table records.

Like in ETL Step 2 the progress is remembered in the table `rivm_lml.etl_progress` by updating the `last_id`

field after publication, where that value represents the `gid` value of `rivm_lml.measurements`.

SOS Publication - Results

Via the standard SOS protocol the results can be tested:

- GetCapabilities: http://sensors.geonovum.nl/sos/service?service=SOS&request=GetCapabilities
- DescribeSensor (station 807, Hellendoorn): http://tinyurl.com/mmsr9hl (URL shortened)
- GetObservation: http://tinyurl.com/ol82sxv (URL shortened)

Easier is looking at results via the SOS REST API. The following are examples:

- http://sensors.geonovum.nl/sos/api/v1/stations REST API, Stations
- http://sensors.geonovum.nl/sos/api/v1/phenomena REST API, Phenomena
- http://sensors.geonovum.nl/sos/api/v1/timeseries REST API, All Time Series List
- http://sensors.geonovum.nl/sos/api/v1/timeseries/260 REST API, Single Time Series MetaData
- http://sensors.geonovum.nl/sos/api/v1/timeseries/100/getData?timespan=PT48H/2014-09-06 REST API, Time Series Data
- http://sensors.geonovum.nl/sos/api/v1/timeseries/260/getData?timespan=2015-07-21TZ/2015-07-28TZ REST API, Time Series Data

CHAPTER 7

Applying RIO for AQ Data

Starting in october 2015 the SOSPilot platform was extended to handle Air Quality data using the RIO model.

RIO is an interpolation method/tool for Air Quality data. Get a global overview from the IrCELine website and the article "Spatial interpolation of air pollution measurements using CORINE land cover data" for scientific details.

The RIO tools are used by RIVM, see the RIVM website:

"Het RIVM presenteert elk uur de actuele luchtkwaliteitskaarten voor stikstofdioxide, ozon en fijn stof in Nederland op de website van het Landelijk Meetnet Luchtkwaliteit (LML). Meetgegevens van representatieve LML-locaties worden gebruikt om te berekenen wat de concentraties voor de rest van Nederland, buiten de meetpunten om, zijn. De huidige rekenmethode om deze kaarten te maken, genaamd INTERPOL, heeft een aantal beperkingen. Hierdoor worden bijvoorbeeld stikstofdioxideconcentraties in stedelijk gebied vaak onderschat.

In België is een betere interpolatiemethode ontwikkeld. Deze methode, de RIO (residual interpolation optimised for ozone) interpolatiemethode, wordt daar gebruikt voor luchtkwaliteitskaarten die het publiek informeren over de actuele luchtkwaliteit. In 2009 is de RIO interpolatiemethode in opdracht van het RIVM zodanig uitgebreid en aangepast dat hij ook in Nederland kan worden gebruikt."

A RIO toolkit, has been developed by VITO, a leading European independent research and technology organisation based in Belgium. The toolkit is implemented in MatLab. Basically the RIO-tool converts Air Quality measurement data (CSV) from a set of dispersed stations to country-wide (RIO needs to be tailored per country) coverage data. RIVM uses RIO to produce and publish daily maps of Air Quality data for NO2, O3 and PM10, like the image below.

7.1 The Plan

The plan within the SOSPilot project is as follows:

1. apply the RIO model/tool to crowd-sourced air quality measurements, in particular from the *Smart Emission Project* Project
2. unlock RIO output via an OGC Web Coverage Service (WCS)

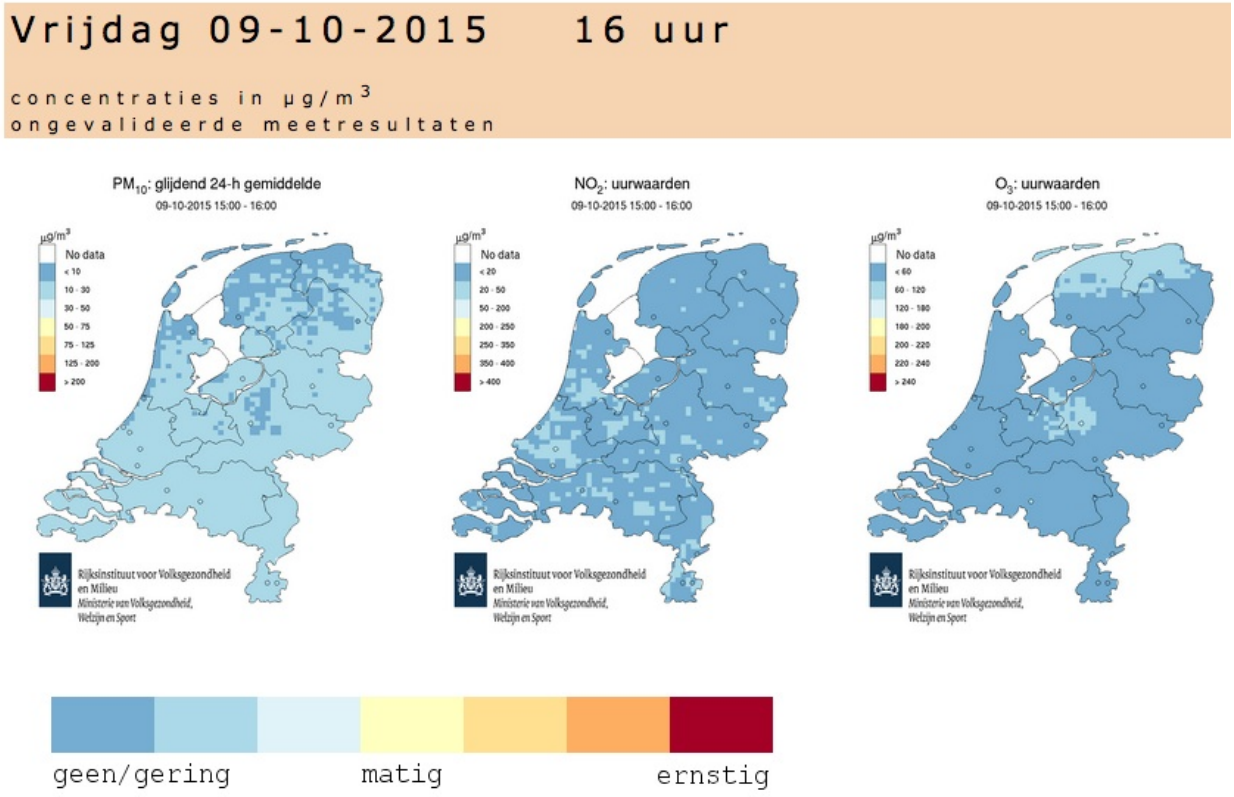


Fig. 7.1: *Website www.lml.rivm.nl - dispersion maps produced with Rio*

Once the above setup is working, various use-cases will be established, for example combining AQ coverage data with other INSPIRE themes like population and Netherlands-Belgium cross-border AQ data dispersion.

WCS is also a protocol for INSPIRE Download Services (together with WFS and SOS). This is also the main reason for the above plan. The RIO model also uses data from other INSPIRE themes, in particular land coverage (CORINE).

See also an overview in this presentation (Grothe 2015) from which the image below was taken.

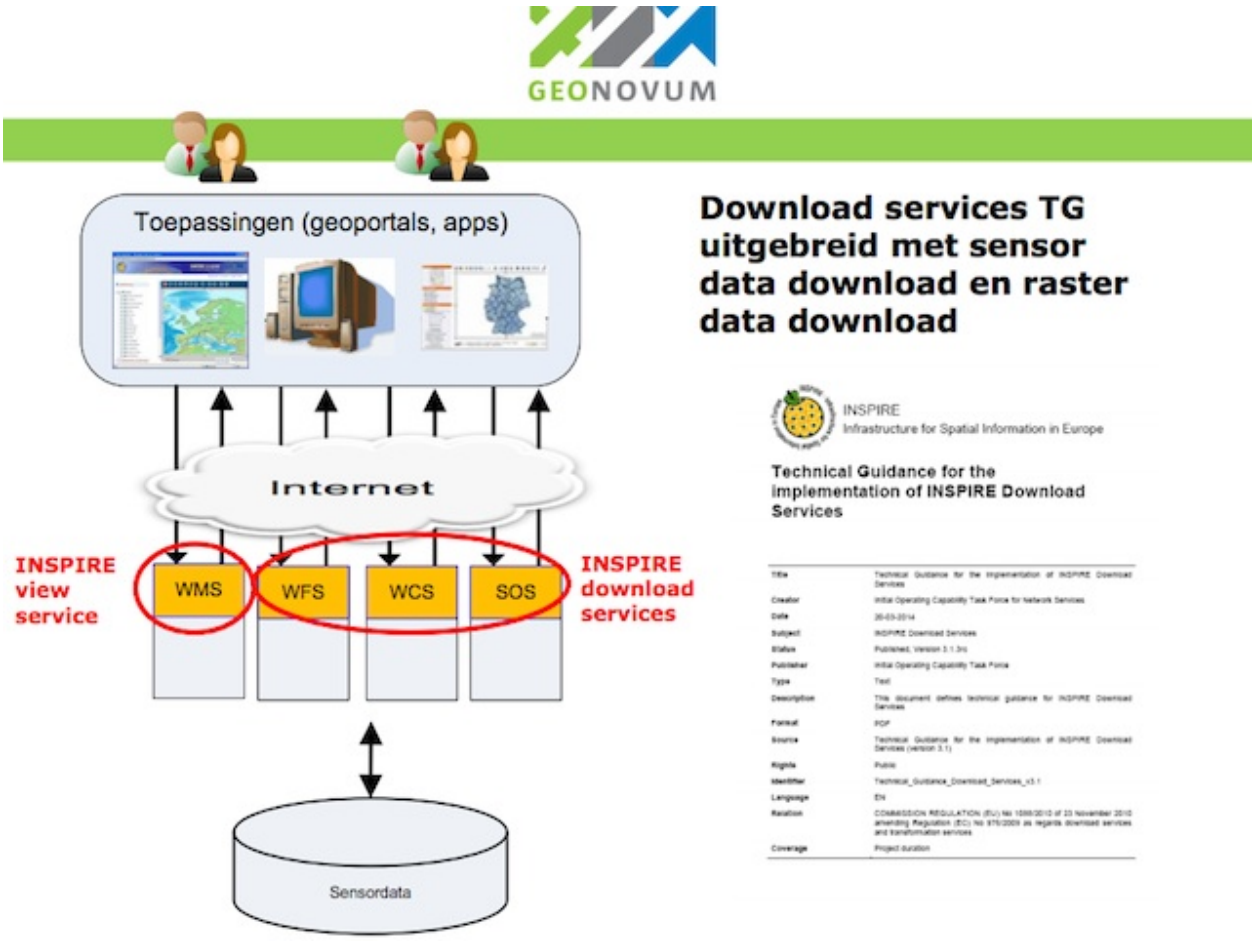


Fig. 7.2: *INSPIRE View and Download Services*

7.2 RIO Output via WCS

This was done as a first step. RIVM has kindly provided us with some RIO input and output datafiles plus documentation. These can be found here. The `aps2raster` directory contains the resulting GeoTIFF files. The work was performed under this GitHub issue.

7.2.1 APS Files

RIO output files are text-files in the so-called APS format. An APS file starts with a single line of metadata. Its format is described in the APS-header documentation. Subsequent lines are the rows forming a 2-dimensional array with the interpolated values.

Below is an example APS header:

```
# APS Header
# Y  M  D  H  C   unit  sys   comment  format  proj  orig_x  orig_y  cols rows  pix_w␣
↪ pix_h
# 15 9 16 10 NO2 ug/m3 RIO  uurwaarden f7.1    1     0.000   620.000 70    80   4.000␣
↪ 4.000
```

The metadata gives enough information to construct a GeoTIFF. The most important parameters are

- origin X, Y of coverage
- pixel width and height
- date and time (hour) of the measurements
- the chemical component (here NO2)

The data values are space-separated values. Empty (NoData) values are denoted with `-999.0`:

```
... -999.0   23.4   23.6 -999.0   24.2   17.3   14.3 ...
...   16.1   17.3   18.4   19.4   21.6   21.9   20.1 ...
```

7.2.2 Converting to GeoTIFF

Developing a simple Python program called aps2raster.py, the `.aps` files are converted to GeoTIFF files. `aps2raster.py` reads an APS file line by line. From the first line the metadata elements are parsed. From the remaining lines a 2-dimensional array of floats is populated. Finally a GeoTIFF file is created with the data and metadata. Projection is always in RD/EPSG:28992.

aps2raster.py uses the GDAL and NumPy Python libraries.

7.2.3 WMS with Raster Styling

Using GeoServer the GeoTIFFs are added as datasources and one layer per GeoTIFF is published. Within GeoServer this automatically creates both a WCS and a WMS layer. As the GeoTIFF does not contain colors but data values, styling is needed to view the WMS layer. This has been done via Styled Layer Descriptor (SLD) files. GeoServer supports SLDs for Rasterdata styling. In order to render the same colors as the RIVM LML daily images a value-interval color-mapping was developed as in this example for NO2:

```
<FeatureTypeStyle>
    <FeatureTypeName>Feature</FeatureTypeName>
    <Rule>
        <RasterSymbolizer>
            <ColorMap type="intervals">
                <ColorMapEntry color="#FFFFFF" quantity="0" label="No Data" opacity=
↪"0.0"/>
                <ColorMapEntry color="#6699CC" quantity="20" label="&lt; 20 ug/m3"␣
↪opacity="1.0"/>
                <ColorMapEntry color="#99CCCC" quantity="50" label="20-50 ug/m3"␣
↪opacity="1.0"/>
                <ColorMapEntry color="#CCFFFF" quantity="200" label="50-200 ug/m3"␣
↪opacity="1.0"/>
                <ColorMapEntry color="#FFFFCC" quantity="250" label="200-250 ug/m3"␣
↪opacity="1.0"/> <!-- Yellow -->
                <ColorMapEntry color="#FFCC66" quantity="350" label="250-350 ug/m3"␣
↪opacity="1.0"/>
```

```
                <ColorMapEntry color="#FF9966" quantity="400" label="350-400 ug/m3"␣
↪opacity="1.0"/>
                <ColorMapEntry color="#990033" quantity="20000" label="&gt; 400 ug/m3
↪" opacity="1.0"/>
            </ColorMap>
        </RasterSymbolizer>
    </Rule>
</FeatureTypeStyle>
```

The SLD files can be found here.

The WMS layers have been added to the existing SOSPilot Heron viewer.

7.2.4 Comparing with RIVM LML

The resulting WMS layers can now be compared to the original PNG files that came with the APS data files from RIVM. Below two examples for NO2 and PM10.

For NO2 below.

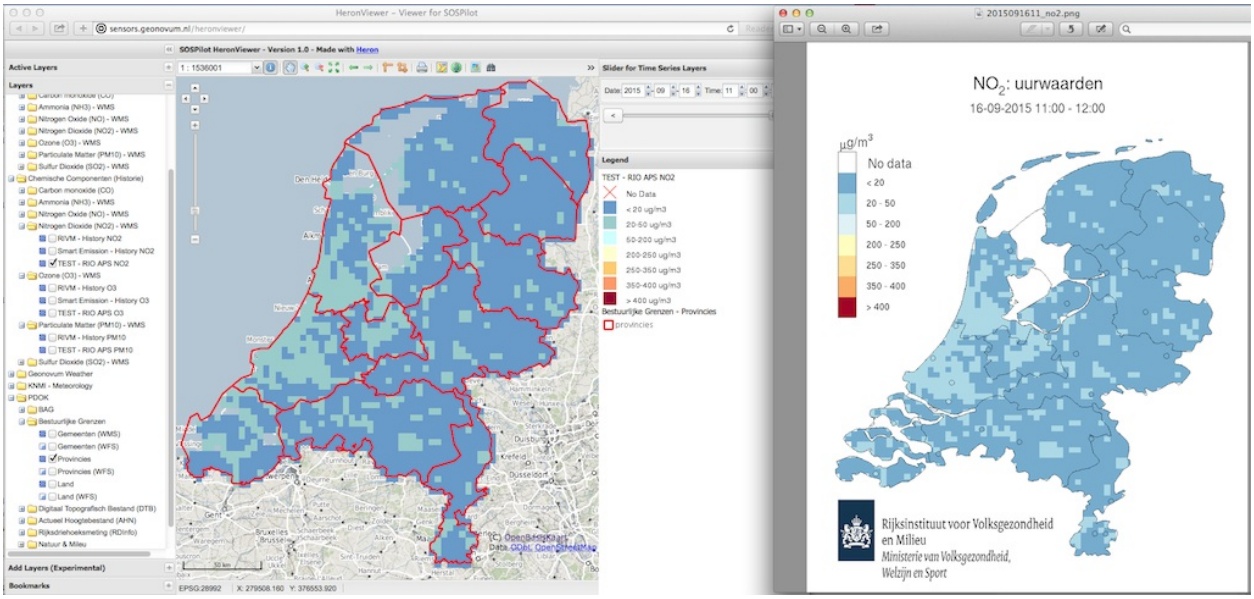


Fig. 7.3: *Comparing with RIVM LML maps (right) - NO2*

And for PM10 below.

As can be seen the maps are identical. Our WMS-maps are on the left, RIVM maps on the right. Our WMS maps are somewhat "rougher" on the edges since we have not cut-out using the coastlines.

7.2.5 WCS in QGIS

The Layers published in GeoServer can be viewed in QGIS by adding a WCS Layer via the standard WCS support in QGIS.

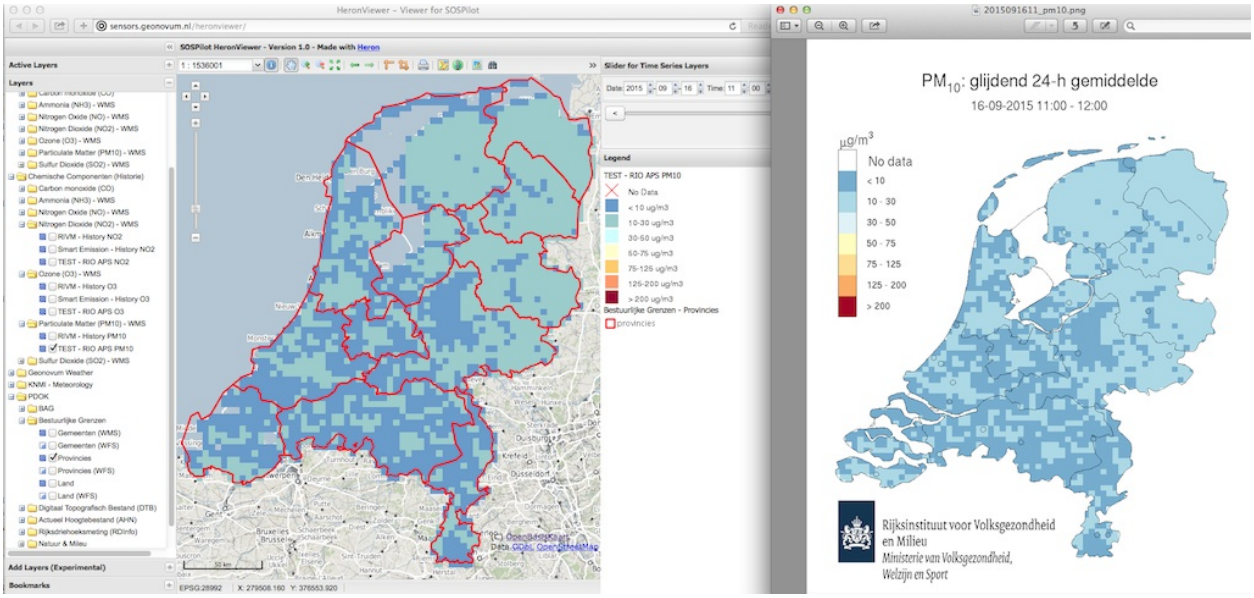


Fig. 7.4: *Comparing with RIVM LML maps (right) - PM10*

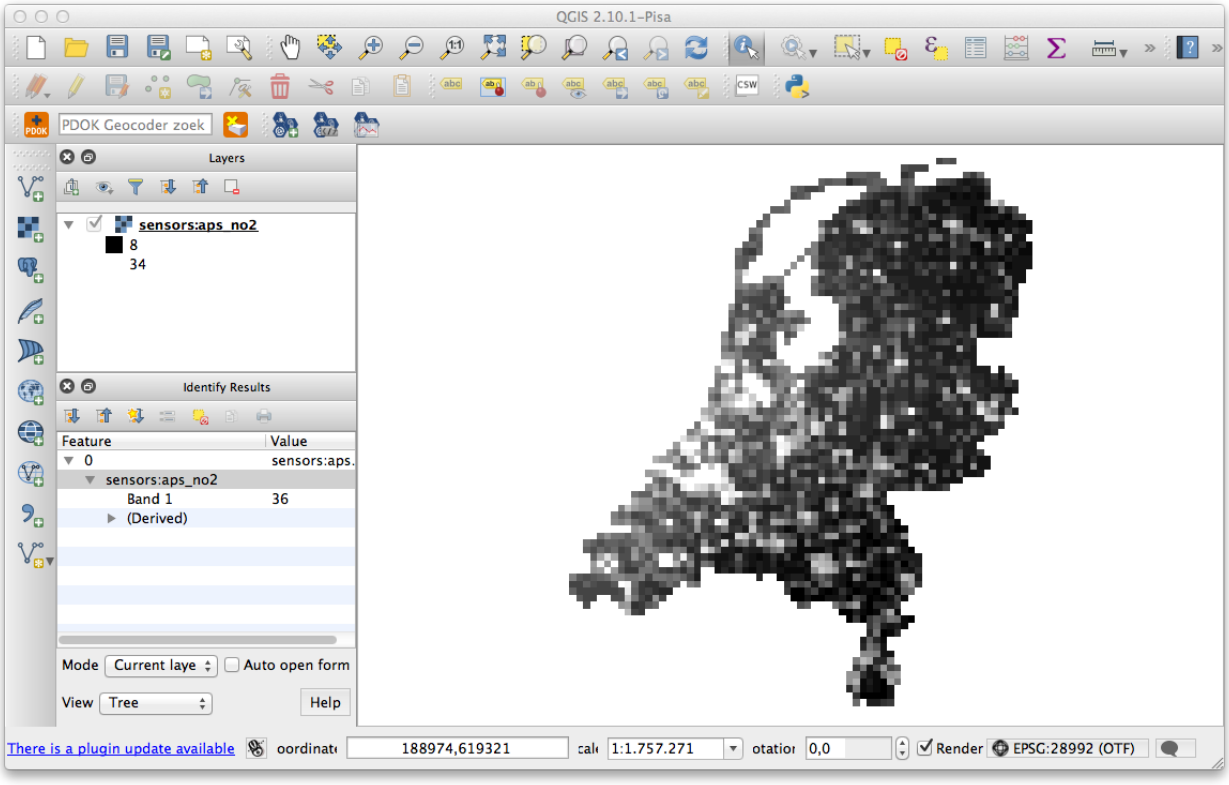


Fig. 7.5: *INSPIRE View and Download Services*

7.3 Viewing Results

Results can be viewed in basically 3 four ways:

- as WMS Layers via the Heron Viewer: http://sensors.geonovum.nl/heronviewer
- as WCS in e.q. QGIS http://sensors.geonovum.nl/gs/sensors/wcs?
- via direct WCS protocol requests
- as raw GeoTIFF raster files

Below some guidance for each viewing method. TBS

7.3.1 Heron Viewer

Easiest is to use the predefined map-context bookmarks (see also the tab "Bookmarks" on the left panel):

- NO2 coverage: http://sensors.geonovum.nl/heronviewer/?bookmark=rivmriono2
- O3 coverage: http://sensors.geonovum.nl/heronviewer/?bookmark=rivmriono3
- PM10 coverage: http://sensors.geonovum.nl/heronviewer/?bookmark=rivmriopm10

Or by hand: go to http://sensors.geonovum.nl/heronviewer

- left are all map-layers organized as a collapsible explorer folders
- find the folder named "Chemische Componenten (Historie)"
- open this folder
- open subfolder named "Nitrogen Dioxide (NO2) - WMS"
- activate the map-layer named "TEST RIO APS NO2"

These values can be compared with the WMS-Time Layer-based values for the same component, for example for NO2:

- within the same submap "Nitrogen Dioxide (NO2) - WMS"
- enable the WMS-Time Layer "RIVM History - NO2"
- use the Timeslider in the upper right of the viewer
- slide the date/time to September 16, 2015 11:00
- compare the values by clicking on the circles, a pop-up opens for each click
- you may first want to disable the stations layers

7.3.2 WCS in QGIS

See above. WCS link is http://sensors.geonovum.nl/gs/sensors/wcs?. There are 3 layers.

7.3.3 WCS Protocol Requests

Direct requests to GeoServer, using the WCS 2.0.1 protocol:

- Fetch WCS Capabilities 2.0.1
- Fetch WCS DescribeCoverage 2.0.1

- Fetch WCS GetCoverage, 41kB GeoTIFF
- Fetch WCS GetCoverage, plain text

7.3.4 Raw Coverage Files

Get GeoTIFF Files from raw GeoTIFF raster files. or via WCS GetCoverage, 41kB GeoTIFF.

7.4 Future Enhancements

7.4.1 WCS and WMS as time-series

Traditional WMS-Time layers are based on vector-data, where a single column in the source data denotes time (or any other dimension like elevation). GeoServer also supports time-series for raster data.

Each coverage now is a snapshot for a single chemical component for a single hour on s single date. To unlock multiple coverages as a time-series the GeoServer ImageMosaic with time-series support can be applied, see this link.

Another posibility is a GeoServer plugin for "Earth Observation profile of coverage standard", see http://docs.geoserver.org/latest/en/user/extensions/wcs20eo/index.html and http://www.opengeospatial.org/standards/requests/81

CHAPTER 8

Weather Data

This chapter describes how the SOSPilot platform as developed initially for Air Quality data can be reused to incorporate (historical) weather data. This has been performed using Open Data from the Dutch Meteorological Institute (KNMI). The nature of the data and the required steps are very similar to Air Quality data:

- obtaining raw source data: both time-series measurements and (georeferenced) stations
- data transformation (ETL)
- providing OWS services: WMS-(Time), WFS and SOS
- visualizing via Clients

See also https://github.com/Geonovum/sospilot/issues/17

8.1 Source Data

Via http://data.knmi.nl many Open Data sets are available. For our purpose we need:

- georeferenced station data
- (historical) measurements data

8.1.1 Stations

The station data can be found via https://data.knmi.nl/portal/KNMI-DataCentre.html, the dataset name is `waarneemstations`. Within a .tar file the file `STD___OPER_P___OBS_____L2.nc` is present. The format is tricky: a NetCDF (Network Common Data Form) file (.nc extension).

To get the file:

- get metadata XML from data info popup: https://data.knmi.nl/webservices/metadata/getDatasetMetadataXML?datasetName=waarneemstations&datasetVersion=1
- extract Atom URL from metadata XML: http://data.knmi.nl/inspire/atom?dataset=urn:xkdc:ds:nl.knmi::waarneemstations/1/

- extract download URL from Atom: http://data.knmi.nl/inspire/download/waarneemstations/1/noversion/0000/00/00/STD___OPER_P___OBS_____L2.nc.zip
- unzip to `STD___OPER_P___OBS_____L2.nc`

See ETL below how to further handle this data.

8.1.2 Measurements

Historical data per station per decade can be found at http://www.knmi.nl/klimatologie/uurgegevens. The .zip files contain a .CSV file with all hourly measurements for a station, e.g. the file `uurgeg_275_2011-2020.txt` contains measurements for station 275 (Deelen) between 2011 and 2020, in practice all measurements between 2011 and the day before today until 24:00. URLs remain constant and are directly accessible, for example for Deelen (station 275): http://www.knmi.nl/klimatologie/uurgegevens/datafiles/275/uurgeg_275_2011-2020.zip etc.

Example contents of `uurgeg_275_2011-2020.txt` on june 9, 2014.

```
BRON:
KONINKLIJK NEDERLANDS METEOROLOGISCH INSTITUUT (KNMI)

SOURCE:
ROYAL NETHERLANDS METEOROLOGICAL INSTITUTE (KNMI)

YYYYMMDD  = datum (YYYY=jaar,MM=maand,DD=dag) / date (YYYY=year,MM=month,DD=day)
HH        = tijd (HH=uur, UT.12 UT=13 MET, 14 MEZT. Uurvak 05 loopt van 04.00 UT tot␣
↪5.00 UT / time (HH uur/hour, UT. 12
UT=13 MET, 14 MEZT. Hourly division 05 runs from 04.00 UT to 5.00 UT
DD        = Windrichting (in graden) gemiddeld over de laatste 10 minuten van het␣
↪afgelopen uur (360=noord, 90=oost,
180=zuid, 270=west, 0=windstil 990=veranderlijk. Zie http://www.knmi.nl/klimatologie/
↪achtergrondinformatie/windroos.pdf /
 Mean wind direction (in degrees) during the 10-minute period preceding the time of␣
↪observation (360=north, 90=east,
 180=south, 270=west, 0=calm 990=variable)
FH        = Uurgemiddelde windsnelheid (in 0.1 m/s). Zie http://www.knmi.nl/
↪klimatologie/achtergrondinformatie/beaufortschaal.pdf
/ Hourly mean wind speed (in 0.1 m/s)
FF        = Windsnelheid (in 0.1 m/s) gemiddeld over de laatste 10 minuten van het␣
↪afgelopen uur / Mean wind speed
(in 0.1 m/s) during the 10-minute period preceding the time of observation
FX        = Hoogste windstoot (in 0.1 m/s) over het afgelopen uurvak / Maximum wind␣
↪gust (in 0.1 m/s) during the
hourly division
T         = Temperatuur (in 0.1 graden Celsius) op 1.50 m hoogte tijdens de␣
↪waarneming / Temperature (in 0.1 degrees
Celsius) at 1.50 m at the time of observation
T10N      = Minimumtemperatuur (in 0.1 graden Celsius) op 10 cm hoogte in de␣
↪afgelopen 6 uur / Minimum temperature
(in 0.1 degrees Celsius) at 0.1 m in the preceding 6-hour period
TD        = Dauwpuntstemperatuur (in 0.1 graden Celsius) op 1.50 m hoogte tijdens de␣
↪waarneming / Dew point temperature
 (in 0.1 degrees Celsius) at 1.50 m at the time of observation
SQ        = Duur van de zonneschijn (in 0.1 uren) per uurvak, berekend uit globale␣
↪straling  (-1 for <0.05 uur)
/ Sunshine duration (in 0.1 hour) during the hourly division, calculated from global␣
↪radiation (-1 for <0.05 hour)
Q         = Globale straling (in J/cm2) per uurvak / Global radiation (in J/cm2)␣
↪during the hourly division
```

```
DR         = Duur van de neerslag (in 0.1 uur) per uurvak / Precipitation duration (in␣
↪0.1 hour) during the hourly division
RH         = Uursom van de neerslag (in 0.1 mm) (-1 voor <0.05 mm) / Hourly␣
↪precipitation amount (in 0.1 mm)
(-1 for <0.05 mm)
P          = Luchtdruk (in 0.1 hPa) herleid naar zeeniveau, tijdens de waarneming /␣
↪Air pressure (in 0.1 hPa)
reduced to mean sea level, at the time of observation
VV         = Horizontaal zicht tijdens de waarneming (0=minder dan 100m, 1=100-200m,␣
↪2=200-300m,..., 49=4900-5000m,
50=5-6km, 56=6-7km, 57=7-8km, ..., 79=29-30km, 80=30-35km, 81=35-40km,..., 89=meer␣
↪dan 70km) / Horizontal visibility
at the time of observation (0=less than 100m, 1=100-200m, 2=200-300m,..., 49=4900-
↪5000m, 50=5-6km, 56=6-7km, 57=7-8km,
..., 79=29-30km, 80=30-35km, 81=35-40km,..., 89=more than 70km)
N          = Bewolking (bedekkingsgraad van de bovenlucht in achtsten), tijdens de␣
↪waarneming (9=bovenlucht onzichtbaar)
 / Cloud cover (in octants), at the time of observation (9=sky invisible)
U          = Relatieve vochtigheid (in procenten) op 1.50 m hoogte tijdens de␣
↪waarneming / Relative atmospheric humidity
 (in percents) at 1.50 m at the time of observation
WW         = Weercode (00-99), visueel(WW) of automatisch(WaWa) waargenomen, voor het␣
↪actuele weer of het weer in het
afgelopen uur. Zie http://www.knmi.nl/klimatologie/achtergrondinformatie/ww_lijst_
↪nederlands.pdf / Present weather code
 (00-99), description for the hourly division. See http://www.knmi.nl/klimatologie/
↪achtergrondinformatie/ww_lijst_engels.pdf
IX         = Weercode indicator voor de wijze van waarnemen op een bemand of␣
↪automatisch station (1=bemand gebruikmakend
 van code uit visuele waarnemingen, 2,3=bemand en weggelaten (geen belangrijk␣
↪weersverschijnsel, geen gegevens), 4=automatisch
 en opgenomen (gebruikmakend van code uit visuele waarnemingen), 5,6=automatisch en␣
↪weggelaten (geen belangrijk
 weersverschijnsel, geen gegevens), 7=automatisch gebruikmakend van code uit␣
↪automatische waarnemingen) / Indicator present
 weather code (1=manned and recorded (using code from visual observations), 2,
↪3=manned and omitted (no significant weather
 phenomenon to report, not available), 4=automatically recorded (using code from␣
↪visual observations), 5,6=automatically
 omitted (no significant weather phenomenon to report, not available),␣
↪7=automatically set (using code from automated observations)
M          = Mist 0=niet voorgekomen, 1=wel voorgekomen in het voorgaande uur en/of␣
↪tijdens de waarneming /
Fog 0=no occurrence, 1=occurred during the preceding hour and/or at the time of␣
↪observation
R          = Regen 0=niet voorgekomen, 1=wel voorgekomen in het voorgaande uur en/of␣
↪tijdens de waarneming /
Rainfall 0=no occurrence, 1=occurred during the preceding hour and/or at the time of␣
↪observation
S          = Sneeuw 0=niet voorgekomen, 1=wel voorgekomen in het voorgaande uur en/of␣
↪tijdens de waarneming /
Snow 0=no occurrence, 1=occurred during the preceding hour and/or at the time of␣
↪observation
O          = Onweer 0=niet voorgekomen, 1=wel voorgekomen in het voorgaande uur en/of␣
↪tijdens de waarneming /
Thunder  0=no occurrence, 1=occurred during the preceding hour and/or at the time of␣
↪observation
Y          = IJsvorming 0=niet voorgekomen, 1=wel voorgekomen in het voorgaande uur en/
↪of tijdens de waarneming /
```

```
Ice formation 0=no occurrence, 1=occurred during the preceding hour and/or at the␣
↪time of observation

# STN,YYYYMMDD,   HH,   DD,   FH,   FF,   FX,    T,  T10,   TD,   SQ,    Q,   DR,␣
↪RH,    P,   VV,    N,    U,   WW,   IX,    M,    R,    S,    O,    Y

  275,20110101,    1,  240,   30,   30,   40,   15,     ,   14,    0,    0,    0,␣
↪0,10217,    1,    1,   99,   34,    7,    1,    0,    0,    0,    0
  275,20110101,    2,  250,   30,   40,   60,   18,     ,   17,    0,    0,    0,␣
↪0,10213,    2,    9,   99,   32,    7,    1,    0,    0,    0,    0
  275,20110101,    3,  250,   40,   40,   70,   21,     ,   20,    0,    0,    0,␣
↪0,10210,    2,    5,   99,   33,    7,    1,    0,    0,    0,    0
   .
   .
  275,20140708,   20,  330,   30,   30,   70,  131,     ,  125,    0,    2,    9,␣
↪2,10119,   65,    8,   96,   57,    7,    0,    1,    0,    0,    0
  275,20140708,   21,  300,   30,   20,   50,  128,     ,  120,    0,    0,    8,␣
↪1,10118,   65,    8,   95,   57,    7,    0,    1,    0,    0,    0
  275,20140708,   22,  300,   30,   30,   50,  128,     ,  120,    0,    0,    0,␣
↪1,10116,   65,    8,   95,   81,    7,    0,    1,    0,    0,    0
  275,20140708,   23,  270,   20,   20,   40,  126,     ,  123,    0,    0,    0,␣
↪1,10115,   59,    8,   98,   61,    7,    0,    1,    0,    0,    0
  275,20140708,   24,  270,   20,   20,   40,  127,  125,  122,    0,    0,    0,␣
↪1,10110,   61,    8,   97,   23,    7,    0,    1,    0,    0,    0
```

"Live" 10-minute data can be found via https://data.knmi.nl/portal/KNMI-DataCentre.html, but again in the NetCDF format. Though downloading is forced via an HTML page with attachment header popup.

Though direct download is possible via the following steps.

- Download the MetaData URL https://data.knmi.nl/webservices/metadata/getDatasetMetadataXML?datasetName=Actuele10mindataKNMIstations&datasetVersion=1
- In the XML an Atom Feed URL for download is found: http://data.knmi.nl/inspire/atom?dataset=urn:xkdc:ds:nl.knmi::Actuele10mindataKNMIstations/1
- The Atom Feed contains a download URL of the form: http://data.knmi.nl/inspire/download/Actuele10mindataKNMIstations/1/noversion/2014/07/09/KMDS__OPER_P___10M_OBS_L2.nc.zip

This pattern `2014/07/09/KMDS__OPER_P___10M_OBS_L2.nc.zip` looks like we could download any date but in reality the current last 10 minutes are always downloaded. The contents of the contained NetCDF file are very similar to the Stations .nc above.

```
$ ncdump KMDS__OPER_P___10M_OBS_L2.nc

netcdf KMDS__OPER_P___10M_OBS_L2 {
dimensions:
    station = 47 ;
    time = 1 ;
variables:
    string station(station) ;
        station:long_name = "Station id" ;
        station:cf_role = "timeseries_id" ;
    double time(time) ;
        time:long_name = "time of measurement" ;
        time:standard_name = "time" ;
        time:units = "seconds since 1950-01-01 00:00:00" ;
    string stationname(station) ;
        stationname:long_name = "Station name" ;
```

```
    double lat(station) ;
        lat:long_name = "station  latitude" ;
        lat:standard_name = "latitude" ;
        lat:units = "degrees_north" ;
    double lon(station) ;
        lon:long_name = "station longitude" ;
        lon:standard_name = "longitude" ;
        lon:units = "degrees_east" ;
    double height(station) ;
        height:long_name = "Station height" ;
        height:standard_name = "height" ;
        height:units = "m" ;
    double dd(station) ;
        dd:_FillValue = -9999. ;
        dd:standard_name = "wind_from_direction" ;
        dd:units = "degree" ;
        dd:long_name = "Wind Direction 10 Min Average" ;
    double ff(station) ;
        ff:_FillValue = -9999. ;
        ff:standard_name = "wind_speed" ;
        ff:units = "m s-1" ;
        ff:long_name = "Wind Speed at 10m 10 Min Average" ;
    double gff(station) ;
        gff:_FillValue = -9999. ;
        gff:standard_name = "wind_speed_of_gust" ;
        gff:units = "m s-1" ;
        gff:long_name = "Wind Gust at 10m 10 Min Maximum" ;
    double ta(station) ;
        ta:_FillValue = -9999. ;
        ta:standard_name = "air_temperature" ;
        ta:units = "degrees Celsius" ;
        ta:long_name = "Air Temperature 1.5m 10 Min Average" ;
    double rh(station) ;
        rh:_FillValue = -9999. ;
        rh:standard_name = "relative_humidity" ;
        rh:units = "%" ;
        rh:long_name = "Relative Humidity 1.5m 1 Min Average" ;
    double pp(station) ;
        pp:_FillValue = -9999. ;
        pp:standard_name = "air_pressure_at_sea_level" ;
        pp:units = "hPa" ;
        pp:long_name = "Air Pressure at Sea Level 1 Min Average" ;
    double zm(station) ;
        zm:_FillValue = -9999. ;
        zm:standard_name = "visibility_in_air" ;
        zm:units = "m" ;
        zm:long_name = "Meteorological Optical Range 10 Min Average" ;
    char iso_dataset ;
        iso_dataset:title = "KMDS__OPER_P___10M_OBS_L2" ;
        iso_dataset:abstract = "Most recent 10 minute in situ observations of the␣
↪Dutch meteorological observation network" ;
        iso_dataset:status = "ongoing" ;
        iso_dataset:type = "dataset" ;
        iso_dataset:uid = "c3a312e2-2d8f-440b-ae7d-3406c9fe2f77" ;
        iso_dataset:topic = "atmosphere" ;
        iso_dataset:keyword = "tbd" ;
        iso_dataset:max-x = 10.f ;
        iso_dataset:min-x = 0.f ;
```

```
        iso_dataset:max-y = 60.f ;
        iso_dataset:min-y = 40.f ;
        iso_dataset:temporal_extent = "1950-01-01 and ongoing" ;
        iso_dataset:date = "2013-10-10" ;
        iso_dataset:dateType = "publication date" ;
        iso_dataset:statement = "Most recent 10 minute in situ observations in situ␣
↪observations of the Dutch meteorological observation network" ;
        iso_dataset:code = "TBD" ;
        iso_dataset:codeSpace = "EPSG" ;
        iso_dataset:accessConstraints = "none" ;
        iso_dataset:useLimitation = "none" ;
        iso_dataset:organisationName_dataset = "Royal Netherlands Meteorological␣
↪Institute (KNMI)" ;
        iso_dataset:email_dataset = "data@knmi.nl" ;
        iso_dataset:role_dataset = "pointOfContact" ;
        iso_dataset:metadata_id = "fbfad5b9-1dd2-425e-bb35-c96386380c0e" ;
        iso_dataset:organisationName_metadata = "Royal Netherlands Meteorological␣
↪Institute (KNMI)" ;
        iso_dataset:role_metadata = "pointOfContact" ;
        iso_dataset:email_metadata = "data@knmi.nl" ;
        iso_dataset:url_metadata = "http://data.knmi.nl" ;
        iso_dataset:datestamp = "2010-11-01" ;
        iso_dataset:language = "eng" ;
        iso_dataset:metadataStandardName = "ISO 19115" ;
        iso_dataset:metadataStandardNameVersion = "Nederlandse metadatastandaard op␣
↪ISO 19115 voor geografie 1.2" ;
    char product ;
        product:units = "1" ;
        product:long_name = "ADAGUC Data Products Standard" ;
        product:ref_doc = "ADAGUC Data Products Standard" ;
        product:ref_doc_version = "1.1" ;
        product:format_version = "1.1" ;
        product:originator = "Royal Netherlands Meteorological Institute (KNMI)" ;
        product:type = "P" ;
        product:acronym = "KMDS__OPER_P___10M_OBS_L2" ;
        product:level = "L2" ;
        product:style = "camelCase" ;
    char projection ;
        projection:EPSG_code = "EPSG:4326" ;

// global attributes:
        :featureType = "timeSeries" ;
        :Conventions = "CF-1.4" ;
        :title = "KMDS__OPER_P___10M_OBS_L2" ;
        :institution = "Royal Netherlands Meteorological Institute (KNMI)" ;
        :source = "Royal Netherlands Meteorological Institute (KNMI)" ;
        :history = "File created from KMDS ASCII file. " ;
        :references = "http://data.knmi.nl" ;
        :comment = "none" ;
data:

 station = "06201", "06203", "06204", "06205", "06206", "06207", "06208",
    "06210", "06211", "06212", "06225", "06235", "06239", "06240", "06242",
    "06248", "06249", "06251", "06257", "06258", "06260", "06267", "06269",
    "06270", "06273", "06275", "06277", "06278", "06279", "06280", "06283",
    "06286", "06290", "06310", "06319", "06330", "06340", "06343", "06344",
    "06348", "06350", "06356", "06370", "06375", "06377", "06380", "06391" ;
```

```
time = 2036070000 ;

stationname = "D15-FA-1", "P11-B", "K14-FA-1C", "A12-CPP", "F16-A",
   "L9-FF-1", "AWG-1", "VALKENBURG AWS", "J6-A", "HOORN-A", "IJMUIDEN WP",
   "DE KOOIJ VK", "F3-FB-1", "AMSTERDAM/SCHIPHOL AP", "VLIELAND",
   "WIJDENES WP", "BERKHOUT AWS", "TERSCHELLING HOORN AWS",
   "WIJK AAN ZEE AWS", "HOUTRIBDIJK WP", "DE BILT AWS", "STAVOREN AWS",
   "LELYSTAD AP", "LEEUWARDEN", "MARKNESSE AWS", "DEELEN", "LAUWERSOOG AWS",
   "HEINO AWS", "HOOGEVEEN AWS", "GRONINGEN AP EELDE", "HUPSEL AWS",
   "NIEUW BEERTA AWS", "TWENTE AWS", "VLISSINGEN AWS", "WESTDORPE AWS",
   "HOEK VAN HOLLAND AWS", "WOENSDRECHT", "ROTTERDAM GEULHAVEN",
   "ROTTERDAM THE HAGUE AP", "CABAUW TOWER AWS", "GILZE RIJEN",
   "HERWIJNEN AWS", "EINDHOVEN AP", "VOLKEL", "ELL AWS",
   "MAASTRICHT AACHEN AP", "ARCEN AWS" ;

lat = 54.325666666667, 52.36, 53.269444444444, 55.399166666667,
   54.116666666667, 53.614444444444, 53.491666666667, 52.170248689194,
   53.824130555556, 52.918055555556, 52.462242867998, 52.926865008825,
   54.853888888889, 52.315408447486, 53.240026656696, 52.632430667762,
   52.642696895243, 53.391265948394, 52.505333893732, 52.648187308904,
   52.098821802977, 52.896643913235, 52.457270486008, 53.223000488316,
   52.701902388132, 52.0548617826, 53.411581103636, 52.434561756559,
   52.749056395511, 53.123676213651, 52.067534268959, 53.194409573306,
   52.27314817052, 51.441334059998, 51.224757511326, 51.990941918858,
   51.447744494043, 51.891830906739, 51.960667359998, 51.969031121385,
   51.564889021961, 51.857593837453, 51.449772459909, 51.658528382201,
   51.196699902606, 50.905256257898, 51.497306260089 ;

lon = 2.93575, 3.3416666666667, 3.6277777777778, 3.8102777777778,
   4.0122222222222, 4.9602777777778, 5.9416666666667, 4.4294613573587,
   2.9452777777778, 4.1502777777778, 4.5549006792363, 4.7811453228565,
   4.6961111111111, 4.7902228464686, 4.9207907082729, 5.1734739738872,
   4.9787572406902, 5.3458010937365, 4.6029300588208, 5.4003881262577,
   5.1797058644882, 5.383478899702, 5.5196324030324, 5.7515738887123,
   5.8874461671401, 5.8723225499118, 6.1990994508938, 6.2589770334531,
   6.5729701105864, 6.5848470019087, 6.6567253619722, 7.1493220605216,
   6.8908745111116, 3.5958241584686, 3.8609657214986, 4.121849767852,
   4.342014, 4.3126638323991, 4.4469005114756, 4.9259216999194,
   4.9352386335384, 5.1453989235756, 5.3770039280214, 5.7065946674719,
   5.7625447234516, 5.7617834850481, 6.1961067840608 ;

height = 42.7, 41.84, 41.8, 48.35, 43.4, 44, 40.5, -0.2, 45.7, 50.9, 4,
   1.22, 50.6, -3.35, 10.79, 0.8, -2.4, 0.73, 8.5, 7.25, 1.9, -1.3, -3.66,
   1.22, -3.35, 48.16, 2.9, 3.6, 15.82, 5.18, 29.07, -0.2, 34.75, 8.03,
   1.68, 11.86, 19.2, 3.5, -4.27, -0.71, 14.94, 0.66, 22.56, 21.95, 30,
   114.3, 19.5 ;

dd = 343.2, 320.3, 0, 355.8, 324.4, 339.8, 340.7, 337.7, 312.1, 331.9,
   335.3, 330.9, 20.7, 336.6, 337, 326.4, 331.4, 338.6, _, 333.8, 338.7,
   337.3, 333.2, 350.5, 331.2, 327.5, 352.6, 347.5, 323.4, 352, 325.3, 0.1,
   326.5, 328.4, 328.1, 331.9, 328.8, 330.4, 333.4, 331.1, 336.4, 326.3,
   337.1, 335.6, 327.8, 271.5, 331.3 ;

ff = 14.40063, 12.71024, 0, 7.480422, 10.61489, 9.930736, 7.767642, 12.29,
   14.5523, 14.31887, 15.59, 7.77, 7.253119, 8.39, 12.31, 9.03, 8.82, 7.23,
   _, 12.58, 4.92, 10.97, 6.88, 7.11, 4.99, 6.1, 8.76, 2.68, 4.29, 5.05,
   2.43, 5.38, 3.07, 12.74, 10.74, 20.89, 7.5, 10.56, 6.14, 8.7, 8.39, 6.64,
   7.09, 4.16, 5.21, 3.03, 2.88 ;
```

```
 gff = 17.47643, 19.11437, 0, 8.184164, 12.64459, 11.1621, 9.07996, 16.54,
    17.28121, 16.76123, 18.21, 13.3, 8.400682, 12.29, 14.35, 12.1, 11.92,
    9.92, _, 14.85, 8.88, 13.42, 10.37, 9.32, 7.92, 8.99, 10.99, 5.07, 6.86,
    7.26, 3.21, 7.75, 4.97, 19.26, 17.94, 23.88, 13.05, 14.12, 11.69, 11.75,
    12.62, 10.78, 12.21, 7.4, 8.72, 5.03, 5.5 ;

 ta = 15, 15.6, 15.5, 16, 15.9, 16.9, 18.2, 16, 14.9, 15.6, _, 16.7, 17.2,
    16.3, 17.3, _, 16.6, 18.9, 16.2, _, 16.4, 18, 17.5, 20.9, 19.7, 17.2,
    21.3, 20.3, 22.6, 24.4, 19, 26, 20.7, 15.4, 14.9, 15.5, 15.2, _, 15.7,
    16.1, 15.8, 16.1, 16, 16.6, 15.9, 14.7, 17.2 ;

 rh = 98, 87, 95, 97, 98, 96, 95, 95, 94, 95, _, 97, 92, 97, 95, _, 98, 91,
    96, _, 97, 97, 97, 87, 93, 99, 82, 92, 88, 77, 97, 76, 90, 92, 94, 95,
    94, _, 94, 96, 96, 95, 96, 97, 96, 99, 96 ;

 pp = 1011.647, 1010.537, 1010.917, 1011.244, 1010.213, 1007.996, 1006.526,
    1008.164, 1011.488, 0, _, 1007.934, 1009.975, 1007.509, 1007.941, _, _,
    1007.273, _, _, 1007.118, _, 1006.237, 1006.516, _, 1005.847, _, _,
    1005.052, 1005.169, _, _, 1004.474, 1010.052, 1010.173, 1008.988,
    1008.784, _, 1008.559, 1007.503, 1007.409, 1007.006, 1006.774, 1006.152,
    _, 1006.434, _ ;

 zm = 3030, 10600, 7760, 6690, 3860, 5750, 6360, 8980, 7200, 4390, _, 5060,
    11400, 6820, 2800, _, 4980, 6270, _, _, 6410, 4350, 5030, 12200, 6690,
    1740, _, _, 8900, 17500, _, _, 5490, 5870, 6860, _, 7520, _, 6770, 7270,
    9540, _, 9970, 7270, 8940, 3430, _ ;

 iso_dataset = "" ;

 product = "" ;

 projection = "" ;
}
```

There is also a simpler table: ftp://ftp.knmi.nl/pub_weerberichten/tabel_10min_data.html but no historical data can be fetched.

The strategy for ETL could be to:

- use the historical for initial DB fill
- incrementally add, each hour, from the 10-minute data.

8.2 ETL Implementation

In this section the ETL is elaborated.

As with AQ the ETL is performed in these steps:

1. Incremental download of source data and preparation
2. Raw data for stations and measurements to Postgres tables, “Core Weather Data”
3. SOS ETL: transform and publish “Core Weather Data” to the 52N SOS DB via SOS-Transactions (SOS-T)

8.2.1 Stations ETL

Needs to be done once. Open source tools like ncdump and GDAL (http://www.gdal.org/frmt_netcdf.html) exist to extract data from the NetCDF file. We'll use `ncdump`. Install on Linux via `apt-get install netcdf-bin`, on Mac OSX via `brew install netcdf`.

The ETL is done in the following steps (see Git dir `/data/knmi/stations`):

1. download `wget http://data.knmi.nl/inspire/download/waarneemstations/1/noversion/0000/00/00/STD___OPER_P___OBS_____L2.nc.zip`
2. unzip to `STD___OPER_P___OBS_____L2.nc`
3. create a text-readable Net CDF dump.

```
ncdump STD___OPER_P___OBS_____L2.nc > stations.ncdump.txt
```

4. create CSV from `stations.ncdump.txt` using custom Python program.

```
./stations2csv.py > stations.csv
```

5. use `ogr2ogr` commandline with `stations.vrt` for DB mappings to read into PostGIS

8.2.2 ETL Step 1. - Harvester

TO BE SUPPLIED

8.2.3 ETL Step 2 - Raw Measurements

This step produces raw weather measurements.

Two tables: `stations` and `measurements`. This is a 1:1 transformation from the raw weather data harvested in Step 1. The `measurements` table refers to the `stations` by a FK `station_code` (WMO in source data). The table `etl_progress` is used to track the last file processed from `lml_files`.

Measurements

TO BE SUPPLIED

8.2.4 ETL Step 3 - SOS Publication

In this step the Raw Weather data (see Step 2) is transformed to "SOS Ready Data", i.e. data that can be handled by the 52North SOS server. "SOS Transactions" (SOS-T) have been proven to work well for the LML AQ data. So from here on publication via SOS-T is further expanded.

SOS Publication - Stetl Strategy

Similar to SOS-T for AQ.

SOS Publication - Sensors

This step needs to be performed only once, or when any of the original Station data changes.

The Stetl config https://github.com/Geonovum/sospilot/blob/master/src/knmi/stations2sensors.cfg uses a Standard Stetl module, `inputs.dbinput.PostgresDbInput` for obtaining Record data from a Postgres database.

```
{{
  "request": "InsertSensor",
  "service": "SOS",
  "version": "2.0.0",
  "procedureDescriptionFormat": "http://www.opengis.net/sensorML/1.0.1",
  "procedureDescription": "{procedure-desc.xml}",
   "observableProperty": [
     (need observable properties: Temperature, Wind direction etc)
  ],
  "observationType": [
    "http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement"
  ],
  "featureOfInterestType": "http://www.opengis.net/def/samplingFeatureType/OGC-OM/2.0/
↪SF_SamplingPoint"
}}
```

The SOSTOutput module will expand `{procedure-desc.xml}` with the Sensor ML template.

SOS Publication - Observations

The Stetl config

The Observation template looks as follows.

```
{{
   "request": "InsertObservation",
   "service": "SOS",
   "version": "2.0.0",
   "offering": "http://sensors.geonovum.nl/knmi/offering/{station_code}",
   "observation": {{
     "identifier": {{
       "value": "{unique_id}",
       "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown"
     }},
     "type": "http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement",
     "procedure": "http://sensors.geonovum.nl/knmi/procedure/{station_code}",
     "observedProperty": "http://sensors.geonovum.nl/knmi/observableProperty/
↪{observation_type}",
     "featureOfInterest": {{
       "identifier": {{
         "value": "http://sensors.geonovum.nl/knmi/featureOfInterest/{station_code}",
         "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown"
       }},
       "name": [
         {{
           "value": "{municipality}",
           "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown"
         }}
       ],
       "geometry": {{
         "type": "Point",
```

```
          "coordinates": [
            {station_lat},
            {station_lon}
          ],
          "crs": {{
            "type": "name",
            "properties": {{
              "name": "EPSG:4326"
            }}
          }}
        }}
      }},
      "phenomenonTime": "{sample_time}",
      "resultTime": "{sample_time}",
      "result": {{
        "uom": "deegrees",
        "value": {sample_value}
      }}
    }}
}}
```

It is quite trivial in `sosoutput.py` to substitute these values from the `measurements`-table records.

Like in ETL Step 2 the progress is remembered in the table `rivm_lml.etl_progress` by updating the `last_id` field after publication, where that value represents the `gid` value of `rivm_lml.measurements`.

SOS Publication - Results

TO BE SUPPLIED Via the standard SOS protocol the results can be tested:

- GetCapabilities: http://sensors.geonovum.nl/sos/service?service=SOS&request=GetCapabilities
- DescribeSensor (station 275, Deelen): to be supplied
- GetObservation: to be supplied

CHAPTER 9

Weather Station

This chapter describes the hard/software setup of a weather station whose measurements are to be exposed via OGC protocols like WMS, WFS and in particular SOS.

9.1 Introduction

In 2008/2009 Geonovum acquired a Davis Vantage Pro II Weather station. This station was/is mounted on the roof of Geonovum building. The outside station-sensors measure barometric pressure, temperature, humidity, rainfall, wind speed and direction. Data is continuously sent (wireless, via RF) to a base station, a.k.a. the "console". The console has a Davis WeatherLink® hard/software add-on that is able to act as a data-logger and provides connectivity over USB.

Data from the weather station was extracted and published to the 52N SOS via a special purpose PC application. The results of this project have been described in a Geo-Info magazine article (2010-1).

As part of the SOSPilot project, we will now "revive" the Davis Weather station and connect its data stream to the SOSPilot infrastructure simliar to the RIVM AQ datastream. Eventually, the goal is to expose the station weather data to OGC services like WMS, WFS and in particular SOS.

As we cannot connect the weather base station (USB), directly to the SOSPilot server "in the cloud", an intermediate "middleman" hard/software component will be required. The main functions of this component are to acquire data from the base station and to transform and publish this data to the SOSPilot server (VPS).

To this end a Raspberry Pi (RPi) microcomputer has been chosen. The RPi is a cheap (around $25,-) credit-card-sized microcomputer that can run free OS-es like Linux (Raspbian, a Debian version) and allows connectivity to USB, WIFI, GSM etc. The RPi is popular with weather amateurs (and various other IoT projects as well).

9.2 Overview

The overall architecture is depicted below.

These components and their interconnections operate as follows.

Fig. 9.1: *Figure 0 - Geonovum Davis VP2 Weather Station and Console connected to RPi*

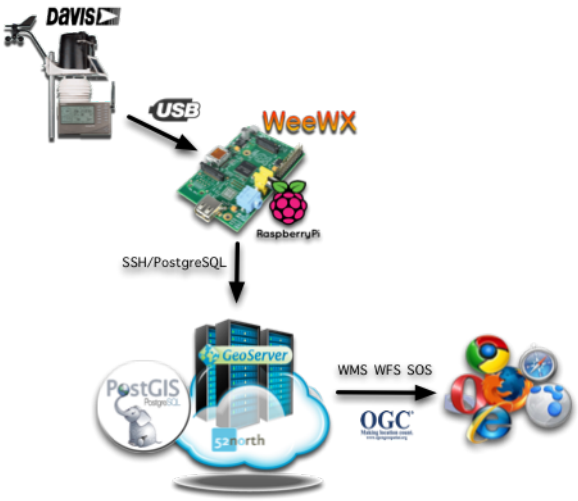


Fig. 9.2: *Figure 1 - Global Setup from Weather Station Through OGC Services*

The Davis weather station is connected to a Raspberry Pi (RPi) micro computer via USB trough the Davis WeatherLink®.

The RPi runs the Raspbian OS with the weather data software weewx. The weewx daemon continuously reads raw weather sample data from the Davis weather station and stores this data locally in a (SQLite or MySQL) database. These measurements, called weewx *Archive Records* are typically a 5-minute summary of multiple samples (weewx *Loop Records*) collected every few seconds.

An ETL process based on Stetl transforms and syncs data from the SQLite *weather archive database* to a remote PostgreSQL server in a "Cloud Server" (the Ubuntu VPS used in the project).

Other weewx plugins generate reports with statistics and summaries. These are synced regularly to be viewed as webpages in a browser.

The VPS runs GeoServer to serve the weather data directly from the Postgres/PostGIS database, using specialized PostgreSQL VIEWs, as WMS, WMS-Time and WFS.

In addition the VPS runs a Stetl ETL process that transforms and and publishes the weather data from PostgreSQL using the SOS-T protocol to the 52North Sensor Web Application server. The latter provides a SOS (Sensor Observation Service). Via the web browser various WMS/WFS and SOS client applications are invoked.

All client applications can be found and accessed via the project landing page: sensors.geonovum.nl:

- weather reports via: sensors.geonovum.nl/weewx
- WMS/WMS-Time/WFS via: sensors.geonovum.nl/heronviewer
- SOS via SOS client apps: sensors.geonovum.nl

The next sections will expand on this overview.

9.3 Architecture

The global architecture is depicted below. In all figures in this section the arrows denote the flow of (weather) data. Circles denote continuous data transformation processes. Rectangles denote application servers or services.

The figure below depicts the software architecture at the global level. ETL (Extract, Transform, Load) processes extract and transform raw weather data from the Davis weather station and publish the transformed data to a variety of application servers/services. Each of these servers/services will provide standard (web) APIs through which client applications can fetch (weather) data.

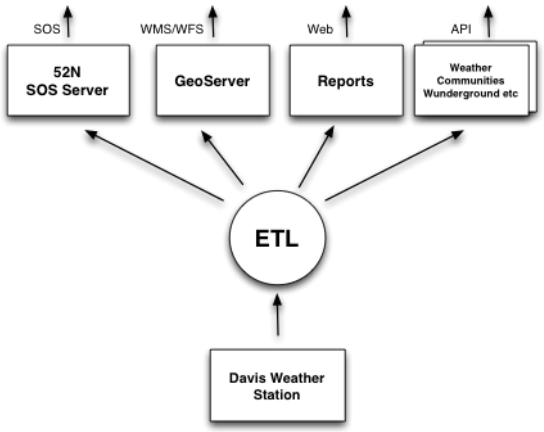


Fig. 9.3: *Figure 2 - Global Software Architecture*

- SOS via the 52N SOS application server
- WMS/WMS-Time/WFS via: GeoServer
- weather reports via standard Apache webserver
- weather APIs via weather-communities like Weather Underground

This global architecture from Figure 2 is expanded into a more detailed design in Figure 3. This shows the various software and storage components, in particular the realization of the ETL-processing.

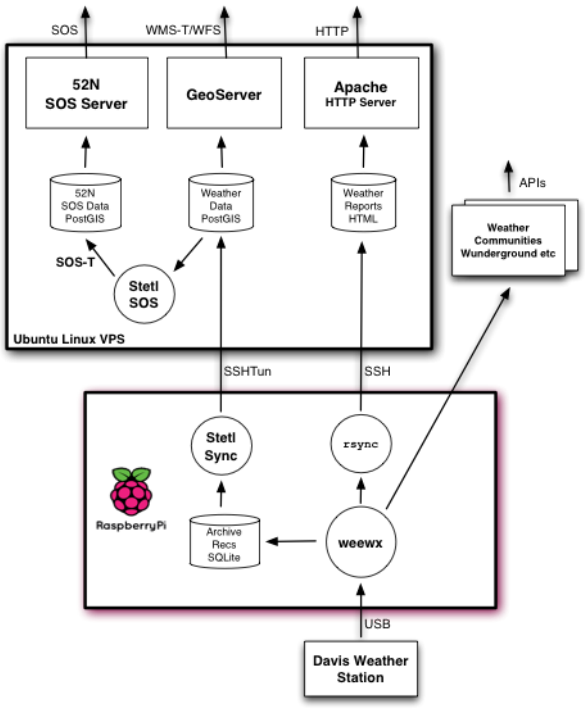


Fig. 9.4: *Figure 3 - Detailed Software Architecture*

The (data) flow in Figure 3 is as follows.

- data is sampled by the `weewx` daemon from the Davis Weather station
- weewx stores *archive records* in a SQLite database
- the *Stetl Sync* process reads the latest data from SQLite database
- *Stetl Sync* publishes these records unaltered to a PostgreSQL database (table: `measurements`)
- several specialized PostgreSQL VIEWs will filter and convert the raw archive data and JOIN data records with the `stations` (location) table date
- the PostgreSQL database (VIEWs) serve directly as data sources for GeoServer WMS/WFS Layers
- GeoServer will also provide a WMS-Dimension (-Time) service using the record timestamp column
- the *Stetl SOS* process reads data from PostgreSQL and transforms this data to SOS-T requests, POSTing these via SOS-T to the 52North SOS
- `weewx` also creates and publishes weather data reports in HTML to be serverd by an Apache server
- in addition `weewx` may publish weather data to various weather community services like Weather Underground (optional)

The above components are divided over two server machines.

- the Raspberry Pi: `weewx` and *Stetl Sync*
- the Ubuntu Linux VPS: GeoServer, SOS server and Apache server plus the PostgreSQL/PostGIS database and the *Stetl SOS* ETL

Connections between the RPi and the VPS are via SSH. An SSH tunnel (`SSHTun`) is maintained to provide a secure connection to the PostgreSQL server on the VPS. This way the PostgreSQL server is never exposed directly via internet. Each of these components are elaborated further below.

Sources for this architecture can be found in GitHub.

- ETL, database and services: https://github.com/Geonovum/sospilot/tree/master/src/weather
- Raspberry Pi system setup: https://github.com/Geonovum/sospilot/tree/master/src/raspberry
- weewx-specific: https://github.com/Geonovum/sospilot/tree/master/src/weewx

In addition, weewx will be configured to report weather data to the Weather Underground personal weather station network at http://www.wunderground.com.

9.4 Raspberry Pi

A Raspberry Pi will be setup as a headless (no GUI) server. Via a USB Cable the Pi will be connected to the Davis datalogger cable. The Pi will run a Debian Linux version (Raspbian) with the free *weewx* weather server and archiver. *weewx* will fetch samples from the Davis weather station, storing its summaries regularly (typically every 5 mins) in a MySQL or SQLite *archive table*. weewx can also can publish data to community Weather networks like Wunderground.

Fig. 9.5: *Figure 4 - Raspberry Pi Package through Install*

See the raspberrypi-install section for the full hardware setup and software installation of the RPi for the project.

9.5 Weather Software

The choice is weewx with SQLite. *weewx* is installed as part of the raspberrypi-install. The *weewx* configuration for the Davis station is maintained in GitHub https://github.com/Geonovum/sospilot/tree/master/src/weewx/davis. After a first test using our WeatherStationAPI custom driver, the Geonovum Davis weather station will be connected.

The web reporting is synced by *weewx* every 5 mins to to our main website (using `rsync` over SSH): http://sensors.geonovum.nl/weewx. This will take about 125kb each 5 mins.

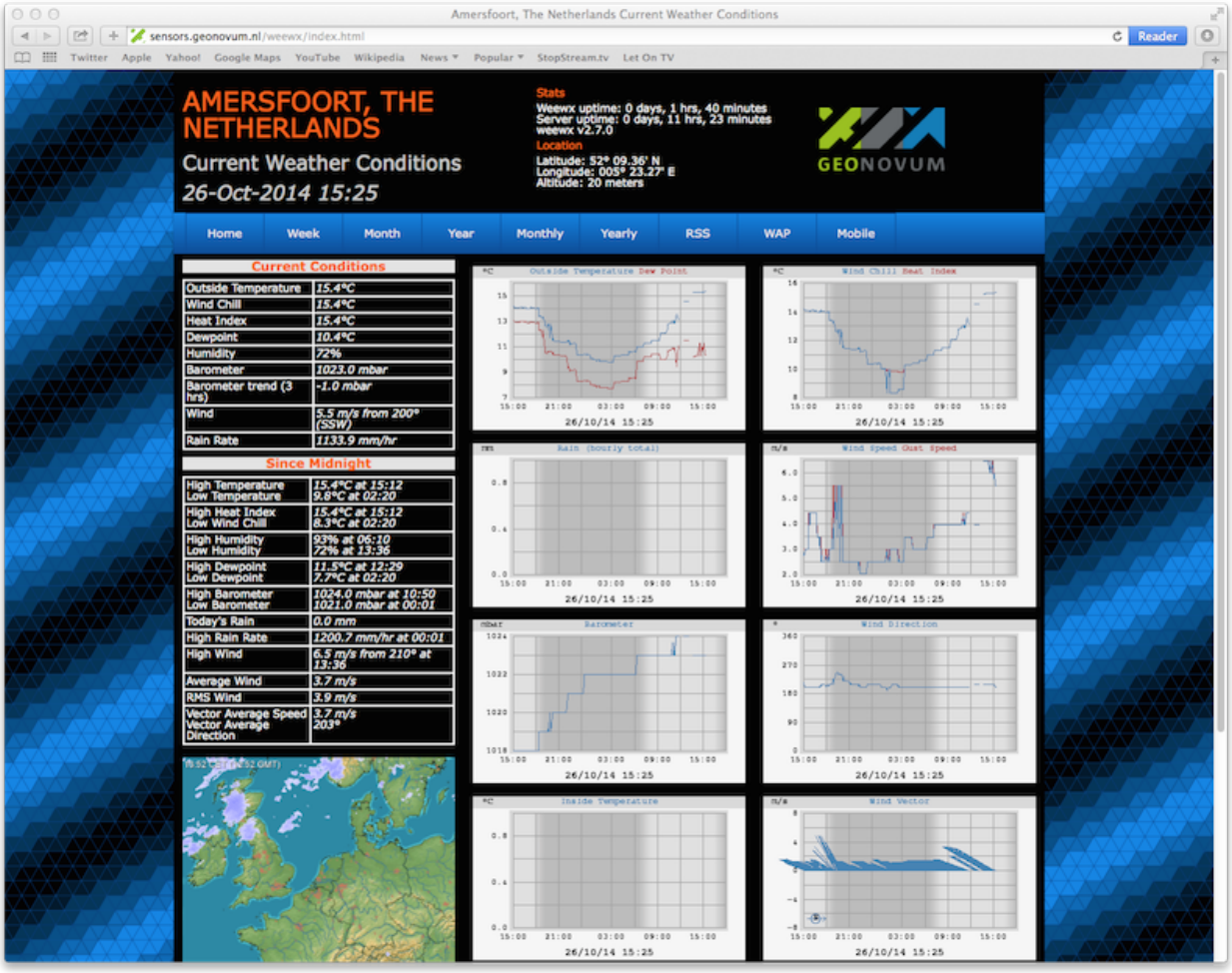


Fig. 9.6: *Figure 5 - weewx standard report screenshot*

In addition *weewx* has been configured to report to the Weather Underground community site. The station is registered as **IUTRECHT96**, http://www.wunderground.com/personal-weather-station/dashboard?ID=IUTRECHT96.

Also, *weewx* has been configured (in sept 2015) to report to the MET UK Weather Observations Website (WOW). The station is registered with WOW site ID **929236001**, http://wow.metoffice.gov.uk/sitehandlerservlet?requestedAction=READ&siteID=929236001. The Dutch KNMI has a localized version, called WOW-NL.

Fig. 9.7: *Figure 6a - Geonovum station on Weather Underground*

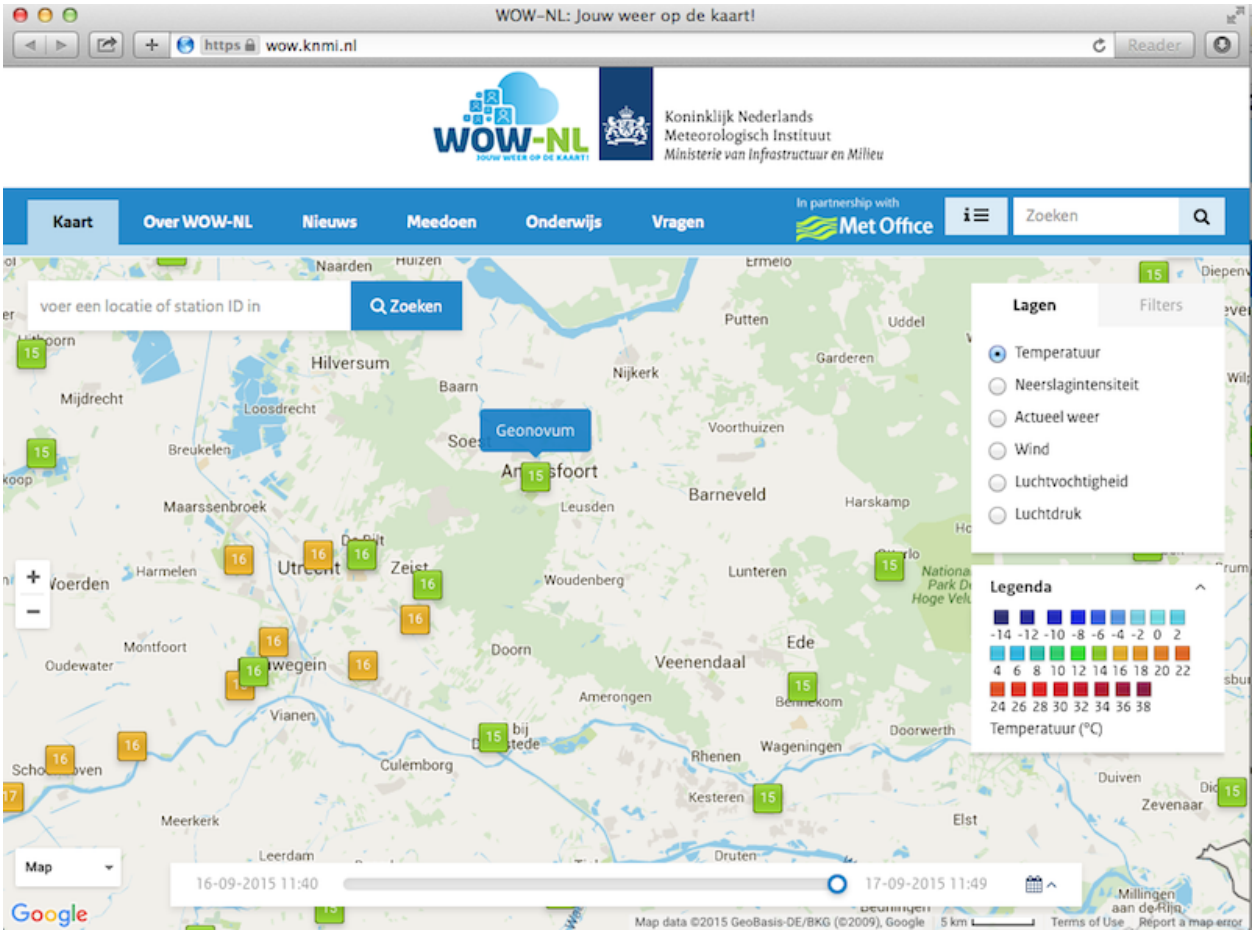


Fig. 9.8: *Figure 6b - Geonovum station on KNMI Weather Observations Website (WOW-NL)*

9.6 PostgreSQL Database

The PostgreSQL database plays a central role. The 52North SOS will maintain its own tables. For the Stetl ETL and GeoServer datasources the following tables and VIEWs are created in the `weather` schema.

```
DROP SCHEMA weather CASCADE;
CREATE SCHEMA weather;

-- Raw measurements table - data from weewx weather archive or possibly other source
-- all units in US metrics assumed!
DROP TABLE IF EXISTS weather.measurements CASCADE;
CREATE TABLE weather.measurements (
  dateTime              INTEGER NOT NULL UNIQUE PRIMARY KEY,
  station_code          INTEGER DEFAULT 33,
  usUnits               INTEGER NOT NULL,
  interval              INTEGER NOT NULL,
  barometer             REAL,
  pressure              REAL,
  altimeter             REAL,
  inTemp                REAL,
  outTemp               REAL,
  inHumidity            REAL,
  outHumidity           REAL,
  windSpeed             REAL,
  windDir               REAL,
  windGust              REAL,
  windGustDir           REAL,
  rainRate              REAL,
  rain                  REAL,
  dewpoint              REAL,
 .
 .
 .
);


--
-- Name: stations; Type: TABLE; Schema: weather; Owner: postgres; Tablespace:
--
DROP TABLE IF EXISTS weather.stations CASCADE;
CREATE TABLE weather.stations (
    gid integer NOT NULL UNIQUE PRIMARY KEY,
    point geometry (Point,4326),
    wmo character varying,
    station_code integer,
    name character varying,
    obs_pres integer,
    obs_wind integer,
    obs_temp integer,
    obs_hum integer,
    obs_prec integer,
    obs_rad integer,
    obs_vis integer,
    obs_clouds integer,
    obs_presweather integer,
    obs_snowdepth integer,
    obs_soiltemp integer,
    lon double precision,
```

```
    lat double precision,
    height double precision
);

CREATE INDEX stations_point_idx ON stations USING gist (point);

INSERT INTO weather.stations (gid, point, wmo, station_code, name, obs_pres, obs_wind,
↪ obs_temp, obs_hum, obs_prec, obs_rad, obs_vis, obs_clouds, obs_presweather, obs_
↪snowdepth, obs_soiltemp, lon, lat, height)
VALUES (1, ST_GeomFromText('POINT(5.372 52.152)', 4326), 'Davis Vantage Pro2', 33,
↪'Geonovum',       1,1,    1,      1,      1,      0,      0,      0,      0,      0,
↪      0, 5.372, 52.152, 32.4);

-- VIEWS

-- SELECT to_timestamp(datetime), "datetime","pressure","outtemp" FROM "weather".
↪"measurements"
DROP VIEW IF EXISTS weather.v_observations CASCADE;
CREATE VIEW weather.v_observations AS
  SELECT
    meas.datetime,
    meas.station_code,
    stations.name as station_name,
    to_timestamp(datetime) as time,
    round(((outtemp-32.0)*5.0/9.0)::numeric) as outtemp_c,
    round((windSpeed*1.61)/3.6::numeric) as windspeed_mps,
    round((windGust*1.61)/3.6::numeric) as windgust_mps,
    round(windDir::numeric) as winddir_deg,
    round(((windchill-32.0)*5.0/9.0)::numeric) as windchill_c,
    meas.rainRate,
    round((pressure*33.8638815)::numeric) as pressure_mbar,
    round(outhumidity::numeric) as outhumidity_perc,
    stations.point as point
  FROM weather.measurements as meas
  INNER JOIN weather.stations AS stations
      ON meas.station_code = stations.station_code ORDER BY datetime DESC;

-- Laatste Metingen per Station
DROP VIEW IF EXISTS weather.v_last_observations CASCADE;
CREATE VIEW weather.v_last_observations AS
  SELECT DISTINCT ON (station_code) station_code,
    station_name,
    datetime,
    time,
    outtemp_c,
    windspeed_mps,
    windgust_mps,
    winddir_deg,
    windchill_c,
    rainRate,
    pressure_mbar,
    outhumidity_perc,
    point
  FROM weather.v_observations;
```

The raw weather data is stored in the `measurements` table (US units). In order to make the tables/VIEWs geospatially enabled, a `stations` table is added. The stations table is modeled after existing KNMI station data. Only a single station is added for now, the Geonovum Davis station. The `measurements` table has a `station_code`

column to facilitate JOINs with the `stations` table.

Via VIEWs more simple and geospatially-enabled data is created. Also the VIEWs take care of conversion from US to metric units. The `weather.v_observations` VIEW contains a selection of weather characteristics joined with station data. `weather.v_last_observations` contains the last (current) observations per station. Below an example of data in the view.

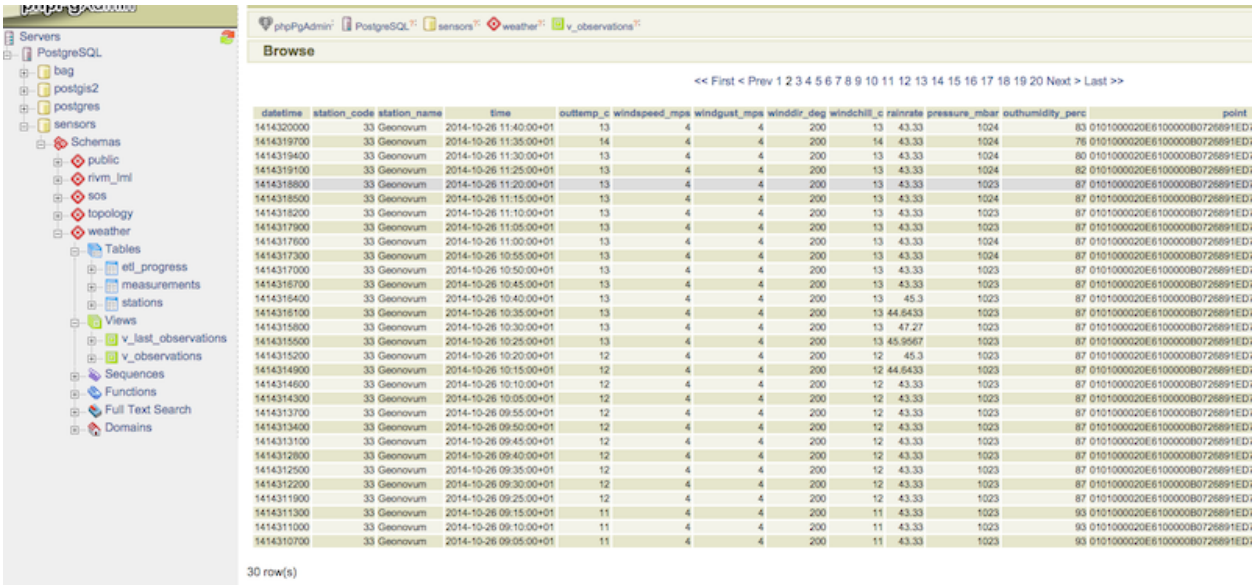


Fig. 9.9: *Figure 7 - PostgreSQL weather.v_observations VIEW*

9.7 Stetl Sync

This section describes the *Stetl Sync* processing within the RPi. Effectively this process will synchronize the latest data from the `weewx` database to a remote PostgreSQL database on the VPS.

All sources can be found here.

weewx stores 'archive' data within a SQLite DB file *weewx.sdb*. Statistical data is derived from this data. Within *weewx.sdb* there is a single table *archive*. The database/table structure (only relevant fields shown).

```
CREATE TABLE archive (
    dateTime INTEGER NOT NULL UNIQUE PRIMARY KEY,
    usUnits INTEGER NOT NULL,
    interval INTEGER NOT NULL,
    barometer REAL,
    pressure REAL,
    altimeter REAL,
    inTemp REAL,
    outTemp REAL,
    inHumidity REAL,
    outHumidity REAL,
    windSpeed REAL,
    windDir REAL,
    windGust REAL,
    windGustDir REAL,
    rainRate REAL,
```

```
    rain REAL,
    dewpoint REAL,
    windchill REAL,
    ....
);
```

Most of the Stetl Sync processing can be realizeed with standard Stetl components like for SQLite input and PostgreSQL publishing. Only synchronization tracking needs a small Stetl input component WeewxDBInput. This component keeps track of the *last archive data record synced* within a PostgreSQL record. At a later stage this may also become a Stetl component so the complete ETL could be effected in Stetl.

The Stetl config is as follows.

```
# weewx archive data in SQLite to Postgres/PostGIS output - Stetl config
#
# Just van den Broecke - 2014
#
# Incrementally reads raw weewx archive records and publishes these to
# PostGIS.

# The main Stetl ETL chain
[etl]
chains = input_weewx_db|output_postgres_insert


# for reading files from weewx SQLite, tracking progress in Postgres
[input_weewx_db]
class = weewxdbinput.WeewxDbInput
host = {host}
port = {port}
database = {database}
user = {user}
password = {password}
schema = {schema}
progress_query = SELECT * from etl_progress WHERE worker = 'weewx2postgres'
progress_update = UPDATE etl_progress SET last_id = %d, last_time = '%s', last_update␣
↪= current_timestamp WHERE worker = 'weewx2postgres'
table = archive
query = SELECT * from archive WHERE dateTime > %d ORDER BY dateTime LIMIT 100
database_name = {weewx_db}
output_format = record_array

[output_std]
class = outputs.standardoutput.StandardOutput

# For inserting file records
[output_postgres_insert]
class = outputs.dboutput.PostgresInsertOutput
input_format = record_array
host = {host}
database = {database}
user = {user}
password = {password}
schema = {schema}
table = {table}
key=dateTime
```

The target table is the PostgreSQL `weather.measurements` table depicted above.

The synchronization state is tracked in a PostgresQL table. A single *worker* (see Stetl config above) is inserted as well:

```
-- ETL progress tabel, houdt bij voor ieder ETL proces ("worker") wat het
-- laatst verwerkte record id is van hun bron tabel.
DROP TABLE IF EXISTS weather.etl_progress CASCADE;
CREATE TABLE weather.etl_progress (
  gid          SERIAL,
  worker       CHARACTER VARYING(25),
  source_table CHARACTER VARYING(25),
  last_id      INTEGER,
  last_time    CHARACTER VARYING(25) DEFAULT '-',
  last_update  TIMESTAMP,
  PRIMARY KEY (gid)
);

-- Define workers
INSERT INTO weather.etl_progress (worker, source_table, last_id, last_update)
  VALUES ('weewx2postgres', 'sqlite_archive', 0, current_timestamp);
```

The Stetl Sync process is scheduled via `cron` to run typically every 4 minutes.

```
# Cronfile for ETL processes on Raspberry Pi

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
SOSPILOT=/opt/geonovum/sospilot/git

# Run ETL step 1: Raw weewx data from SQLite to remote Postgres DB on VPN
*/4 * * * * cd $SOSPILOT/src/weather/weewx2pg; ./pi-etl.sh >> /var/log/sospilot/
↪weewx2pg.log 2>&1
```

The `pi-etl.sh` shell-script will first setup an SSH tunnel, then call the Stetl-wrapper `weewx2pg.sh` and then tear down the SSH-tunnel.

```
#!/bin/bash
#
# Script to invoke ETL on the Raspberry Pi
# Uses an SSH tunnel to connect to Postgres on the VPS
#

# Kill possible (hanging) background SSH tunnel
function killTunnel() {
    pstree -p sadmin | grep 'ssh(' | cut -d'(' -f2 | cut -d')' -f1|xargs kill -9 > /
↪dev/null 2>&1
}


# Kill possible (hanging) background SSH tunnel
killTunnel

# Setup SSH tunnel to remote host
ssh -f -L 5432:sensors:5432 sadmin@sensors -4 -g -N
sleep 10
ps aux | grep 5432

# Do the ETL
./weewx2pg.sh

# Kill the background SSH tunnel
```

```
killTunnel
```

The `weewx2pg.sh` script is as follows.

```
#!/bin/bash
#
# ETL for converting/harvesting weewx archive data into PostGIS
#

# Usually requried in order to have Python find your package
export PYTHONPATH=.:$PYTHONPATH

stetl_cmd=stetl

# debugging
# stetl_cmd=../../../../stetl/git/stetl/main.py

# Set Stetl options

. ../options.sh

$stetl_cmd -c weewx2pg.cfg -a "$options"
```

The `options.sh` script will set various (shell) variables to be substituted in the Stetl config.

```
#!/bin/sh
#
# Sets host-specific variables
# To add your localhost add options-<your hostname>.sh in this directory

# All file locations are relative to the specific ETL subdirs like weewx2pg
. ../options-`hostname`.sh

export options="host=localhost port=5432 weewx_db=$WEEWX_DB user=$PGUSER password=
↪$PGPASSWORD database=sensors schema=weather table=measurements"
```

By calling the `options-<hostname>.sh` script, various host-specific/secured variables are set.

9.8 GeoServer

The `stations` table and two VIEWs are used as data sources for weather-layers:

- `weather.stations` as a source for a WMS Layer `sensors:weather_stations`
- `weather.v_observations` as a source for a timeseries WMS-Dimension Layer `sensors:weather_observations` using continuous interval option
- `weather.v_last_observations` for a WMS last observation layer `sensors:weather_last_observations`

These three layers were easily integrated in the SOSPilot Heron Viewer.

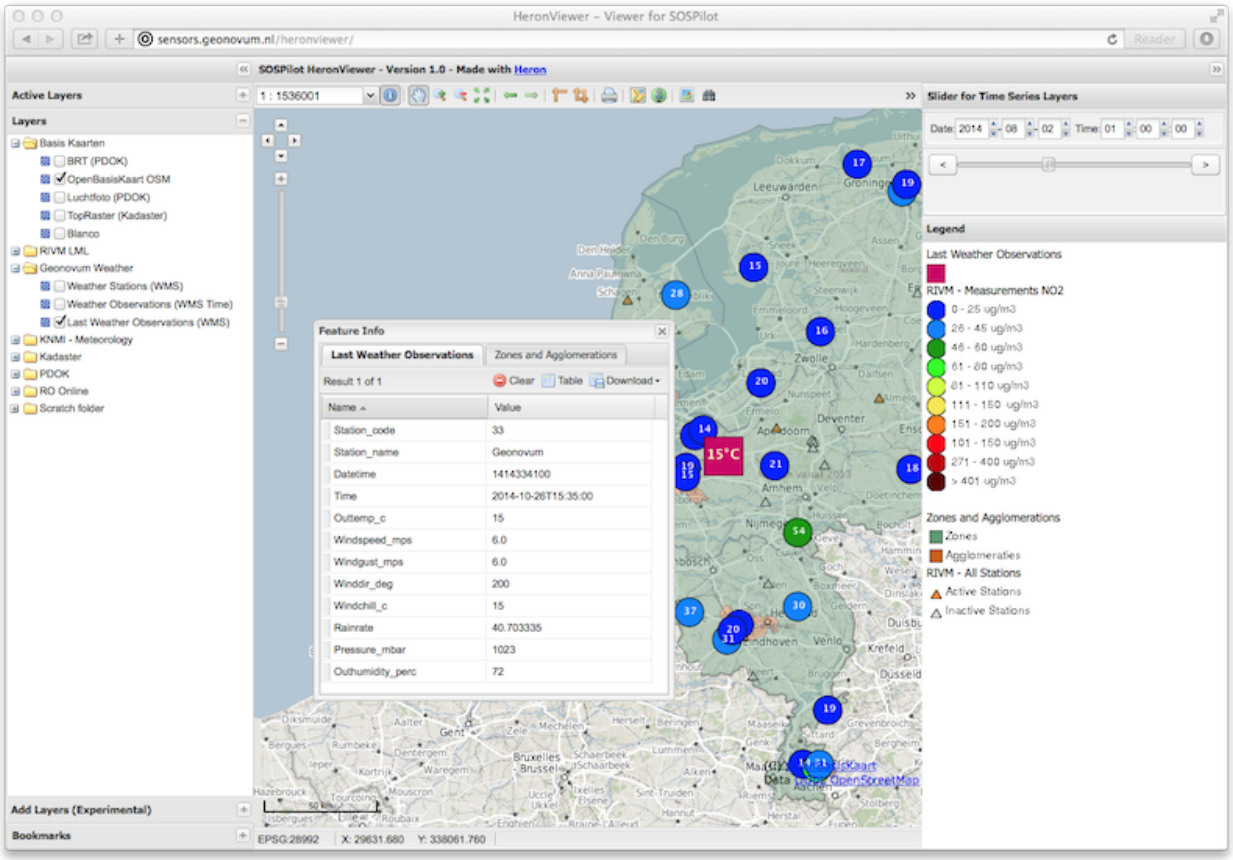


Fig. 9.10: *Figure 8 - Weather Data WMS Layers in Heron viewer*

9.9 Stetl SOS

This ETL process reads measurements from PostgreSQL and transforms/publishes these to the SOS via SOS-T. The design Similar to the RIVM LML AQ SOS publishing setup.

Source code: https://github.com/Geonovum/sospilot/tree/master/src/weather/pg2sos

As the weather data is published in the same SOS as the AQ data, both data types can be combined in any of the SOS browser clients. An example can be seen in Figure 8: both outside temperature (deg) and humidity (%) are combined with NO2 (Nitrogen Dioxide) and PM10 (Particulate Matter up to 10 micrometers in size).

Fig. 9.11: *Figure 9 - Weather Data SOS Data integrated in 52N JS Client*

Test with SOS requests: **NB The SOS is not active anymore**

- Station: DescribeSensor
- Temperature observations: GetObservation

9.10 Links

- *RGI-189 Sensoren als databronnen aan de geo-informatie infrastructuur*, Wiel Wauben, KNMI http://www.knmi.nl/~wauben/HIM/SWE%20KNMI%20evaluatie%20v3.pdf

CHAPTER 10

Raspberry Pi Installation

Below the setup and installation of the Raspberry Pi (RPi) for `weewx` weather software with the Davis Vantage Pro2 weather station is described.

10.1 Conventions

General conventions.

10.1.1 Directories

We keep the following directory conventions:

- `/opt` additional, manually installed software
- `/opt/<product>/<product-version>` subdires e.g. `/opt/weewx/weewx-2.7.0`
- `/opt/bin` own shell scripts
- `/opt/geonovum/sospilot/git` all project software synced with GitHub https://github.com/Geonovum/sospilot
- `/home/sadmin` home dir beheer account

Under `/opt/<product>/<product-version>` we often make a dynamic link to the latest version of a product, for example `/opt/weewx/weewx` is a dynlink to `/opt/weewx/weewx-2.7.0`.

Add to `/etc/profile`

```
export PATH=${PATH}:/opt/bin
```

10.1.2 System

Ordered Oct 16, 2014. Delivered Oct 17, 2014. From http://www.kiwi-electronics.nl.

Fig. 10.1: *Figure 1 - Raspberry Pi Package through Install*

Specs.

```
Raspberry Pi Model B+ bundle                    € 68,95
  - with 2A power adapter
  - microSD 32GB Transcend class 10
  - Color case: Transparant Multicomp enclosure

Wi-Pi Draadloze USB Adapter voor Raspberry Pi   € 18,95

Belkin 4-Port Powered Mobile USB 2.0 Hub        € 16,95

Specs:
Chip Broadcom BCM2835 SoC
Core architecture ARM11
CPU 700 MHz Low Power ARM1176JZFS Applications Processor
GPU Dual Core VideoCore IV® Multimedia Co-Processor
Provides Open GL ES 2.0, hardware-accelerated OpenVG, and
1080p30 H.264 high-profile decode
Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering
and DMA infrastructure
Memory 512MB SDRAM
Operating System Boots from Micro SD card, running Linux operating system
Dimensions 85 x 56 x 17mm
Power Micro USB socket 5V, 2A

Connectors:
Ethernet 10/100 BaseT Ethernet socket
Video Output HDMI (rev 1.3 & 1.4)
Composite RCA (PAL and NTSC)
```

```
Audio Output 3.5mm jack, HDMI
USB 4 x USB 2.0 Connector
GPIO Connector 40-pin 2.54 mm (100 mil) expansion header: 2x20 strip
Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines
Camera Connector 15-pin MIPI Camera Serial Interface (CSI-2)
JTAG Not populated
Display Connector Display Serial Interface (DSI) 15 way flat flex cable connector
with two data lanes and a clock lane
Memory Card Slot SDIO
```

See also https://www.adafruit.com/datasheets/pi-specs.pdf. The Belkin 4-Port Powered Mobile USB is recommended when attaching WiPi and Davis USB devices, as the RPi may have loss USB-power issues.

10.1.3 Hardware Setup

Multicomp enclosure install: https://www.youtube.com/watch?v=1uFMFZMGO_A

10.1.4 OS Setup

SD card appeared to be non-readable. Reformatted and put NOOBS (probably could have put Raspbian directly) on card. Described here http://www.raspberrypi.org/help/noobs-setup . We will install the Raspbian OS, a Debian Linux variant for the Pi:

```
format SD card using  https://www.sdcard.org/downloads/formatter_4
select complete format
unpack NOOBS and put files on card
put card in Pi and Reboot
select Raspian
Standard install steps
NB the password to be entered is for the user 'pi' not root!
user pi is sudoer
root password may be set using sudo passwd root
chosen non-GUI at startup, can always get GUI via startx command
```

10.1.5 Wifi Setup

Wifi setup with WiPi. See doc http://www.element14.com/community/servlet/JiveServlet/downloadBody/49107-102-1-257014/Wi_Pi.User_Manual.pdf Steps.

```
$ root@raspberrypi:~# sudo nano /etc/network/interfaces
```

The file will already have a network entry for the localhost, or loopback network interface, and the Ethernet socket, known as eth0. We're going to add a new interface called wlan0, There are two slightly different ways of editing this file, depending on which of type broad types of encryption is in use on the WiFi network you wish to connect to. In the case of WPA/WPA2, add the following lines to the end of the interfaces document:

```
auto wlan0
iface wlan0 inet dhcp
wpa-ssid <name of your WiFi network>
wpa-psk <password of your WiFi network>
```

In the case of WEP, add the following instead

auto wlan0 iface wlan0 inet dhcp wireless-essid <name of your WiFi network> wireless-key <password of your WiFi network>

Result in `/etc/network/interfaces`

```
root@georasp:~# cat /etc/network/interfaces
auto lo

iface lo inet loopback
iface eth0 inet dhcp

# allow-hotplug wlan0
# iface wlan0 inet manual
# wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp

auto wlan0
iface wlan0 inet dhcp
wpa-ssid <name of your WiFi network>
wpa-psk <password of your WiFi network>
```

But to have mutiple WLANs and not have passwords in files, this approach is more flexible and more secure. http://www.geeked.info/raspberry-pi-add-multiple-wifi-access-points/

Our */etc/network/interfaces* is now

```
auto lo

iface lo inet loopback
# allow-hotplug eth0
iface eth0 inet dhcp

allow-hotplug wlan0
auto wlan0
iface wlan0 inet dhcp

pre-up wpa_supplicant -Dwext -i wlan0 -c /etc/wpa_supplicant.conf -B
```

And in the file `/etc/wpa_supplicant.conf` configure multiple WIFI stations. For each station generate a PSK as follows `wpa_passphrase <ssid> <passphrase>`. `/etc/wpa_supplicant.conf` will become something like:

```
ctrl_interface=/var/run/wpa_supplicant
#ap_scan=2

network={
        ssid="<station #1 name>"
        scan_ssid=1
        proto=WPA RSN
        key_mgmt=WPA-PSK
        pairwise=CCMP TKIP
        group=CCMP TKIP
        psk=<generated PSK #1>
}

network={
        ssid="<station #2 name>"
        scan_ssid=1
        proto=WPA RSN
```

```
        key_mgmt=WPA-PSK
        pairwise=CCMP TKIP
        group=CCMP TKIP
        psk=<generated PSK #2>
}
```

The latter approach with `wpa_supplicant` did somehow not work so we remained in the first simple approach without `wpa_supplicant`, only a simple `/etc/network/interfaces` config.

Bogger: Wifi seems to go down from time to time with `wlan0: CTRL-EVENT-DISCONNECTED reason=4` in syslog. Will use a script in cron to always keep Wifi up. For topic see http://www.raspberrypi.org/forums/viewtopic.php?t=54001&p=413095. See script at https://github.com/Geonovum/sospilot/blob/master/src/raspberry/wificheck.sh and Monitoring section below.

10.1.6 Hostname

In `/etc/hostname` set to `georasp`..

10.1.7 Accounts

Two standard accounts: `root` ("root admin") en `sadmin` ("sensors admin"). NB account `root` is never a login account on Ubuntu/Debian!

Het beheer-account `root` heeft root-rechten.

Het account `sadmin` heeft ook wat rechten maar minder. Het account `sadmin` heeft lees/schrijfrechten op directories voor custom installaties (zie onder).

10.1.8 Software Installation

Via Ubuntu/Debian Advanced Packaging Tool (APT) . Hiermee is op zeer eenvoudige wijze niet alleen alle software, inclusief de meeste GIS tools gemakkelijk te installeren, maar ook up-to-date te houden. Bijvoorbeeld een complete Java installatie gaat met : `apt-get install sun-java6-jdk`. APT wordt altijd door het `root` account (met root via sudo of sudo -i) uitgevoerd.

Alleen in een uiterst geval waarbij een software product niet in het APT systeem zit of niet in een gewenste versie is een handmatige ("custom") installatie gedaan. Hierbij is de volgende conventie aangehouden: custom installaties worden door het account `root`.

10.2 Software - General

Install of standard packages.

10.2.1 nginx Web Server

As Apache2 seems to have a relative large footprint, many prefer nginx as webserver on RPi. (Though for now, no webserver is used nor required). Setup.

```
apt-get install nginx

# start/stop server
/etc/init.d/nginx start
/etc/init.d/nginx stop
```

Config under `/etc/nginx` especially, default website at `/etc/nginx/sites-available/default`

```
server {
        #listen   80; ## listen for ipv4; this line is default and implied
        #listen   [::]:80 default_server ipv6only=on; ## listen for ipv6

        root /usr/share/nginx/www;
        index index.html index.htm;

        # Make site accessible from http://localhost/
        server_name localhost;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ /index.html;
                # Uncomment to enable naxsi on this location
                # include /etc/nginx/naxsi.rules
        }

        location /doc/ {
                alias /usr/share/doc/;
                autoindex on;
                allow 127.0.0.1;
                allow ::1;
                deny all;
        }
}
```

10.3 Installation - Project Software

Software and documentation for the project, e.g. `weewx` config, are in the project GitHub: https://github.com/Geonovum/sospilot

Installed under `/opt/geonovum/sospilot`

```
cd /opt/geonovum/sospilot
git clone https://github.com/Geonovum/sospilot.git git
```

NB all documentation (Sphinx) is automagically published after each Git commit/push to ReadTheDocs.org: http://sospilot.readthedocs.org via a standard GitHub Post-commit hook.

The following refresh script is handy to undo local changes and sync with master.

```
# Refresh from original Repo
# WARNING will remove all local changes!!!
# except for files not in Git

git fetch --all
git reset --hard origin/master
```

See https://github.com/Geonovum/sospilot/blob/master/refresh-git.sh

10.4 Installation - Weather Software

10.4.1 weewx - Weather Station server

Home: weewx.

Install under `/opt/weewx`. Custom install as user *sadmin* in order to facilitate custimization.

See http://www.weewx.com/docs/setup.htm

Steps.

```
# Install Dependencies
# required packages:
apt-get install python-configobj
apt-get install python-cheetah
apt-get install python-imaging
apt-get install fonts-freefont-ttf  # Fonts in reporting

# optional for extended almanac information:
apt-get install python-dev
apt-get install python-setuptools
easy_install pip
pip install pyephem

# Weewx install after download
cd /opt/weewx
tar xzvf archive/weewx-2.7.0.tar.gz
ln -s weewx-2.7.0 weewx

cd weewx

# Change install dir in setup.cfg as follows
# Configuration file for weewx installer. The syntax is from module
# ConfigParser. See http://docs.python.org/library/configparser.html

[install]

# Set the following to the root directory where weewx should be installed
home = /opt/weewx/weewxinst

# Given the value of 'home' above, the following are reasonable values
prefix =
exec-prefix =
install_lib = %(home)s/bin
install_scripts = %(home)s/bin

# build en install in /opt/weewx/weewxinst
./setup.py build
./setup.py install

# test install
# change
cd /opt/weewx/weewxinst
change station_type = Simulator in weewx.conf
```

```
# link met aangepaste configs uit Geonovum GitHub (na backup oude versies)
ln -s /opt/geonovum/sospilot/git/src/weewx/test/weewx.conf /opt/weewx/weewxinst
ln -s /opt/geonovum/sospilot/git/src/weewx/test/skin.conf /opt/weewx/weewxinst/skins/
↪Standard
ln -s /opt/geonovum/sospilot/git/src/weewx/test/weatherapidriver.py /opt/weewx/
↪weewxinst/bin/user

# test OK
sadmin@georasp /opt/weewx/weewxinst $ ./bin/weewxd weewx.conf
LOOP:   2014-10-19 16:18:50 CEST (1413728330) {'heatindex': 32.67858297022247,
↪'barometer': 31.099999998967093, 'windchill': 32.67858297022247,
'dewpoint': 27.203560993945757, 'outTemp': 32.67858297022247, 'outHumidity': 79.
↪99999996901272, 'UV': 2.5568864075841278,
'radiation': 182.63474339886625, 'rain': 0, 'dateTime': 1413728330, 'windDir': 359.
↪9999998140763, 'pressure': 31.099999998967093,
'windSpeed': 5.164547900449179e-09, 'inTemp': 63.00000002065819, 'windGust': 6.
↪197456769996279e-09, 'usUnits': 1, 'windGustDir': 359.9999998140763}
LOOP:   2014-10-19 16:18:52 CEST (1413728332) {'heatindex': 32.67676549144743,
↪'barometer': 31.099999990703814, 'windchill': 32.67676549144743,
'dewpoint': 27.20178958368346, 'outTemp': 32.67676549144743, 'outHumidity': 79.
↪99999972111442, 'UV': 2.555313141990661,
'radiation': 182.52236728504724, 'rain': 0, 'dateTime': 1413728332, 'windDir': 359.
↪9999983266865, 'pressure': 31.099999990703814,
'windSpeed': 4.648092932768577e-08, 'inTemp': 63.00000018592372, 'windGust': 5.
↪577711537085861e-08, 'usUnits': 1, 'windGustDir': 359.9999983266865}

# install weewx daemon in /etc/init.d (als root)
# aanpassen settings in daemon in GitHub  /opt/geonovum/sospilot/git/src/weewx/test/
↪weewx-daemon.sh

# PATH should only include /usr/* if it runs after the mountnfs.sh script
WEEWX_HOME=/opt/weewx/weewxinst
PATH=/sbin:/usr/sbin:/bin:/usr/bin
WEEWX_BIN=$WEEWX_HOME/bin/weewxd
WEEWX_CFG=$WEEWX_HOME/weewx.conf
DESC="weewx weather system"
NAME=weewx
WEEWX_USER=sadmin:sadmin
PIDFILE=$WEEWX_HOME/$NAME.pid
DAEMON=$WEEWX_BIN
DAEMON_ARGS="--daemon --pidfile=$PIDFILE $WEEWX_CFG"
SCRIPTNAME=/etc/init.d/$NAME

cp /opt/geonovum/sospilot/git/src/weewx/davis/weewx-deamon.sh /etc/init.d/weewx
update-rc.d weewx defaults
/etc/init.d/weewx start
/etc/init.d/weewx status
* Status of weewx weather system: running

# weewx log bekijken
tail -f /var/log/syslog

# memory in gaten houden
  PID USER      PR  NI    VIRT    RES    SHR  S  %CPU %MEM     TIME+ COMMAND
 4688 sadmin    20   0    170936  36776  4608 S   0.0  0.5   3:15.23 weewxd  (16.10.
↪14 16:22)
```

```
# nginx ontsluiting
location /weewx {
    alias /opt/weewx/weewxinst/public_html;
    autoindex on;
    allow 127.0.0.1;
    allow ::1;
    allow all;
}
```

10.4.2 Extra voor TFA Nexus Pro

TE923 driver. Nodig *pyusb*

```
pip install pyusb
# geeft: DistributionNotFound: No distributions matching the version for pyusb

# tweede try:
apt-get install python-usb
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  python-usb
0 upgraded, 1 newly installed, 0 to remove and 5 not upgraded.
Need to get 17.7 kB of archives.
After this operation, 132 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main python-usb armhf 0.4.3-
↪1 [17.7 kB]
Fetched 17.7 kB in 0s (37.9 kB/s)
Selecting previously unselected package python-usb.
(Reading database ... 83706 files and directories currently installed.)
Unpacking python-usb (from .../python-usb_0.4.3-1_armhf.deb) ...
Setting up python-usb (0.4.3-1) ...

# root@otterpi:/opt/weewx/weewxinst# lsusb
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 05e3:0608 Genesys Logic, Inc. USB-2.0 4-Port HUB
Bus 001 Device 005: ID 1130:6801 Tenx Technology, Inc.
Bus 001 Device 006: ID 148f:5370 Ralink Technology, Corp. RT5370 Wireless Adapter

# nu andere error van weewx
Dec 10 15:02:51 otterpi weewx[2645]: te923: Found device on USB bus=001 device=005
Dec 10 15:02:51 otterpi weewx[2645]: te923: Unable to claim USB interface 0: could␣
↪not claim interface 0: Operation not permitted
Dec 10 15:02:51 otterpi weewx[2645]: wxengine: Unable to open WX station hardware:␣
↪could not claim interface 0: Operation not permitted
Dec 10 15:02:51 otterpi weewx[2645]: wxengine: Caught WeeWxIOError: could not claim␣
↪interface 0: Operation not permitted
Dec 10 15:02:51 otterpi weewx[2645]:     ****  Exiting...

# may have to do with udev usb rules
http://www.tdressler.net/ipsymcon/te923.html

vi /etc/udev/rules.d/99-te923.rules
```

```
Inhalt:
ATTRS{idVendor}=="1130", ATTRS{idProduct}=="6801", MODE="0660", GROUP="plugdev", RUN=
↪"/bin/sh -c 'echo -n $id:1.0 > /sys/bus/usb/drivers/usbhid/unbind'"

udevadm control --reload-rules

adduser sadmin plugdev
```

10.5 Installation - Weather Hardware

The Davis weather station is mounted on the Geonovum roof. The Vantage Pro2 console is connected via RF to the outside station. The console includes Weatherlink for archiving and USB connectivity. Via the console the station can be configured. Via weewx the status can be obtained

Several sensors seem to be non-active.

```
./wee_config_vantage --info --config /opt/weewx/weewxinst/weewx.conf
Using configuration file /opt/weewx/weewxinst/weewx.conf.
Querying...
Davis Vantage EEPROM settings:

    CONSOLE TYPE:                   VantagePro2

    CONSOLE FIRMWARE:
      Date:                         Jul 14 2008
      Version:                      1.80

    CONSOLE SETTINGS:
      Archive interval:             1800 (seconds)
      Altitude:                     98 (meter)
      Wind cup type:                large
      Rain bucket type:             0.2 MM
      Rain year start:              10
      Onboard time:                 2014-11-03 14:48:51

    CONSOLE DISPLAY UNITS:
      Barometer:                    hPa
      Temperature:                  degree_10F
      Rain:                         mm
      Wind:                         km_per_hour

    CONSOLE STATION INFO:
      Latitude (onboard):           +52.2
      Longitude (onboard):          +5.4
      Use manual or auto DST?       AUTO
      DST setting:                  N/A
      Use GMT offset or zone code?  ZONE_CODE
      Time zone code:               21
      GMT offset:                   N/A

    TRANSMITTERS:
      Channel 1:                    iss
      Channel 2:                    (N/A)
      Channel 3:                    (N/A)
      Channel 4:                    (N/A)
      Channel 5:                    (N/A)
```

```
    Channel 6:                    (N/A)
    Channel 7:                    (N/A)
    Channel 8:                    (N/A)

  RECEPTION STATS:
    Total packets received:       109
    Total packets missed:         3
    Number of resynchronizations: 0
    Longest good stretch:         41
    Number of CRC errors:         0

  BAROMETER CALIBRATION DATA:
    Current barometer reading:    29.405 inHg
    Altitude:                     98 feet
    Dew point:                    255 F
    Virtual temperature:          -89 F
    Humidity correction factor:   23
    Correction ratio:             1.005
    Correction constant:          +0.000 inHg
    Gain:                         0.000
    Offset:                       9.000

  OFFSETS:
    Wind direction:               +0 deg
    Inside Temperature:           +0.0 F
    Inside Humidity:              +0%
    Outside Temperature:          -1.0 F
    Outside Humidity:             +0%
```

Also http://www.weewx.com/docs/usersguide.htm#wee_config_vantage to clear archive and set archive interval to 5 mins.

Problem: temperature and humidity sensors not working! Working after 3 hours on 20:00, then failing at 0800. Also low station battery message.

May also be the Davis Supercap Problem: http://vp-kb.wikispaces.com/Supercap+fault

10.6 Installation - ETL Tools

10.6.1 XSLT Processor

Zie http://en.wikipedia.org/wiki/XSLT. *XSLT (XSL Transformations) is a declarative, XML-based language used for the transformation of XML documents into other XML documents.*

Installatie van XSLT processor voor commandline. o.a. gebruikt voor INSPIRE GML transformaties.

```
apt-get install xsltproc
```

10.6.2 SQLite

weewx uses SQLite to store weather records. Command line tools.

```
apt-get install sqlite3
```

10.6.3 Postgres Client

Just need *psql* for now plus libs (*psycopg2*) for Stetl.

```
apt-get  install postgresql-client
```

10.6.4 GDAL/OGR

Volgens de website www.gdal.org.

GDAL is a translator library for raster geospatial data formats that is released under an X/MIT style Open Source license by the Open Source Geospatial Foundation. The related OGR library (which lives within the GDAL source tree) provides a similar capability for simple features vector data.

Installatie is simpel via APT.

```
$ apt-get install gdal-bin python-gdal

# Error...! 2e keer gaat goed na  apt-get update --fix-missing
Fetched 15.6 MB in 18s (838 kB/s)
Failed to fetch http://mirrordirector.raspbian.org/raspbian/pool/main/m/mysql-5.5/
 ↪mysql-common_5.5.38-0+wheezy1_all.deb  404  Not Found
Failed to fetch http://mirrordirector.raspbian.org/raspbian/pool/main/m/mysql-5.5/
 ↪libmysqlclient18_5.5.38-0+wheezy1_armhf.deb  404  Not Found

Setting up libgeos-3.3.3 (3.3.3-1.1) ...
Setting up proj-bin (4.7.0-2) ...
Setting up gdal-bin (1.9.0-3.1) ...
python-gdal_1.9.0-3.1_armhf.deb
```

10.6.5 Stetl - Streaming ETL

Zie http://stetl.org

First all dependencies!

```
apt-get install python-pip python-lxml libgdal-dev python-psycopg2
```

Normaal doen we `pip install stetl` maar nu even install uit Git vanwege te verwachten updates. Install vanuit GitHub versie onder `/opt/stetl/git` (als user *sadmin*).

```
$ mkdir /opt/stetl
$ cd /opt/stetl
$ git clone https://github.com/geopython/stetl.git git
$ cd git
$ python setup.py install  (als root)

$ stetl -h
# 2014-10-21 18:40:37,819 util INFO Found cStringIO, good!
# 2014-10-21 18:40:38,585 util INFO Found lxml.etree, native XML parsing, fabulous!
# 2014-10-21 18:40:41,636 util INFO Found GDAL/OGR Python bindings, super!!
# 2014-10-21 18:40:41,830 main INFO Stetl version = 1.0.7rc13
```

Installatie Testen.

```
$ which stetl
# /usr/local/bin/stetl

cd /opt/stetl/git/examples/basics
./runall.sh
# OK!
```

10.6.6 Python Jinja2

Needed for Stetl Jinja2 templating Filter.

```
pip install jinja2
Downloading/unpacking jinja2
  Downloading Jinja2-2.7.3.tar.gz (378kB): 378kB downloaded
  Running setup.py (path:/tmp/pip_build_root/jinja2/setup.py) egg_info for package␣
↪jinja2

    warning: no files found matching '*' under directory 'custom_fixers'
    warning: no previously-included files matching '*' found under directory 'docs/_
↪build'
    warning: no previously-included files matching '*.pyc' found under directory
↪'jinja2'
    warning: no previously-included files matching '*.pyc' found under directory 'docs
↪'
    warning: no previously-included files matching '*.pyo' found under directory
↪'jinja2'
    warning: no previously-included files matching '*.pyo' found under directory 'docs
↪'
Downloading/unpacking markupsafe (from jinja2)
  Downloading MarkupSafe-0.23.tar.gz
  Running setup.py (path:/tmp/pip_build_root/markupsafe/setup.py) egg_info for␣
↪package markupsafe

Installing collected packages: jinja2, markupsafe
  Running setup.py install for jinja2

    warning: no files found matching '*' under directory 'custom_fixers'
    warning: no previously-included files matching '*' found under directory 'docs/_
↪build'
    warning: no previously-included files matching '*.pyc' found under directory
↪'jinja2'
    warning: no previously-included files matching '*.pyc' found under directory 'docs
↪'
    warning: no previously-included files matching '*.pyo' found under directory
↪'jinja2'
    warning: no previously-included files matching '*.pyo' found under directory 'docs
↪'
  Running setup.py install for markupsafe

    building 'markupsafe._speedups' extension
    x86_64-linux-gnu-gcc -pthread -fno-strict-aliasing -DNDEBUG -g -fwrapv -O2 -Wall -
↪Wstrict-prototypes -fPIC -I/usr/include/python2.7 -c markupsafe/_speedups.c -o␣
↪build/temp.linux-x86_64-2.7/markupsafe/_speedups.o
    markupsafe/_speedups.c:12:20: fatal error: Python.h: No such file or directory
     #include <Python.h>
                        ^
```

```
    compilation terminated.
    ================================================================================
    WARNING: The C extension could not be compiled, speedups are not enabled.
    Failure information, if any, is above.
    Retrying the build without the C extension now.


    ================================================================================
    WARNING: The C extension could not be compiled, speedups are not enabled.
    Plain-Python installation succeeded.
    ================================================================================
Successfully installed jinja2 markupsafe
Cleaning up...
```

10.7 Installation - Maintenance

10.7.1 Remote Access

The RPi is not accessible from outside the LAN. For small maintenance purposes we may setup a reverse SSH tunnel such that we can access the RPi from a known system, 'remote', to which the RPi can connect via SSH. This way the RPi is only accessible from 'remote' and the communication is encrypted.

Setting up and maintaining a tunnel is best done with `autossh`. See more info at http://linuxaria.com/howto/permanent-ssh-tunnels-with-autossh

Steps as follows.

```
# install autossh
$ apt-get install autossh

# add user without shell on RPi and remote
useradd -m -s /bin/false autossh

# Generate keys op RPi
ssh-keygen -t rsa

# store on remote in /home/autossh/.ssh/authorized_keys

# add to /etc/rc.local on RPi/opt/bin/start-tunnels.sh with content
sleep 120
export AUTOSSH_LOGFILE=/var/log/autossh/autossh.log
export AUTOSSH_PIDFILE=/var/run/autossh/autossh.pid
# export AUTOSSH_POLL=60
# export AUTOSSH_FIRST_POLL=30
# export AUTOSSH_GATETIME=30
export AUTOSSH_DEBUG=1
rm -rf /var/run/autossh
mkdir  /var/run/autossh
chown autossh:autossh /var/run/autossh

su -s /bin/sh autossh -c
  'autossh -M 0 -q -f -N -o "ServerAliveInterval 60" -o "ServerAliveCountMax 3" -R
↪<localport>:localhost:22 autossh@<remote>'
```

Now we can login to the RPi, but only from 'remote' with `ssh <user>@localhost -p <localport>`.

10.7.2 Monitoring

As the RPi will be running headless and unattended within a LAN, it is of utmost importance that 'everything remains running'. To this end cronjobs are run with the following crontab file.

```
# Cronfile for keeping stuff alive on unattended Raspberry Pi
# Some bit crude like reboot, but effective mostly
# Author: Just van den Broecke <justb4@gmail.com>
#
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
SRC=/opt/geonovum/sospilot/git/src

# Do checks on weewx and network every N mins
*/6  * * * * $SRC/weewx/weewxcheck.sh
*/10 * * * * $SRC/raspberry/wificheck.sh
*/15 * * * * $SRC/raspberry/rpistatus.sh
0   4  * * *   shutdown -r +5
0   3  * * *   $SRC/weewx/backup-weewx.sh
```

The *weewx* daemon appears to be stopping randomly. Not clear why, but looks like this happens when there are network problems. To check and restart if needed the following script is run.

```
#! /bin/sh
# Author: Just van den Broecke <justb4@gmail.com>
# Restart weewx if not running.
#

WEEWX_HOME=/opt/weewx/weewxinst
WEEWX_BIN=$WEEWX_HOME/bin/weewxd

NPROC=`ps ax | grep $WEEWX_BIN | grep $NAME.pid | wc -l`
if [ $NPROC -gt 1 ]; then
    echo "weewx running multiple times on `date`! Attempting restart." >> /var/log/
↪weewxcheck.log
    /etc/init.d/weewx restart
elif [ $NPROC = 1 ]; then
    echo "Weewx is ok: $status"
else
    echo "weewx not running on `date`! Attempting restart." >> /var/log/weewxcheck.log
    /etc/init.d/weewx restart
fi
```

Restarts are also logged so we can see how often this happens.

The WiPi seems to have stability problems. This is a whole area of investigation on WIFI-stations/drivers/parameters etc, that could take days if not weeks... For now a script is run, that checks if the WIfi (*wlan0* device) is up or else restarts the interface/Wifi. For topic see http://www.raspberrypi.org/forums/viewtopic.php?t=54001&p=413095. See script at https://github.com/Geonovum/sospilot/blob/master/src/raspberry/wificheck.sh

```
#!/bin/bash
################################################################
# NOTE! THIS IS A MODIFIED VERSION OF THE ORIGINAL PROGRAM
# WRITTEN BY KEVIN REED.  TO GET THE ORIGINAL PROGRAM SEE
# THE URL BELOW:
#
# A Project of TNET Services, Inc
#
```

```
# Title:     WiFi_Check
# Author:    Kevin Reed (Dweeber)
#            dweeber.dweebs@gmail.com
#            Small adaptions by Just van den Broecke <justb4@gmail.com>
# Project:   Raspberry Pi Stuff
#
# Copyright: Copyright (c) 2012 Kevin Reed <kreed@tnet.com>
#            https://github.com/dweeber/WiFi_Check
#
# Purpose:
#
# Script checks to see if WiFi has a network IP and if not
# restart WiFi
#
# Uses a lock file which prevents the script from running more
# than one at a time.  If lockfile is old, it removes it
#
# Instructions:
#
# o Install where you want to run it from like /usr/local/bin
# o chmod 0755 /usr/local/bin/WiFi_Check
# o Add to crontab
#
# Run Every 5 mins - Seems like ever min is over kill unless
# this is a very common problem.  If once a min change */5 to *
# once every 2 mins */5 to */2 ...
#
# */5 * * * * /usr/local/bin/WiFi_Check
#
################################################################
# Settings
# Where and what you want to call the Lockfile
lockfile='/var/run/WiFi_Check.pid'

# Which Interface do you want to check/fix
wlan='wlan0'

# Which address do you want to ping to see if you can connect
pingip='194.109.6.93'

################################################################
echo
echo "Starting WiFi check for $wlan"
date
echo

# Check to see if there is a lock file
if [ -e $lockfile ]; then
    # A lockfile exists... Lets check to see if it is still valid
    pid=`cat $lockfile`
    if kill -0 &>1 > /dev/null $pid; then
        # Still Valid... lets let it be...
        #echo "Process still running, Lockfile valid"
        exit 1
    else
        # Old Lockfile, Remove it
        #echo "Old lockfile, Removing Lockfile"
        rm $lockfile
```

```
    fi
fi
# If we get here, set a lock file using our current PID#
#echo "Setting Lockfile"
echo $$ > $lockfile

# We can perform check
echo "Performing Network check for $wlan"
/bin/ping -c 2 -I $wlan $pingip > /dev/null 2> /dev/null
if [ $? -ge 1 ] ; then
    echo "Network connection down on `date`! Attempting reconnection." >> /var/log/
↪wificheck.log
    /sbin/ifdown $wlan
    sleep 10
    /sbin/ifup --force $wlan
else
    echo "Network is Okay"
fi


# Check is complete, Remove Lock file and exit
#echo "process is complete, removing lockfile"
rm $lockfile
exit 0

#################################################################
# End of Script
```

The overall RPi status is checked every 15 mins and the results posted to the VPS. In particular the network usage is monitored via *vnstat*. The script can be found at https://github.com/Geonovum/sospilot/blob/master/src/raspberry/rpistatus.sh and is as follows.

```
#! /bin/sh
# Author: Just van den Broecke <justb4@gmail.com>
# Status of RPi main resources. Post to VPS if possible.
#

log=/var/log/rpistatus.txt
remote=sadmin@sensors:/var/www/sensors.geonovum.nl/site/pi

echo "Status of `hostname` on date: `date`" > $log
uptime  >> $log 2>&1

echo "\n=== weewx ===" >> $log
/etc/init.d/weewx status >> $log
echo "archive stat: `ls -l /opt/weewx/weewxinst/archive`" >> $log 2>&1
echo "archive recs: `sqlite3 /opt/weewx/weewxinst/archive/weewx.sdb 'select count(*)␣
↪from archive'`" >> $log 2>&1

echo "\n=== restarts ===" >> $log
echo "weewx:" >> $log
wc -l /var/log/weewxcheck.log | cut -d'/' -f1 >> $log 2>&1
echo "\nWifi:" >> $log
wc -l /var/log/wificheck.log  | cut -d'/' -f1 >> $log 2>&1

echo "\n=== bandwidth (vnstat)" >> $log
vnstat >> $log 2>&1
```

```
echo "\n=== network (ifconfig)" >> $log
ifconfig >> $log 2>&1

echo "\n=== disk usage (df -h) ===" >> $log
df -h >> $log 2>&1

echo "\n=== memory (free -m)===" >> $log
free -m >> $log 2>&1

scp $log $remote
```

A typical result is as follows. See http://sensors.geonovum.nl/pi/rpistatus.txt.

```
Status of georasp on date: Thu Oct 23 13:11:31 CEST 2014
 13:11:31 up 16:39,  3 users,  load average: 0.18, 0.17, 0.16

=== weewx ===
Status of weewx weather system:: running.
archive stat: total 196
-rw-r--r-- 1 sadmin sadmin    189 Oct 20 13:02 one_archive_rec.txt
-rw-r--r-- 1 sadmin sadmin  43008 Oct 23 13:08 stats.sdb
-rw-r--r-- 1 sadmin sadmin 145408 Oct 23 13:08 weewx.sdb
archive recs: 850

=== restarts ===
weewx:
0

Wifi:
0

=== bandwidth (vnstat)

                          rx      /      tx      /     total    /   estimated
 eth0: Not enough data available yet.
 wlan0:
       Oct '14      1.32 MiB  /    2.34 MiB  /    3.66 MiB  /    3.00 MiB
         today      1.32 MiB  /    2.34 MiB  /    3.66 MiB  /       4 MiB


=== network (ifconfig)
eth0      Link encap:Ethernet  HWaddr b8:27:eb:12:6a:ef
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:16829 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16829 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2825670 (2.6 MiB)  TX bytes:2825670 (2.6 MiB)

wlan0     Link encap:Ethernet  HWaddr 00:c1:41:06:0f:42
          inet addr:10.0.0.241  Bcast:10.255.255.255  Mask:255.0.0.0
```

```
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:52305 errors:0 dropped:0 overruns:0 frame:0
          TX packets:31157 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9209831 (8.7 MiB)  TX bytes:11348504 (10.8 MiB)


=== disk usage (df -h) ===
Filesystem      Size  Used Avail Use% Mounted on
rootfs           28G  3.0G   24G  12% /
/dev/root        28G  3.0G   24G  12% /
devtmpfs        215M     0  215M   0% /dev
tmpfs            44M  276K   44M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs            88M     0   88M   0% /run/shm
/dev/mmcblk0p5   60M  9.6M   50M  17% /boot

=== memory (free -m)===
             total       used       free     shared    buffers     cached
Mem:           437        224        212          0         33        136
-/+ buffers/cache:         55        382
Swap:           99          0         99
```

10.7.3 Backup

weewx db backup

Only weewx.sdb the SQLite DB has to be backed up. The stats file will always be regenerated.

See script: https://github.com/Geonovum/sospilot/blob/master/src/weewx/backup-weewx.sh added to root cronfile

Local dir: */opt/weewx/weewxinst/backup*

Backed up to sensors VPS: */home/sadmin/weewx-backup* dir.

Pi SD Card Disk Backup

Follow instructions on http://sysmatt.blogspot.nl/2014/08/backup-restore-customize-and-clone-your.html to make a restorable .tar.gz (i.s.o. dd diskclone).

```
$ apt-get install dosfstools
# was already installed
# attach USB SDcardreader with 16GB SD Card
$ dmesg
[39798.700351] sd 0:0:0:1: [sdb] 31586304 512-byte logical blocks: (16.1 GB/15.0 GiB)
[39798.700855] sd 0:0:0:1: [sdb] Write Protect is off
[39798.700888] sd 0:0:0:1: [sdb] Mode Sense: 03 00 00 00
[39798.701388] sd 0:0:0:1: [sdb] No Caching mode page found
[39798.701451] sd 0:0:0:1: [sdb] Assuming drive cache: write through
[39798.706669] sd 0:0:0:2: [sdc] Attached SCSI removable disk
[39798.707165] sd 0:0:0:2: Attached scsi generic sg2 type 0
[39798.709292] sd 0:0:0:1: [sdb] No Caching mode page found
[39798.709355] sd 0:0:0:1: [sdb] Assuming drive cache: write through
[39798.710838]  sdb: sdb1
[39798.714637] sd 0:0:0:1: [sdb] No Caching mode page found
[39798.714677] sd 0:0:0:1: [sdb] Assuming drive cache: write through
[39798.714701] sd 0:0:0:1: [sdb] Attached SCSI removable disk
[39798.715493] scsi 0:0:0:3: Direct-Access     Generic  SM/xD-Picture    0.00 PQ: 0␣
↪ANSI: 2
```

```
[39798.718181] sd 0:0:0:3: [sdd] Attached SCSI removable disk
[39798.724978] sd 0:0:0:3: Attached scsi generic sg3 type 0

root@georasp:~# df
Filesystem     1K-blocks    Used Available Use% Mounted on
rootfs          29077488 3081180  24496200  12% /
/dev/root       29077488 3081180  24496200  12% /
devtmpfs          219764       0    219764   0% /dev
tmpfs              44788     280     44508   1% /run
tmpfs               5120       0      5120   0% /run/lock
tmpfs              89560       0     89560   0% /run/shm
/dev/mmcblk0p5     60479    9779     50700  17% /boot

root@georasp:~# parted -l
Model: Generic SD/MMC (scsi)
Disk /dev/sdb: 16.2GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type     File system  Flags
 1      4194kB  16.2GB  16.2GB  primary  fat32        lba


Model: SD USD (sd/mmc)
Disk /dev/mmcblk0: 31.3GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type      File system  Flags
 1      4194kB  828MB   824MB   primary   fat32        lba
 2      830MB   31.3GB  30.5GB  extended
 5      835MB   898MB   62.9MB  logical   fat32        lba
 6      902MB   31.3GB  30.4GB  logical   ext4
 3      31.3GB  31.3GB  33.6MB  primary   ext4


root@georasp:~# parted /dev/sdb
GNU Parted 2.3
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mklabel msdos
Warning: The existing disk label on /dev/sdb will be destroyed and all data on this␣
↪disk will be lost. Do you want to continue?
Yes/No? Yes
(parted) mkpart primary fat16 1MiB 64MB
(parted) mkpart primary ext4 64MB -1s
(parted) print
Model: Generic SD/MMC (scsi)
Disk /dev/sdb: 16.2GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  64.0MB  62.9MB  primary               lba
 2      64.0MB  16.2GB  16.1GB  primary

(parted) quit
Information: You may need to update /etc/fstab.
```

```
root@georasp:~# parted -l
Model: Generic SD/MMC (scsi)
Disk /dev/sdb: 16.2GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  64.0MB  62.9MB  primary               lba
 2      64.0MB  16.2GB  16.1GB  primary


Model: SD USD (sd/mmc)
Disk /dev/mmcblk0: 31.3GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type      File system  Flags
 1      4194kB  828MB   824MB   primary   fat32        lba
 2      830MB   31.3GB  30.5GB  extended
 5      835MB   898MB   62.9MB  logical   fat32        lba
 6      902MB   31.3GB  30.4GB  logical   ext4
 3      31.3GB  31.3GB  33.6MB  primary   ext4


root@georasp:~# mkfs.vfat /dev/sdb1
mkfs.vfat 3.0.13 (30 Jun 2012)

root@georasp:~# mkfs.ext4 -j  /dev/sdb2
mke2fs 1.42.5 (29-Jul-2012)
warning: 512 blocks unused.

Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
984960 inodes, 3932160 blocks
196633 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4026531840
120 block groups
32768 blocks per group, 32768 fragments per group
8208 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

root@georasp:~# mkdir /tmp/newpi
root@georasp:~# mount /dev/sdb2 /tmp/newpi
root@georasp:~# mkdir /tmp/newpi/boot
root@georasp:~# mount /dev/sdb1 /tmp/newpi/boot

root@georasp:~# df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
rootfs           28G  3.0G   24G  12% /
/dev/root        28G  3.0G   24G  12% /
devtmpfs        215M     0  215M   0% /dev
tmpfs            44M  284K   44M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs            88M     0   88M   0% /run/shm
/dev/mmcblk0p5   60M  9.6M   50M  17% /boot
/dev/sdb2        15G   38M   14G   1% /tmp/newpi
/dev/sdb1        60M     0   60M   0% /tmp/newpi/boot
root@georasp:~# crontab -l
# Cronfile for keeping stuff alive on unattended Raspberry Pi
# Some bit crude like reboot, but effective mostly
# Author: Just van den Broecke <justb4@gmail.com>
#
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
SRC=/opt/geonovum/sospilot/git/src

# Do checks on weewx and network every N mins
*/6  * * * * $SRC/weewx/weewxcheck.sh
*/10 * * * * $SRC/raspberry/wificheck.sh
*/15 * * * * $SRC/raspberry/rpistatus.sh
0   4  * * *   shutdown -r +5
root@georasp:~# crontab -r
root@georasp:~# ls
Desktop
root@georasp:~# /etc/init.d/weewx stop
[ ok ] Stopping weewx weather system: weewx.
root@georasp:~#

# get the backup tools
wget -O sysmatt-rpi-tools.zip  https://github.com/sysmatt-industries/sysmatt-rpi-
↪tools/archive/master.zip

# do rsync
$ rsync -av --one-file-system / /boot /tmp/newpi/

# ...wait long time, many files....

root@georasp:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
rootfs           28G  3.0G   24G  12% /
/dev/root        28G  3.0G   24G  12% /
devtmpfs        215M     0  215M   0% /dev
tmpfs            44M  284K   44M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs            88M     0   88M   0% /run/shm
/dev/mmcblk0p5   60M  9.6M   50M  17% /boot
/dev/sdb2        15G  3.0G   11G  22% /tmp/newpi
/dev/sdb1        60M  9.6M   51M  16% /tmp/newpi/boot

# NOOBS stuff repair
$ edit /tmp/newpi/boot/cmdline.txt
# root=/dev/mmcblk0p6 must become root=/dev/mmcblk0p2
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2␣
↪rootfstype=ext4 elevator=deadline rootwait
```

```
# fstab
# in /tmp/newpi/etc/fstab
proc            /proc           proc    defaults          0       0
/dev/mmcblk0p5  /boot           vfat    defaults          0       2
/dev/mmcblk0p6  /               ext4    defaults,noatime  0       1
# should become
proc            /proc           proc    defaults          0       0
/dev/mmcblk0p1  /boot           vfat    defaults          0       2
/dev/mmcblk0p   /               ext4    defaults,noatime  0       1
```

Overdracht. */etc/network/interfaces* aanpassen.

```
# SD-card in USBReader mounten:
mkdir /tmp/oldpi
mount /dev/sdb6 /tmp/oldpi
/tmp/oldpi/etc/network/interfaces
```

10.7.4 Kiosk Mode

The RPi will be conencted to a TV-screen in a public room at Geonovum (kitchen). As to have something interesting to see, the Rpi will be put in a "kiosk" mode. This can be achieved quite simply using a webbrowser and a website. As there is no user interaction possible via mouse/keyboard this simple setup suffices.

The website will be a continuous change of pages/URLs. For this a simple JavaScript app is made that changes pages into an *iframe*. See https://github.com/Geonovum/sospilot/blob/master/www/kiosk/index.html. This app will be published to http://sensors.geonovum.nl/kiosk thus can be updated at all times without needing access to the RPi. If bandwidth becomes an issue we may move the app to the RPi later.

In order to always start a browser, X-Windows needs to be started at boottime. This is described here: http://www.opentechguides.com/how-to/article/raspberry-pi/5/raspberry-pi-auto-start.html and here a complete tutorial https://www.danpurdy.co.uk/web-development/raspberry-pi-kiosk-screen-tutorial

In order to start the browser (Chromium) the following is useful: https://lokir.wordpress.com/2012/09/16/raspberry-pi-kiosk-mode-with-chromium and http://askubuntu.com/questions/487488/how-to-open-chromium-in-full-screen-kiosk-mode-in-minimal-windows-manager-enviro

The following was executed.

```
# install chromium and tools
apt-get install chromium x11-xserver-utils unclutter

# enable X desktop at boot
raspi-config

# choose option3

# edit /etc/xdg/lxsession/LXDE/autostart as follows
@lxpanel --profile LXDE
@pcmanfm --desktop --profile LXDE
# @xscreensaver -no-splash
@xset s off
@xset -dpms
@xset s noblank
@chromium --noerrdialogs --kiosk http://sensors.geonovum.nl/kiosk

apt-get install tightvncserver
tightvncserver
```

```
# start als user pi http://www.penguintutor.com/linux/tightvnc
su pi -c '/usr/bin/tightvncserver :1'
```

10.7.5 Links

- http://garethhowell.com/wp/connect-raspberry-pi-3g-network
- http://www.jamesrobertson.eu/blog/2014/jun/24/setting-up-a-huawei-e3131-to-work-with-a.html
- http://christianscode.blogspot.nl/2012/11/python-huawei-e3131-library.html
- Reverse tunneling to access the Pi from outside: http://www.thirdway.ch/En/projects/raspberry_pi_3g/index.php
- Use *autossh* to maintain tunnel: http://unix.stackexchange.com/questions/133863/permanent-background-ssh-connection-to-create-reverse-tunnel-what-is-correct-wa
- http://ccgi.peterhurn.plus.com/wordpress/raspberry-pi-weather-station-installation-notes/

CHAPTER 11

Server Inrichting

Hier staat de inrichting beschreven voor de Linux Server (Ubuntu) met FOSS geo-toepassingen. Dit betreft een Ubuntu-server waarop allerlei basis en geo-gerelateerde packages zijn toegevoegd.

11.1 Conventies

Bij het inrichten van een server is het zeer belangrijk om vaste conventies aan te houden. De algemene conventies staan hier beschreven.

11.1.1 Directories

Het voordeel van een Linux systeem is dat er altijd vaste directories zijn waarbij een heldere scheiding is tussen programma's, data, bibliotheken, configuratie etc. Daarnaast is het goed om voor additionele directories een vaste conventie te hebben. Deze is als volgt:

- `/opt` additionele, handmatig geinstalleerde software
- `/opt/<leverancier of product>/<product-versie>` installatie dirs voor additionele, handmatig geinstalleerde software
- `/opt/download` downloads voor additionele, handmatig geinstalleerde software
- `/opt/bin` eigen additionele shell scripts (bijv. backup)
- `/var/sensors` alle eigen (geo)data, configuraties, tilecaches, packed backups to be transfered
- `/var/www` alle web applicaties/sites
- `/var/www/sensors.geonovum.nl/site` sensors website (platte HTML)
- `/var/www/sensors.geonovum.nl/webapps` alle Tomcat applicaties: GeoServer, GeoWebCache, deegree etc
- `/var/www/sensors.geonovum.nl/cgi-bin` proxy en python scripts
- `/home/sadmin` home dir beheer account

Onder /opt/<leverancier of product> kan vaak ook een dynamic link staan naar de laatste versie van een product, bijvoorbeeld /opt/nlextract/active is een dynlink naar een versie van NLExtract bijvoorbeeld naar /opt/nlextract/1.1.5.

Voeg ook toe in /etc/profile zodat de scripts in /opt/bin gevonden worden

```
export PATH=/opt/bin:${PATH}
```

Omdat de meeste toepassingen gebruik maken van Apache Virtual Hosts met een prefix op sensors, zoals bijvoorbeeld inspire.sensors is hier ook een conventie op zijn plaats:

- /etc/apache2/sites-available/sensors.geonovum.nl.conf bevat de Apache configuratie
- /var/www/sensors.geonovum.nl bevat de website content (HTML etc)

11.1.2 Systeem

Aangeschaft 7 mei 2014.

```
[vps44500 systeeminformatie]
IPv4 address..: 185.21.189.59
IPv6 address..: 2a02:348:a1:bd3b::1
Hostname......: vps44500.public.cloudvps.com
Image.........: ubuntu1404-bare.conf
RAM...........: 8192MB
Disk..........: 240 GB
Cpus..........: 3
```

11.1.3 Host

Host is sensors.geonovum.nl IP sensors.geonovum.nl has address 185.21.189.59

In /etc/hosts shorthand sensors toegevoegd, voor o.a. vhosts apache.

11.1.4 Accounts

Elke server krijgt 2 standaard accounts: root ("root admin") en sadmin ("sensors admin"). NB de account root wordt (door Ubuntu) nooit aangemaakt als login account!

Het beheer-account root heeft root-rechten.

Het account sadmin heeft ook wat rechten maar minder. Dit account heeft lees/schrijfrechten op directories voor custom installaties (zie onder), de websites onder /var/www behalve partner-projecten en data/configuratie directories onder /var/sensors.

```
$ id sadmin
uid=1001(sadmin) gid=1002(sadmin) groups=1002(sadmin)

$ id root
uid=0(root) gid=0(root) groups=0(root)
```

11.1.5 Software Installatie

Alle verdere software is via de Ubuntu/Debian Advanced Packaging Tool (APT) gedaan. Hiermee is op zeer eenvoudige wijze niet alleen alle software, inclusief de meeste GIS tools gemakkelijk te installeren, maar ook up-to-date te houden. Bijvoorbeeld een complete Java installatie gaat met : `apt-get install sun-java6-jdk`. APT wordt altijd door het `root` account (met root via sudo of sudo -i) uitgevoerd.

Alleen in een uiterst geval waarbij een software product niet in het APT systeem zit of niet in een gewenste versie is een handmatige ("custom") installatie gedaan. Hierbij is de volgende conventie aangehouden: custom installaties worden door het account `root`.

11.1.6 Backup

Systeem backup met deze tool http://www.cloudvps.com/community/knowledge-base/cloudvps-backup-script-introduction:

```
[CloudVPS Backup Account]
Server........: backup-030.cloudvps.com
Username......: vps44500
Quota.........: 20 GB
Protocols.....: SFTP, FTP and rsync over SSH
```

De configuratie kan evt geregeld worden met: `/root/cloudvps-backup-installer.sh` Ik heb nu de email recipients aangepast naar

Er draait elke nacht (om 01:41 , zie crontab -l) een backup

```
$ crontab -l
41 1 * * * /usr/local/bin/cloudvpsbackup > /dev/null 2>&1
```

Config onder `/etc/cloudvps/`, maar edit deze via `/root/cloudvps-backup-installer.sh`. Logfiles staan onder: `/var/log/backups/`.

11.1.7 Disk Gebruik

Op 25.5.14, na install alle support tools en server software, zonder data.

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1      237G  4.5G  220G   2% /
none            4.0K     0  4.0K   0% /sys/fs/cgroup
udev            3.9G  4.0K  3.9G   1% /dev
tmpfs           788M  204K  787M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            3.9G     0  3.9G   0% /run/shm
none            100M     0  100M   0% /run/user
```

11.1.8 Java Monitor

Zie http://java-monitor.com. Hiermee wordt voortdurend de status/gezondheid van de Tomcat Java server gemonitored. Indien er een probleem is wordt email gestuurd.

```
# download probe
# unpack in /opt/java-monitor.com
# drop war in /var/www/sensors.geonovum.nl/webapps
```

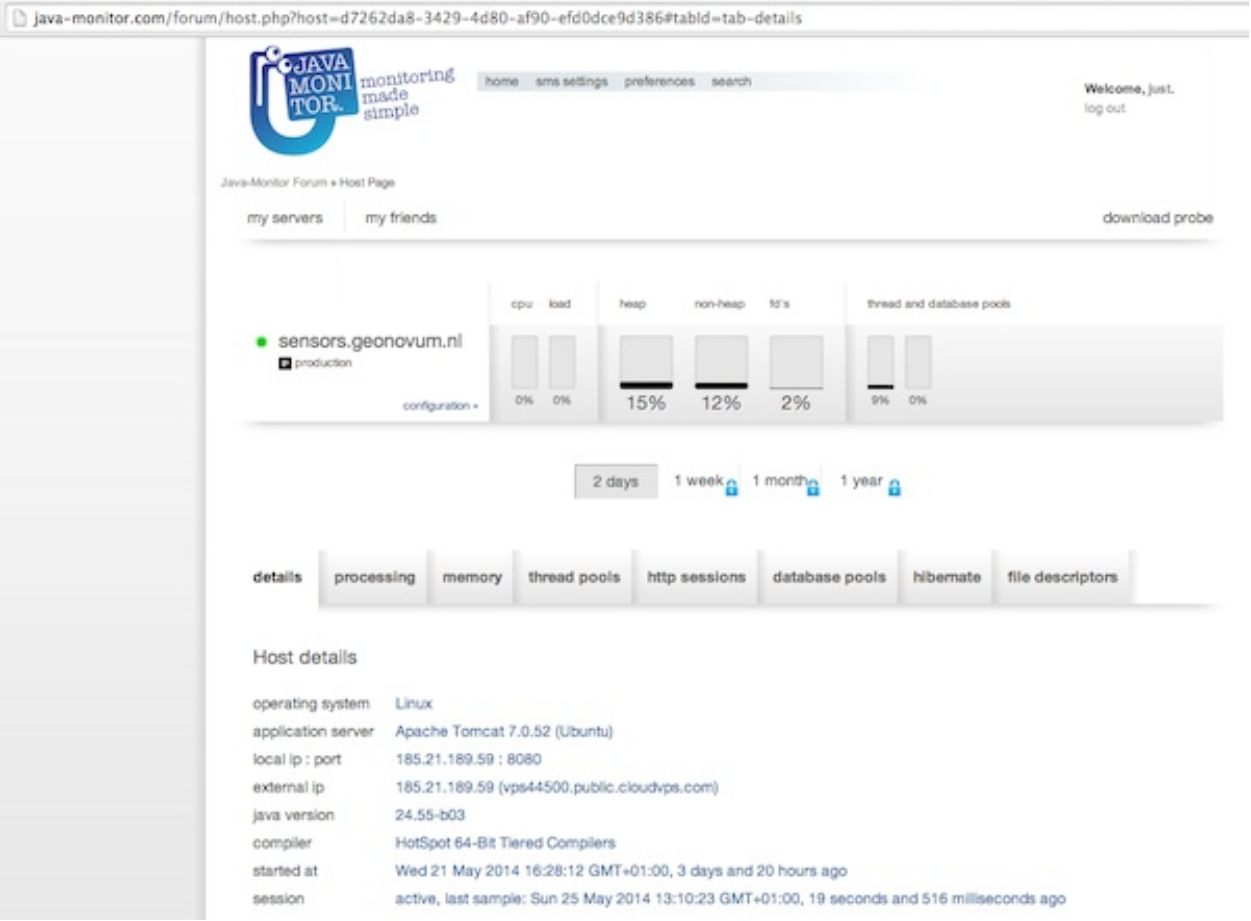


Fig. 11.1: *Figure 1 - Java Monitor*

11.2 Server Software - Algemeen

Hieronder standaard packages.

11.2.1 Apache Web Server

De standaard Apache web server (versie 2).

```
# installatie apache package (default installs mpm worker)
apt-get install apache2
apt-get install apache2-utils

# in /etc/apache2/apache2.conf zet
# ServerName sensors
```

Zet servertokens to Minimal in `/etc/apache2/conf-available/security.conf`

Website: `/var/www/sensors.geonovum.nl`

```
$ mkdir /var/www/sensors.geonovum.nl
$ mkdir /var/www/sensors.geonovum.nl/site     HTML site
$ mkdir /var/www/sensors.geonovum.nl/cgi-bin  proxy scripts etc
$ mkdir /var/www/sensors.geonovum.nl/admin    admin site
$ mkdir /var/www/sensors.geonovum.nl/webapps  java servers (.war deploy)
```

De uiteindelijke config in `/etc/apache2/sites-available/sensors.geonovum.nl.conf`

```
<VirtualHost sensors:80>
    ServerName sensors.geonovum.nl

    DocumentRoot /var/www/sensors.geonovum.nl/site

   ScriptAlias /cgi-bin/ /var/www/sensors.geonovum.nl/cgi-bin/
    <Directory "/var/www/sensors.geonovum.nl/cgi-bin">
         AllowOverride None
         Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
         Order allow,deny
         Allow from all
    </Directory>

    ServerAdmin just@justobjects.nl

    DirectoryIndex index.html index.php index.jsp

    Alias /sadm "/var/www/sensors.geonovum.nl/sadm"
   <Directory "/var/www/sensors.geonovum.nl/sadm">
        Options Indexes FollowSymlinks MultiViews
        AuthType Basic
        AuthName "Sensors Admin"
        AuthUserFile /etc/apache2/.htpasswd
        Require user sadmin

        AllowOverride None
        Order allow,deny
        Allow from all

    </Directory>

    <Location /tomcat/examples>
      ProxyPass ajp://sensors:8009/examples
      ProxyPassReverse http://sensors/examples
    </Location>

    <Location /gs>
      ProxyPass ajp://sensors:8009/gs
      ProxyPassReverse http://sensors/gs
    </Location>

    <Location /sos>
      ProxyPass ajp://sensors:8009/sos
      ProxyPassReverse http://sensors/sos
    </Location>

    LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"" combined
```

```
    CustomLog /var/log/apache2/sensors.geonovum.nl-access.log combined
    ErrorLog /var/log/apache2/sensors.geonovum.nl-error.log

</VirtualHost>
```

Site aktiveren met *a2ensite sensors.geonovum.nl*.

Dit wordt de beheer site http://sensors.geonovum.nl/sadm. Wachtwoord zetten met:

```
htpasswd -c /etc/apache2/.htpasswd sadmin
```

Maak een hidden link voor website administratie en beveilig deze met een htaccess paswoord.

11.2.2 Java

Java van Oracle installeren. Niet OpenJDK (ivm GeoServer problemen). Kan/mag niet via Ubuntu maar via PPA: https://launchpad.net/~webupd8team/+archive/java. Die download weer van Oracle...

Zie: http://www.webupd8.org/2012/01/install-oracle-java-jdk-7-in-ubuntu-via.html

Stappen.

```
$ add-apt-repository ppa:webupd8team/java
Oracle Java (JDK) Installer (automatically downloads and installs Oracle JDK6 / JDK7 /
↪ JDK8). There are no actual Java files in this PPA.

More info:
- for Oracle Java 7: http://www.webupd8.org/2012/01/install-oracle-java-jdk-7-in-
↪ubuntu-via.html
- for Oracle Java 8: http://www.webupd8.org/2012/09/install-oracle-java-8-in-ubuntu-
↪via-ppa.html

Debian installation instructions: http://www.webupd8.org/2012/06/how-to-install-
↪oracle-java-7-in-debian.html
More info: https://launchpad.net/~webupd8team/+archive/java
Press [ENTER] to continue or ctrl-c to cancel adding it

gpg: keyring `/tmp/tmp09u8e2c5/secring.gpg' created
gpg: keyring `/tmp/tmp09u8e2c5/pubring.gpg' created
gpg: requesting key EEA14886 from hkp server keyserver.ubuntu.com
gpg: /tmp/tmp09u8e2c5/trustdb.gpg: trustdb created
gpg: key EEA14886: public key "Launchpad VLC" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
OK

    $ apt-get update

    $ apt-get install oracle-java7-installer
```

Resultaat

```
$ java -version
java version "1.7.0_55"
Java(TM) SE Runtime Environment (build 1.7.0_55-b13)
Java HotSpot(TM) 64-Bit Server VM (build 24.55-b03, mixed mode)
```

11.2.3 Tomcat

Zie https://help.ubuntu.com/13.10/serverguide/tomcat.html.

Installeren

```
$ apt-get install tomcat7

# check
$ lynx localhost:8080
```

Logs in `/var/log/tomcat7/`. Config in `/etc/tomcat7`, met name `/etc/tomcat7/server.xml`.

Verder, documentatie, manager en voorbeelden.

```
    $ apt-get install tomcat7-docs
$ apt-get install tomcat7-admin
$ apt-get install tomcat7-examples
```

Schrijfrechten in `/etc/tomcat7`.

```
    $ chgrp -R tomcat7 /etc/tomcat7
    $ chmod -R g+w /etc/tomcat7
    $ ls -l /etc/tomcat7
drwxrwxr-x 3 root tomcat7   4096 May  9 13:47 Catalina
-rw-rw-r-- 1 root tomcat7   6426 Feb 27 13:18 catalina.properties
-rw-rw-r-- 1 root tomcat7   1394 Jan 25 21:13 context.xml
-rw-rw-r-- 1 root tomcat7   2370 Feb 21 07:11 logging.properties
drwxrwxr-x 2 root tomcat7   4096 May  9 13:48 policy.d
-rw-rw-r-- 1 root tomcat7   6500 Feb 27 13:18 server.xml
-rw-rw---- 1 root tomcat7   1530 Jan 25 21:13 tomcat-users.xml
-rw-rw-r-- 1 root tomcat7 162905 Jan 25 21:13 web.xmlusers.xml
-rw-rw-r-- 1 root tomcat7 162905 Oct 26  2012 web.xml
```

Manager user aanmaken (sadmin).

Access to the manager application is protected by default: you need to define a user with the role “manager-gui” in `/etc/tomcat7/tomcat-users.xml` before you can access it.

```
<user username="sadmin" password="*" roles="manager-gui,admin-gui"/>
```

The second one is the host-manager webapp, which you can access by default at http://sensors.geonovum.nl:8080/host-manager. It can be used to create virtual hosts dynamically.

Access to the host-manager application is also protected by default: you need to define a user with the role “admin-gui” in `/etc/tomcat7/tomcat-users.xml` before you can access it.

Koppelen van Tomcat met de Apache server gaat via `mod_proxt_ajp` een standaard onderdeel van Apache. Enablen van deze module (in de Host):

```
a2enmod proxy_ajp
```

In `/etc/tomcat7/server.xml` AJP enablen.

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

In Apache configuraties moet dan elke request voor de Tomcat webapp via de AJP Proxy naar Tomcat geleid worden. Een voorbeeld is hier voor Tomcat voorbeelden binnen vanaf de Host naar de base Geoserver, http://sensors.geonovum.nl

```
<Location /tomcat/examples>
  ProxyPass ajp://sensors:8009/examples
  ProxyPassReverse http://sensors/examples
</Location>
```

En users aan tomcat groep toevoegen.

```
usermod -aG tomcat7 sadmin
usermod -aG tomcat7 root
```

Zet JAVA_OPTS in `/etc/init.d/tomcat7`.

```
JAVA_OPTS="-Djava.awt.headless=true -server -Xmx2048M -Xms512M -
↪XX:SoftRefLRUPolicyMSPerMB=36000
 -XX:MaxPermSize=512m -XX:+UseParallelGC"
```

Later gezet naar:

```
JAVA_OPTS="-Djava.awt.headless=true -server -Xmx3072M -Xms512M -
↪XX:SoftRefLRUPolicyMSPerMB=36000
-XX:MaxPermSize=1024m -XX:+UseParallelGC"
```

NB JAVA_OPTS op standaard plek zetten `/etc/init.d/tomcat7` (in 'if' statement) werkte niet!! Gezet na execute $DEFAULT, dan pakt ie wel op!!

testen: http://sensors.geonovum.nl/tomcat/examples/jsp/jsp2/el/basic-arithmetic.jsp, OK!

Virtual hosts vvia Apache en koppelen aan domein. In `/etc/tomcat7/server.xml` voeg toe.

```
<Host name="sensors.geonovum.nl"  appBase="/var/www/sensors.geonovum.nl/webapps"
                unpackWARs="true" autoDeploy="true">
 <Alias>sensors</Alias>

<!-- Access log processes all example.
     Documentation at: /docs/config/valve.html
     Note: The pattern used is equivalent to using pattern="common" -->
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
   prefix="sensors_access_log." suffix=".txt"
   pattern="%h %l %u %t &amp;quot;%r&amp;quot; %s %b" />

</Host>
```

Toevoegen in `/etc/apache2/sites-available/sensors.geonovum.nl.conf`

```
<Location /gs>
  ProxyPass ajp://sensors:8009/gs
  ProxyPassReverse http://sensors/gs
</Location>

<Location /sos>
  ProxyPass ajp://sensors:8009/sos
  ProxyPassReverse http://sensors.geonovum.nl/sos
</Location>
```

Logfiles volgen van Tomcat: `tail -f /var/log/tomcat7/catalina.out`.

11.3 Server Software - Geo

11.3.1 Extra Package Sources

Ubuntu GIS, https://wiki.ubuntu.com/UbuntuGIS. Voor laatste versies belangrijkste FOSS geo-tools.

```
    apt-get install python-software-properties
    add-apt-repository ppa:ubuntugis/ubuntugis-unstable
    add-apt-repository ppa:kakrueger/openstreetmap
apt-get update
```

Helaas nog niet beschikbaar voor Ubuntu 14.04 (Trusty) !!!

11.3.2 PostgreSQL en PostGIS

PostgreSQL is een OS relationele database (RDBMS). PostGIS is een extentie die van PostgreSQL een ruimtelijke (spatial) database maakt. Installatie gaat via APT

```
$ apt-get install postgis postgresql postgresql-contrib
 Setting up postgresql (9.3+154) ...
 Setting up postgresql-contrib-9.3 (9.3.4-1) ...
 Setting up postgresql-contrib (9.3+154) ...
 Setting up odbcinst (2.2.14p2-5ubuntu5) ...
 Setting up odbcinst1debian2:amd64 (2.2.14p2-5ubuntu5) ...
 Setting up libgdal1h (1.10.1+dfsg-5ubuntu1) ...
 Setting up postgis (2.1.2+dfsg-2) ...

# create users (bijv oase) with this pattern
su postgres
createuser sensors
psql template1
alter user sensors password '***';
\q
```

Server Instrumentation, met admin pack.

```
$ sudo -u postgres psql
psql (9.1.10)
Type "help" for help.

postgres=# CREATE EXTENSION adminpack;
CREATE EXTENSION
```

Installatie controleren met

```
psql -h localhost -U postgres template1

 $ pg_lsclusters
 Ver Cluster Port Status Owner    Data directory              Log file
 9.3 main    5432 online postgres /var/lib/postgresql/9.3/main /var/log/postgresql/
↪postgresql-9.3-main.log
```

Enablen locale connecties in `/etc/postgresql/9.3/main/pg_hba.conf`.

```
# Database administrative login by Unix domain socket
local   all             postgres                                md5
```

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     md5
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
```

Evt postgres wachtwoord resetten: http://stackoverflow.com/questions/12720967/is-possible-to-check-or-change-postgresql-user-password

Beheer van PostgreSQL via web met `phppgadmin`.

```
$ apt-get install phppgadmin
# Get:1 http://us.archive.ubuntu.com/ubuntu/ saucy/main php5-pgsql amd64 5.5.3+dfsg-
↪1ubuntu2 [65.3 kB]
# Get:2 http://us.archive.ubuntu.com/ubuntu/ saucy/main libjs-jquery all 1.7.2+dfsg-
↪2ubuntu1 [78.8 kB]
# # Get:3 http://us.archive.ubuntu.com/ubuntu/ saucy/main postgresql-doc-9.1 all 9.1.
↪10-1 [1,607 kB]
# Get:4 http://us.archive.ubuntu.com/ubuntu/ saucy/main postgresql-doc all 9.
↪3+146really9.1+148 [6,416 B]
# Get:5 http://us.archive.ubuntu.com/ubuntu/ saucy/universe phppgadmin all 5.1-1 [704␣
↪kB]

# restart apache
ln -s /usr/share/phppgadmin /var/www/default/<geheim> (onder admin)

 # edit /etc/phppgadmin/config.inc.php
 // If extra login security is true, then logins via phpPgAdmin with no
 // password or certain usernames (pgsql, postgres, root, administrator)
 // will be denied. Only set this false once you have read the FAQ and
 // understand how to change PostgreSQL's pg_hba.conf to enable
 // passworded local connections.
 $conf['extra_login_security'] = false;
```

Postgis en template opzetten. Ook dit nodig om Postgis extension aan te maken.

```
$ apt-get -s install postgresql-9.1-postgis-2.1
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  postgresql-9.1-postgis-scripts
The following NEW packages will be installed:
  postgresql-9.1-postgis-2.1 postgresql-9.1-postgis-scripts
0 upgraded, 2 newly installed, 0 to remove and 3 not upgraded.
Inst postgresql-9.1-postgis-scripts (2.1.0-5~saucy1 ubuntugis-unstable:13.10/saucy␣
↪[all])
Inst postgresql-9.1-postgis-2.1 (2.1.0-5~saucy1 ubuntugis-unstable:13.10/saucy␣
↪[amd64])
Conf postgresql-9.1-postgis-scripts (2.1.0-5~saucy1 ubuntugis-unstable:13.10/saucy␣
↪[all])
Conf postgresql-9.1-postgis-2.1 (2.1.0-5~saucy1 ubuntugis-unstable:13.10/saucy␣
↪[amd64])
```

Anders krijg je op `CREATE EXTENSION postgis` dit

```
ERROR: could not open extension control file "/usr/share/postgresql/9.1/extension/
↪postgis.control": No such file or directory
```

Template DB''postgis2'' opzetten.

```
    su postgres
    createdb postgis2
psql -h localhost postgis2
postgis2=# CREATE EXTENSION postgis;
# CREATE EXTENSION
postgis2=# CREATE EXTENSION postgis_topology;
# CREATE EXTENSION
```

Ook in PostGIS staat goede RD geconfigureerd (`towgs84` ontbreekt dit keer niet!!).

```
+proj=sterea +lat_0=52.15616055555555 +lon_0=5.38763888888889
+k=0.9999079 +x_0=155000 +y_0=463000 +ellps=bessel
+towgs84=565.417,50.3319,465.552,-0.398957,0.343988,-1.8774,4.0725
+units=m +no_defs
```

Ook 900913 (Google) is goed.

> De database `postgis2` zal steeds als PostgrSQL `template` worden gebruikt bij het aanmaken van specifieke database zoals `georzlab` en `inspire`. Door de update in `spatial_ref_sys` is dan de goede RD configuratie, maar het is goed om altijd te controleren.

Test met dump inlezen. Haal dump met.

```
wget http://data.nlextract.nl/opentopo/workshop/geodata/bag-jan13-gooi-eo.backup

    createdb -U postgres  -T postgis2 bag
pg_restore -d bag -U postgres bag-jan13-gooi-eo.backup
```

Lijkt goed te gaan. Alleen metatabellen (onder VIEWs `geometry_columns`) nakijken. Bijv.

```
select ST_AsEWKT(geopunt)  from bag_test.adres limit 3;
                                st_asewkt
-----------------------------------------
 SRID=28992;POINT(119657.88 480340.86 0)
 SRID=28992;POINT(119846.04 478236.32 0)
 SRID=28992;POINT(118514.126 476795.241 0)
```

11.3.3 GeoServer

GeoServer via Apache-AJP-Tomcat.

- .war van GS-download onder `/opt/geoserver/<versie>` als gs.war
- bijv `/opt/geoserver/2.5.0/gs.war`
- eigen config in `/var/sensors/config/geoserver`
- in `/etc/init.d/tomcat7:` `export GEOSERVER_DATA_DIR=/var/sensors/config/geoserver`
- deploy door `cp /opt/geoserver/2.5.0/gs.war /var/www/sensors.geonovum.nl/webapps`

- `/gs` is gemakkelijker als korte naam/URL
- de URL wordt `http://sensors.geonovum.nl/gs/<evt workspace>`

Om permissie-problemen te voorkomen doen we.

```
chown -R tomcat7:tomcat7 /var/www/sensors.geonovum.nl/webapps
chown -R tomcat7:tomcat7 /var/sensors/config/geoserver
```

GeoServer was upgraded to 2.8.0 on oct 10, 2015.

11.3.4 Sensorweb SOS Server

"The OGC Sensor Observation Service aggregates readings from live, in-situ and remote sensors. The service provides an interface to make sensors and sensor data archives accessible via an interoperable web based interface."

Installatie van de INSPIRE version of SOS server from 52North.

From Simon Jirka 19.05.14: "We have now packaged together a new installation file of the INSPIRE SOS together with the REST interface:

http://52north.org/files/sensorweb/INSPIRE/52N-SOS-INSPIRE-with-RestAPI_20140519.zip

The ZIP archive also contains a short README file with a link to the installation guide and some additional information on the INSPIRE SOS. "

Deze ondersteunt OGC SOS 1.0 en 2.0 standaard en is de OGC referentie implementatie voor SOS. Daarnaast is ook REST en INSPIRE support toegevoegd voor deze versie. De installatie is net als standaard 52N SOS server met paar uitzonderingen voor INSPIRE config.

Zie http://52north.org/communities/sensorweb/sos/index.html . Installatie volgens instructies op https://wiki.52north.org/bin/view/SensorWeb/SensorObservationServiceIVDocumentation#Installation

- database aangemaakt: naam 'sensors' template postgis2, user 'sensors'
- database schema aangemaakt in DB 'sensors': naam: 'sos' (tbv SOS server tables)
- Apache proxy:

Als volgt in `/etc/apache2/sites-available/sensors.geonovum.nl.conf` (sensors is localhost naam zoals in `/etc/hosts`)

```
  <Location /sos>
    ProxyPass ajp://sensors:8009/sos
    ProxyPassReverse http://sensors/sos
  </Location>

* SOS-download onder ``/opt/52north/sos/20140519``
* war file hernoemen naar sos.war en install: ``cp sos.war /var/www/sensors.geonovum.
↪nl/webapps/``
* via ``tail -f /var/log/tomcat7/catalina.out &`` logfile volgen
* server aktief op ``http://sensors.geonovum.nl/sos``
* melding "You first have to complete the installation process! Click here to start␣
↪it."
* Wizard stappen volgen, schema 'sos' binnen database, daarna via Batch InsertSensor/
↪InsertObservation
* Service URL is ``http://sensors.geonovum.nl/sos/sos``
* moet endpoint aangeven: bijv http://sensors.geonovum.nl/sos/sos/kvp?service=SOS&
↪request=GetCapabilities
```

“Please enter credentials to login into the administrator panel below. You can reset your admin password by executing the file sql/reset_admin.sql (located inside the SOS installation directory in the webapps folder of your application server) on your database. Problemen: memory out of heap,

Tomcat instellingen naar

```
``JAVA_OPTS="-Djava.awt.headless=true -server -Xmx3072M -Xms512M -
↪XX:SoftRefLRUPolicyMSPerMB=36000 -XX:MaxPermSize=1024m -XX:+UseParallelGC"`` .
```

Followed de README.

```
After deploying the WAR file open the SOS page in a browser (http://sensors.geonovum.
↪nl/sos) and follow the installation steps:

1) Datasource configuration: Select PostgreSQL/PostGIS as datasource
   - Enable the Multi language support checkbox in the Advanced Database␣
↪configuration section  (DONE)
   - Re-Installations: Uncheck the Create tables checkbox in the Actions section
2) Settings:
   - CRS ( optional): Change the default CRS and limit the supported CRS  (LEFT AS IS)
   - I18N: Set the default language as ISO 639-2/B alpha 3 code  (DONE, set to 'dut')
   - INSPIRE: Change value if necessary  (LEFT AS IS)
3) Follow the instructions
```

Verdere gegevens:

- Logfile: /var/lib/tomcat7/logs/52n-sos-webapp.log

Patches

Since the install from 19052014, the following patches were applied.

- 20140519-patch: `85658 May 21 17:26 coding-sensorML-v101-4.0.2-SNAPSHOT.jar`
- 20140612-patch: `do-core-0.1.3-SNAPSHOT.jar` and `hibernate-common-4.0.2-SNAPSHOT.jar`

The 20140612-patch solves 2 issues:

1. all Observation identifiers were listed in GetCapabilities: https://github.com/Geonovum/sospilot/issues/2
2. observable property needed to be unique: https://github.com/Geonovum/sospilot/issues/3

Replaced

```
-rw-r--r-- 1 tomcat7 tomcat7   23436 May 19 10:58 do-core-0.1.3-SNAPSHOT.jar
-rw-r--r-- 1 tomcat7 tomcat7  289999 May 19 10:58 hibernate-common-4.0.2-SNAPSHOT.jar
-rw-r--r-- 1 tomcat7 tomcat7   63249 May 19 10:58 cache-4.0.2-SNAPSHOT.jar

with

-rw-r--r-- 1 tomcat7 tomcat7   23529 Jun 12 14:21 do-core-0.1.3-SNAPSHOT.jar
-rw-r--r-- 1 tomcat7 tomcat7  289876 Jun 12 14:21 hibernate-common-4.0.2-SNAPSHOT.jar
-rw-r--r-- 1 tomcat7 tomcat7   72842 Jul  1 16:41 cache-4.0.2-SNAPSHOT.jar
```

Patching is done by: Stop Tomcat, Copy patch .jar to `/var/www/sensors.geonovum.nl/webapps/sos/WEB-INF/lib`, Start Tomcat.

11.4 Installatie - ETL Tools

11.4.1 ImageMagick

Handig voor alllerlei image conversies, oa in gebruik bij NLExtract en MapFish Print.

```
apt-get install imagemagick

# 8:6.7.7.10-5ubuntu3
```

11.4.2 XSLT Processor

Zie http://en.wikipedia.org/wiki/XSLT. *XSLT (XSL Transformations) is a declarative, XML-based language used for the transformation of XML documents into other XML documents.*

Installatie van XSLT processor voor commandline. o.a. gebruikt voor INSPIRE GML transformaties.

```
apt-get install xsltproc
```

11.4.3 GDAL/OGR

Volgens de website www.gdal.org.

GDAL is a translator library for raster geospatial data formats that is released under an X/MIT style Open Source license by the Open Source Geospatial Foundation. The related OGR library (which lives within the GDAL source tree) provides a similar capability for simple features vector data.

Installatie is simpel via APT.

```
$ apt-get install gdal-bin python-gdal

0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Inst gdal-bin (1.10.1+dfsg-5ubuntu1 Ubuntu:14.04/trusty [amd64])
Conf gdal-bin (1.10.1+dfsg-5ubuntu1 Ubuntu:14.04/trusty [amd64])
Setting up python-numpy (1:1.8.1-1ubuntu1) ...
Setting up python-gdal (1.10.1+dfsg-5ubuntu1) ...
```

11.4.4 Stetl - Streaming ETL

Zie http://stetl.org

Eerst alle dependencies!

```
apt-get install python-pip
apt-get install python-lxml
apt-get install postgresql-server-dev-9.3
apt-get install python-gdal libgdal-dev
apt-get install python-psycopg2
```

Normaal doen we `pip install stetl` maar nu even install uit Git vanwege te verwachten updates.Install vanuit GitHub versie onder `/opt/stetl/git`.

```
$ mkdir /opt/stetl
$ cd /opt/stetl
$ git clone https://github.com/geopython/stetl.git git
$ cd git
$ python setup.py install

$ stetl -h
# 2014-05-25 13:43:40,930 util INFO running with lxml.etree, good!
# 2014-05-25 13:43:40,931 util INFO running with cStringIO, fabulous!
# 2014-05-25 13:43:40,936 main INFO Stetl version = 1.0.5
```

Installatie Testen.

```
$ which stetl
# /usr/local/bin/stetl

cd /opt/stetl/git/examples/basics
./runall.sh
# OK!
```

11.4.5 Python Jinja2

Nodig voor Stetl Jinja2 templating Filter.

```
pip install jinja2
Downloading/unpacking jinja2
  Downloading Jinja2-2.7.3.tar.gz (378kB): 378kB downloaded
  Running setup.py (path:/tmp/pip_build_root/jinja2/setup.py) egg_info for package␣
↪jinja2

    warning: no files found matching '*' under directory 'custom_fixers'
    warning: no previously-included files matching '*' found under directory 'docs/_
↪build'
    warning: no previously-included files matching '*.pyc' found under directory
↪'jinja2'
    warning: no previously-included files matching '*.pyc' found under directory 'docs
↪'
    warning: no previously-included files matching '*.pyo' found under directory
↪'jinja2'
    warning: no previously-included files matching '*.pyo' found under directory 'docs
↪'
Downloading/unpacking markupsafe (from jinja2)
  Downloading MarkupSafe-0.23.tar.gz
  Running setup.py (path:/tmp/pip_build_root/markupsafe/setup.py) egg_info for␣
↪package markupsafe

Installing collected packages: jinja2, markupsafe
  Running setup.py install for jinja2

    warning: no files found matching '*' under directory 'custom_fixers'
    warning: no previously-included files matching '*' found under directory 'docs/_
↪build'
    warning: no previously-included files matching '*.pyc' found under directory
↪'jinja2'
    warning: no previously-included files matching '*.pyc' found under directory 'docs
↪'
```

```
    warning: no previously-included files matching '*.pyo' found under directory
↪'jinja2'
    warning: no previously-included files matching '*.pyo' found under directory 'docs
↪'
  Running setup.py install for markupsafe

    building 'markupsafe._speedups' extension
    x86_64-linux-gnu-gcc -pthread -fno-strict-aliasing -DNDEBUG -g -fwrapv -O2 -Wall -
↪Wstrict-prototypes -fPIC -I/usr/include/python2.7 -c markupsafe/_speedups.c -o␣
↪build/temp.linux-x86_64-2.7/markupsafe/_speedups.o
    markupsafe/_speedups.c:12:20: fatal error: Python.h: No such file or directory
     #include <Python.h>
                        ^
    compilation terminated.
    ================================================================================
    WARNING: The C extension could not be compiled, speedups are not enabled.
    Failure information, if any, is above.
    Retrying the build without the C extension now.


    ================================================================================
    WARNING: The C extension could not be compiled, speedups are not enabled.
    Plain-Python installation succeeded.
    ================================================================================
Successfully installed jinja2 markupsafe
Cleaning up...
```

11.5 Installatie - Project Software

Software en documentatie voor project zit in Geonovum GitHub: https://github.com/Geonovum/sospilot

We installeren deze onder `/opt/geonovum/sospilot`

```
cd /opt/geonovum/sospilot
git clone https://github.com/Geonovum/sospilot.git git
```

NB alle documentatie (Sphinx) wordt automatisch gepubliceerd naar ReadTheDocs.org: http://sospilot.readthedocs.org via een GitHub Post-commit hook.

11.6 Installatie - Ontwikkeltools

Hieronder de installaties voor de verschillende tools mbt software ontwikkelen.

11.6.1 Ant - Java Build Tool

Volgens de Ant website.

Apache Ant is a Java-based build tool. In theory, it is kind of like Make, but without Make's wrinkles.

Installatie:

```
apt-get install ant
ant -version
# Apache Ant(TM) version 1.9.2 compiled on July 14 2013
#
# /usr/share/ant contains install
```

11.6.2 Maven - Lifecycle Tool

Volgens de Maven website.

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

Installatie:

```
$ apt-get install maven2
$ mvn -version
# Apache Maven 2.2.1 (rdebian-10)
# Java version: 1.7.0_45
# Java home: /usr/lib/jvm/java-7-oracle/jre
# Default locale: en_US, platform encoding: UTF-8
# OS name: "linux" version: "3.11.0-12-generic" arch: "amd64" Family: "unix"

# configuratie (globaal) in
# /usr/share//maven2/conf

# per-user conf en repository in ~/user/.m2 bijv
# /home/sadmin/.m2
```

Hmm, we should have used the standard `apt-get install maven` to get Maven 3...

On July 11, 2014, did

```
$ apt-get remove maven2
$ apt-get install maven
```

11.6.3 Git - Source Code Beheer

apt-get install git-core Zie https://help.ubuntu.com/13.10/serverguide/git.html

11.6.4 ncdump - dumping NetCDF files

Used for extracting a.o. KNMI weather data files. Install

```
apt-get install netcdf-bin
```

11.6.5 weewx - Weather Station server

Used for testing weewx.

Dir: */opt/weewx*. We do custom install as user *sadmin* in order to make tweaking easier.

See http://www.weewx.com/docs/setup.htm

Steps.

```
# Install Dependencies
# required packages:
apt-get install python-configobj
apt-get install python-cheetah
apt-get install python-imaging
apt-get install fonts-freefont-ttf  # Fonts in reporting

# optional for extended almanac information:
apt-get install python-dev
pip install pyephem

# Weewx install after download
cd /opt/weewx
tar xzvf archive/weewx-2.7.0.tar.gz
ln -s weewx-2.7.0 weewx

cd weewx

# Change install dir in setup.cfg as follows
# Configuration file for weewx installer. The syntax is from module
# ConfigParser. See http://docs.python.org/library/configparser.html

[install]

# Set the following to the root directory where weewx should be installed
home = /opt/weewx/weewxinst

# Given the value of 'home' above, the following are reasonable values
prefix =
exec-prefix =
install_lib = %(home)s/bin
install_scripts = %(home)s/bin

# build en install in /opt/weewx/weewxinst
./setup.py build
./setup.py install

# link met aangepaste configs uit Geonovum GitHub (na backup oude versies)
ln -s /opt/geonovum/sospilot/git/src/weewx/test/weewx.conf /opt/weewx/weewxinst
ln -s /opt/geonovum/sospilot/git/src/weewx/test/skin.conf /opt/weewx/weewxinst/skins/
↪Standard
ln -s /opt/geonovum/sospilot/git/src/weewx/test/weatherapidriver.py /opt/weewx/
↪weewxinst/bin/user

# test OK
sadmin@vps44500:/opt/weewx/weewxinst$ ./bin/weewxd weewx.conf
('Created packet: %s', "{'barometer': 29.681039574719435, 'windchill': 56.48,
↪'dewpoint': 52.656315478047,
'pressure': 29.681039574719435, 'outHumidity': 87, 'heatindex': 56.48, 'dateTime':␣
↪1413323976, 'windDir': 200,
'outTemp': 56.48, 'windSpeed': 14.47, 'rainRate': 43.33, 'usUnits': 1}")
LOOP:   2014-10-14 23:59:36 CEST (1413323976) {'barometer': 29.681039574719435,
↪'windchill': 56.48, 'dewpoint': 52.656315478047,
'pressure': 29.681039574719435, 'outHumidity': 87, 'heatindex': 56.48, 'dateTime':␣
↪1413323976, 'windDir': 200, 'outTemp': 56.48,
'windSpeed': 14.47, 'rainRate': 43.33, 'usUnits': 1}
```

```
# install weewx daemon in /etc/init.d (als root)
# aanpassen settings in daemon in GitHub  /opt/geonovum/sospilot/git/src/weewx/test/
↪weewx-daemon.sh

# PATH should only include /usr/* if it runs after the mountnfs.sh script
WEEWX_HOME=/opt/weewx/weewxinst
PATH=/sbin:/usr/sbin:/bin:/usr/bin
WEEWX_BIN=$WEEWX_HOME/bin/weewxd
WEEWX_CFG=$WEEWX_HOME/weewx.conf
DESC="weewx weather system"
NAME=weewx
WEEWX_USER=sadmin:sadmin
PIDFILE=$WEEWX_HOME/$NAME.pid
DAEMON=$WEEWX_BIN
DAEMON_ARGS="--daemon --pidfile=$PIDFILE $WEEWX_CFG"
SCRIPTNAME=/etc/init.d/$NAME

cp /opt/geonovum/sospilot/git/src/weewx/test/weewx-daemon.sh /etc/init.d
update-rc.d weewx defaults
/etc/init.d/weewx start
/etc/init.d/weewx status
* Status of weewx weather system: running

# weewx log bekijken
tail -f /var/log/syslog

# memory in gaten houden
  PID USER      PR  NI    VIRT    RES    SHR  S  %CPU %MEM     TIME+ COMMAND
 4688 sadmin    20   0    170936  36776  4608 S   0.0  0.5   3:15.23 weewxd  (16.10.
↪14 16:22)
 5269 sadmin    20   0    173920  39024  4792 S   0.0  0.5   2:07.12 weewxd
```

11.7 Tot hier gekomen op 25.5.2014

11.8 TODO

Onderstaande alleen installeren indien nodig.

11.8.1 Sphinx - Documentatie

Zie http://sphinx.pocoo.org. *Sphinx is a tool that makes it easy to create intelligent and beautiful documentation, written by Georg Brandl and licensed under the BSD license.*

Installatie Sphinx v1.1.3

```
$ apt-get install sphinx-doc
$ apt-get install python-sphinx

# 1.1.3
NIET MET easy_install -U Sphinx
```

Tutorial http://matplotlib.sourceforge.net/sampledoc. PDF generation installatie via Latex: http://linuxandfriends.com/2009/10/06/install-latex-in-ubuntu-linux.

```
apt-get  install texlive-full
```

11.9 Installatie - Beheer

11.9.1 IPTables Firewall

https://help.ubuntu.com/community/IptablesHowTo We laten alleen HTTP(S) en SSH door naar buiten (eth0/176.9.2.29 en fe80::5054:ff:fed8:5cf7 voor IPv6) en Munin, poort 4949, voor binnen (eth1). We doen dit met `iptables` en maken de rules persisten met `iptables-persistent`. Dit moet voor IP v4 en v6!!

/opt/bin/iptables-start.sh,

```
# https://help.ubuntu.com/community/IptablesHowTo
# http://www.linux-noob.com/forums/index.php?/topic/1280-iptables-block-all-ports-
↪except-20-21/
# complete tutorial: https://www.frozentux.net/iptables-tutorial/iptables-tutorial.
↪html
iptables-stop.sh

iptables -P INPUT DROP
iptables -I INPUT 1 -i lo -j ACCEPT
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --dport ssh -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --dport 443 -j ACCEPT
iptables -A INPUT -i eth1 -p tcp --dport 4949 -s 192.168.100.0/24 -j ACCEPT
SERVER_IP="176.9.2.29"
iptables -A INPUT -p icmp --icmp-type 8 -s 0/0 -d $SERVER_IP -m state --state NEW,
↪ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type 0 -s $SERVER_IP -d 0/0 -m state --state␣
↪ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type 8 -s $SERVER_IP -d 0/0 -m state --state NEW,
↪ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 0 -s 0/0 -d $SERVER_IP -m state --state␣
↪ESTABLISHED,RELATED -j ACCEPT
iptables -L -V

# en voor v6, let op -p icmpv6 --icmpv6-type
ip6tables -P INPUT DROP
ip6tables -I INPUT 1 -i lo -j ACCEPT
ip6tables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
ip6tables -A INPUT -i eth0 -p tcp --dport ssh -j ACCEPT
ip6tables -A INPUT -i eth0 -p tcp --dport 80 -j ACCEPT
ip6tables -A INPUT -i eth0 -p tcp --dport 443 -j ACCEPT
SERVER_IP="fe80::5054:ff:fed8:5cf7"
# use --icmpv6-type
ip6tables -A INPUT -p icmpv6 --icmpv6-type 8 -s 0/0 -d $SERVER_IP -m state --state␣
↪NEW,ESTABLISHED,RELATED -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type 0 -s $SERVER_IP -d 0/0 -m state --state␣
↪ESTABLISHED,RELATED -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type 8 -s $SERVER_IP -d 0/0 -m state --state␣
↪NEW,ESTABLISHED,RELATED -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type 0 -s 0/0 -d $SERVER_IP -m state --state␣
↪ESTABLISHED,RELATED -j ACCEPT
ip6tables -L -V
```

/opt/bin/iptables-stop.sh

```
    echo "Stopping firewall and allowing everyone..."
    iptables -F
    iptables -X
    iptables -t nat -F
    iptables -t nat -X
    iptables -t mangle -F
    iptables -t mangle -X
    iptables -P INPUT ACCEPT
    iptables -P FORWARD ACCEPT
    iptables -P OUTPUT ACCEPT
    iptables -L -V

    iptables -L -v
Chain INPUT (policy DROP 8 packets, 484 bytes)
 pkts bytes target     prot opt in     out     source               destination
   36 11344 ACCEPT     all  --  lo     any     anywhere             anywhere
  229 24367 ACCEPT     all  --  any    any     anywhere             anywhere
↪    ctstate RELATED,ESTABLISHED
    2   128 ACCEPT     tcp  --  eth0   any     anywhere             anywhere
↪    tcp dpt:ssh
    0     0 ACCEPT     tcp  --  eth0   any     anywhere             anywhere
↪    tcp dpt:http
    0     0 ACCEPT     tcp  --  eth0   any     anywhere             anywhere
↪    tcp dpt:https
    1    84 ACCEPT     icmp --  any    any     anywhere             static.29.2.9.176.
↪clients.your-server.de  icmp echo-request state NEW,RELATED,ESTABLISHED
    0     0 ACCEPT     icmp --  any    any     anywhere             static.29.2.9.176.
↪clients.your-server.de  icmp echo-reply state RELATED,ESTABLISHED

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 199 packets, 48858 bytes)
 pkts bytes target     prot opt in     out     source               destination
    2   168 ACCEPT     icmp --  any    any     static.29.2.9.176.clients.your-server.
↪de  anywhere             icmp echo-reply state RELATED,ESTABLISHED
    0     0 ACCEPT     icmp --  any    any     static.29.2.9.176.clients.your-server.
↪de  anywhere             icmp echo-request state NEW,RELATED,ESTABLISHED
```

Persistent maken over reboots met `ip-tables-persistent` http://tomearp.blogspot.nl/2012/07/using-iptables-save-and-restore-with.html

```
$ apt-get install iptables-persistent
$ ip6tables-save > /etc/iptables/rules.v6
$ iptables-save > /etc/iptables/rules.v4

# rules worden bewaard in /etc/iptables/rules.v4|6
$ cat /etc/iptables/rules.v4
# Generated by iptables-save v1.4.18 on Mon Dec 23 14:12:21 2013
*mangle
:PREROUTING ACCEPT [353:57105]
:INPUT ACCEPT [353:57105]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [313:92148]
:POSTROUTING ACCEPT [313:92148]
COMMIT
# Completed on Mon Dec 23 14:12:21 2013
```

```
# Generated by iptables-save v1.4.18 on Mon Dec 23 14:12:21 2013
*nat
:PREROUTING ACCEPT [9:516]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [8:563]
:POSTROUTING ACCEPT [8:563]
COMMIT
# Completed on Mon Dec 23 14:12:21 2013
# Generated by iptables-save v1.4.18 on Mon Dec 23 14:12:21 2013
*filter
:INPUT DROP [9:516]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [311:91932]
-A INPUT -i lo -j ACCEPT
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 443 -j ACCEPT
-A INPUT -d 176.9.2.29/32 -p icmp -m icmp --icmp-type 8 -m state --state NEW,RELATED,
 ↪ESTABLISHED -j ACCEPT
-A INPUT -d 176.9.2.29/32 -p icmp -m icmp --icmp-type 0 -m state --state RELATED,
 ↪ESTABLISHED -j ACCEPT
-A OUTPUT -s 176.9.2.29/32 -p icmp -m icmp --icmp-type 0 -m state --state RELATED,
 ↪ESTABLISHED -j ACCEPT
-A OUTPUT -s 176.9.2.29/32 -p icmp -m icmp --icmp-type 8 -m state --state NEW,RELATED,
 ↪ESTABLISHED -j ACCEPT
COMMIT
# Completed on Mon Dec 23 14:12:21 2013

cat /etc/iptables/rules.v6
# Generated by ip6tables-save v1.4.18 on Mon Dec 23 14:29:44 2013
*mangle
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
COMMIT
# Completed on Mon Dec 23 14:29:44 2013
# Generated by ip6tables-save v1.4.18 on Mon Dec 23 14:29:44 2013
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
COMMIT
# Completed on Mon Dec 23 14:29:44 2013
# Generated by ip6tables-save v1.4.18 on Mon Dec 23 14:29:44 2013
*filter
:INPUT DROP [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 443 -j ACCEPT
-A INPUT -d fe80::5054:ff:fed8:5cf7/128 -p ipv6-icmp -m icmp6 --icmpv6-type 8 -m␣
 ↪state --state NEW,RELATED,ESTABLISHED -j ACCEPT
```

```
-A INPUT -d fe80::5054:ff:fed8:5cf7/128 -p ipv6-icmp -m icmp6 --icmpv6-type 0 -m␣
↪state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -s fe80::5054:ff:fed8:5cf7/128 -p ipv6-icmp -m icmp6 --icmpv6-type 0 -m␣
↪state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -s fe80::5054:ff:fed8:5cf7/128 -p ipv6-icmp -m icmp6 --icmpv6-type 8 -m␣
↪state --state NEW,RELATED,ESTABLISHED -j ACCEPT
COMMIT
# Completed on Mon Dec 23 14:29:44 2013
```

11.9.2 Webalizer

Zie http://www.mrunix.net/webalizer/. *The Webalizer is a fast, free web server log file analysis program. It produces highly detailed, easily configurable usage reports in HTML format, for viewing with a standard web browser.*

Installatie,

```
$ apt-get install webalizer
# installeer webalizer configuratie in /etc/webalizer/

# zorg dat output zichtbaar is via dir onder /var/www/default/sadm/webalizer

# enable DNS lookups
touch  /var/cache/webalizer/dns_cache.db
```

11.9.3 Optimaliseren van Tomcat

Zetten server parameters. Zie ook: http://docs.geoserver.org/stable/en/user/production/container.html

```
# in /etc/default/tomcat7
JAVA_OPTS="-Djava.awt.headless=true -server -Xmx8192M -Xms512M -
↪XX:SoftRefLRUPolicyMSPerMB=36000 -XX:MaxPermSize=512m -XX:+UseParallelGC"
```

GDAL bindings, nu nog even, niet evt later.

```
# TODO (nu nog even niet)
# GDAL JNI
# WARNING: Native library load failed.java.lang.UnsatisfiedLinkError: no gdaljni in␣
↪java.library.path

# try to install gdal java bindings
# see https://imageio-ext.dev.java.net/files/documents/7505/124115/ImageioExt-
↪SetupGuide.pdf
# http://docs.geoserver.org/stable/en/user/data/raster/gdal.html
apt-get install swig
# Be sure you have properly downloaded SWIG, the Simplified Wrapper and Interface␣
↪Generator
# which allow to produce JAVA bindings for C/C++ code. You can obtain it by simply␣
↪running:
```

11.9.4 Break-in attempts blokkeren met denyhosts

Zie http://denyhosts.sourceforge.net

Analyseert de /var/log/auth.log file op break-in en herhaaldelijk inloggen (bijv. dictionary attacks) en voegt hosts toe aan /etc/hosts.deny

```
apt-get install denyhosts

# installs 2.6-10
```

Configuratie in `/etc/denyhosts.cfg` (email adres en Subject aanpassen)

Om deblokkeren, zie. Data files staan onder */var/lib*: http://www.cyberciti.biz/faq/linux-unix-delete-remove-ip-address-that-denyhosts-blocked/

11.9.5 Optimaliseren van Java

Dit is nodig met name om image-rendering te optimaliseren binnen alle Java-gebaseerde tools zoals GeoServer. Moet opnieuw bij elke Java JDK upgrade...

Zie http://docs.geoserver.org/stable/en/user/production/java.html. NB dit moet iedere keer als de sun-java JDK wordt geupgrade !! En recenter: http://geoserver.geo-solutions.it/edu/en/install_run/jai_io_install.html

```
Go to the JAI download page and download the Linux installer for version 1.1.3,␣
↪choosing the appropriate architecture:

i586 for the 32 bit systems
amd64 for the 64 bit ones (even if using Intel processors)
Copy the file into the directory containing the JDK/JRE and then run it. For example,␣
↪on an Ubuntu 32 bit system:

# install JAI+JAI imageIO
# Go to the JAI download page and download the Linux installer for version 1.1.3,
# choosing the appropriate architecture:
# i586 for the 32 bit systems
# amd64 for the 64 bit ones (even if using Intel processors)

$ mkdir /opt/jai+imageio
$ wget http://download.java.net/media/jai/builds/release/1_1_3/jai-1_1_3-lib-linux-
↪amd64-jdk.bin
$ wget http://download.java.net/media/jai-imageio/builds/release/1.1/jai_imageio-1_1-
↪lib-linux-amd64-jdk.bin

# Copy the file into the directory containing the JDK/JRE and then run it. For␣
↪example, on an Ubuntu 64 bit system:

Script
#!/bin/sh
# Copy the file into the directory containing the JDK/JRE and then run it. For␣
↪example, on an Ubuntu 64 bit system:
# Do this as root! sudo su - first
cp /opt/jai+imageio/jai-1_1_3-lib-linux-amd64-jdk.bin /usr/lib/jvm/java-7-oracle/
cd /usr/lib/jvm/java-7-oracle/
sh jai-1_1_3-lib-linux-amd64-jdk.bin
    # accept license
rm jai-1_1_3-lib-linux-amd64-jdk.bin

    # Then jai_imageio
    # If you encounter difficulties (Unpacking...
    # tail: cannot open `+215' for reading:
    # No such file or directory) , you may need to export the environment variable
```

```
    # _POSIX2_VERSION=199209. For example, on a Ubuntu 64 bit Linux system:

cp /opt/jai+imageio/jai_imageio-1_1-lib-linux-amd64-jdk.bin /usr/lib/jvm/java-7-
↪oracle/
cd /usr/lib/jvm/java-7-oracle/
export _POSIX2_VERSION=199209
sh jai_imageio-1_1-lib-linux-amd64-jdk.bin
# accept license
rm jai_imageio-1_1-lib-linux-amd64-jdk.bin
```

11.9.6 Extra Fonts

Hoeft blijkbaar niet bij elke Java JDK upgrade...

Installeren MS fonts zie http://corefonts.sourceforge.net en http://embraceubuntu.com/2005/09/09/installing-microsoft-fonts.

```
apt-get install msttcorefonts
# installs in /usr/share/fonts/truetype/msttcorefonts
```

Installeren fonts in Java (for geoserver).

- Few fonts are included with Java by default, and for most people the the official documentation falls short of a useful explanation. It is unclear exactly where Java looks for fonts, so the easiest way to solve this problems is to copy whatever you need to a path guaranteed to be read by Java, which in our case is `/usr/lib/jvm/java-7-oracle`
- First install the fonts you want. The MS Core Fonts (Arial, Times New Roman, Verdana etc.) can be installed by following the instructions on http://corefonts.sourceforge.net/.
- **Now copy the .ttf files to `/usr/lib/jvm/java-7-oracle/` and run (ttmkfdir is obsolete??),** from http://askubuntu.com/questions/22448/not-all-ttf-fonts-visible-from-the-sun-jdk this install

Commands

```
mkfontscale
mkfontdir
fc-cache -f -v
```

All that remains is to restart any Java processes you have running, and the new fonts should be available.

11.9.7 UMN MapServer

Volgens de website www.mapserver.org.

MapServer is an Open Source platform for publishing spatial data and interactive mapping applications to the web. Originally developed in the mid-1990's at the University of Minnesota, MapServer is released under an MIT-style license, and runs on all major platforms.

Installatie is simpel via APT.

```
apt-get install mapserver-bin
# Setting up mapserver-bin (6.4.0-5~saucy3)

# ook de CGI installeren
apt-get install cgi-mapserver
# Setting up cgi-mapserver (6.4.0-5~saucy3)
```

```
# installs mapserv in /usr/lib/cgi-bin

# installatie testen
/usr/lib/cgi-bin/mapserv -v
MapServer version 6.4.0 OUTPUT=GIF OUTPUT=PNG OUTPUT=JPEG OUTPUT=KML SUPPORTS=PROJ
SUPPORTS=GD SUPPORTS=AGG SUPPORTS=FREETYPE SUPPORTS=CAIRO SUPPORTS=SVG_SYMBOLS␣
↪SUPPORTS=RSVG
SUPPORTS=ICONV SUPPORTS=FRIBIDI SUPPORTS=WMS_SERVER SUPPORTS=WMS_CLIENT
SUPPORTS=WFS_SERVER SUPPORTS=WFS_CLIENT
SUPPORTS=WCS_SERVER SUPPORTS=SOS_SERVER SUPPORTS=FASTCGI SUPPORTS=THREADS
SUPPORTS=GEOS INPUT=JPEG INPUT=POSTGIS INPUT=OGR INPUT=GDAL INPUT=SHAPEFILE
```

Een UMN MapServer tutorial http://mapserver.gis.umn.edu/new_users

In Apache CGI enablen: uncomment `AddHandler cgi-script .cgi`

Gebruik van CGI wrapper zodat lelijke *map=* uit URL kan staat op http://mapserver.gis.umn.edu/docs/howto/cgi-wrapper-script. Hieronder een voorbeeld van een CGI wrapper:

```
#!/bin/sh
# shortcut for mapserver with specific mapfile
# allows friendly URLs like http://my.com/ms/map1?service=wms...
# i.s.o. cgi with full mapfile path
#
MAPSERV="/usr/lib/cgi-bin/mapserv"
MAPFILE="/home/ticheler/kadaster_webapp/umn_kadkaart/kadaster_nl_topografie.map"

if [ "${REQUEST_METHOD}" = "GET" ]; then
  if [ -z "${QUERY_STRING}" ]; then
    QUERY_STRING="map=${MAPFILE}"
  else
    QUERY_STRING="map=${MAPFILE}&${QUERY_STRING}"
  fi
  exec ${MAPSERV}
else
  echo "Sorry, I only understand GET requests."
fi
exit 1
```

11.9.8 MapProxy

Zie http://mapproxy.org/docs/latest/install.html.

```
$ apt-get install python-imaging python-yaml libproj0
$ apt-get install libgeos-dev python-lxml libgdal-dev python-shapely

# Setting up python-shapely (1.2.14-1) ...
$ apt-get install build-essential python-dev libjpeg-dev zlib1g-dev libfreetype6-dev
$ pip install Pillow
# was already installed in
# /usr/lib/python2.7/dist-packages: Pillow-2.0.0
$ pip install MapProxy
# ... Downloading MapProxy-1.6.0.tar.gz
$ mapproxy-util --version
# Mapproxy 1.6.0
```

11.9.9 MongoDB Beheer

Met http://www.phpmoadmin.com, wel eerst PHP5 MongoDB driver installeren: ‘‘ apt-get install php5-mongo‘‘. Verder is beheer URL afgeschermd.

CHAPTER 12

FIWARE - Evaluation

Starting in october 2015 the SOSPilot platform was extended to use and deploy FIWARE.

12.1 The Plan

The architecture used for this setup is depicted below. This architecture emerged dynamically as described below.

This setup is similar as in this FIWARE presentation.

From bottom to top the setup is as follows:

- Sensors or manual input use the UltraLight 2.0 (UL2.0) and MQTT protocols for managing services and devices and sending observations
- The FIWARE IoTAgentCpp device API is used
- Client libraries (Python, Arduino) are used to facilitate using the UL2.0 and MQTT protocols
- The client may reside anywhere on the Internet
- the server `sensors.geonovum.nl` hosts the FIWARE components `Orion Context Broker` (Orion CB or OCB), `IoTAgentCpp` and `Short Term History (STH)`
- all persistence for these components is done in `MongoDB`
- `IoTAgentCpp` translates requests from the UL2.0/MQTT clients to Orion CB NGSI requests
- `OCB` persists the *latest* values as Entities in MongoDB
- `Short Term History (STH aka STH Comet)` subscribes to Orion CB NGSI Entity attribute change events and stores *Timeseries* in MongoDB
- `WireCloud` (WC), the Mashup environment runs in the FIWARE Lab server at lab.fiware.org
- WC communicates to (any) OCB using the NGSI10 protocol
- within WC mashups are produced in order to view and interact with the sensor data

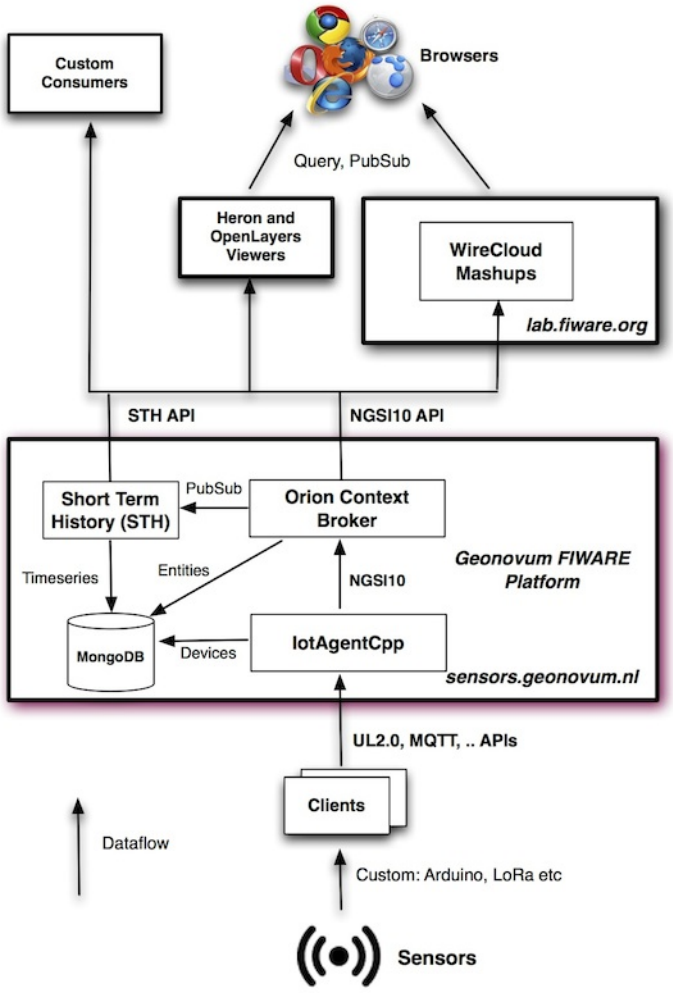


Fig. 12.1: *Architecture for FIWARE within SOSPilot platform*

- by developing an OpenLayers NGSI10 Vector Layer, viewers like HeronViewer can show (real-time) sensor data

NGSI10 is a specification from the Open Mobile Alliance (OMA). The FI-WARE version of the OMA NGSI 10 interface is a RESTful API via HTTP.

STH provides Timeseries storage and an API to request Timeseries data. See https://github.com/telefonicaid/fiware-sth-comet.

The above setup provides APIs for a complete IoT platform:

For data producers (sensors):

- APIs for sensor-provisioning (via UL2.0)
- APIs (UL2.0, MQTT) for sensors to send observations

For data consumers:

- API to request latest values (NGSI10 Query API)
- API to subscribe to real-time updates (NGSI10 Publish-Subscribe API)
- API to request historic timeseries data (STH API)

12.2 FIWARE Docs

This document provides most of the details of the above components: http://fiware-iot-stack.readthedocs.org/en/latest/index.html

12.3 Via lab.fiware.org

Initially it was attempted to realize the above architecture via the public FIWARE Lab platform but we found some isssues. In a later stage the above setup was realized within the Geonovum (Ubuntu/Linux) server VPS. Some info follows:

Registering at the international FIWARE Lab worked ok, but could not get the Orion CB with the IDAS IoTAgent working. The interaction between the two seemed to be broken. This was reported, see specifics here on StackOverflow:

And appearantly this issue was found by others as well.

After an unsuccessful attempt to compile and run the OCB and IoT Agent on Ubuntu 14.04-3 it was decided to use Docker on the Geonovum SOSPilot VPS (Ubuntu/Linux server VPS). Via Docker seems the best/recommended option anyway as CentOS is the primary target platform for FIWARE. See next sections.

12.4 Installing FIWARE - with Docker

See http://www.slideshare.net/dmoranj/iot-agents-introduction

12.4.1 Install Docker

See https://docs.docker.com/installation/ubuntulinux. Install via Docker APT repo.

Steps.

```
# Kernel version OK for Docker
$ uname -r
3.13.0-66-generic

# Add key
apt-key adv --keyserver hkp://pgp.mit.edu:80 --recv-keys␣
↪58118E89F3A912897C070ADBF76221572C52609D

# Add to repo by putting this line in /etc/apt/sources.list.d/docker.list
add deb https://apt.dockerproject.org/repo ubuntu-trusty main to

$ apt-get update
$ apt-cache policy docker-engine

# install docker engine
apt-get update
apt-get install docker-engine

# test
docker run hello-world
docker run -it ubuntu bash

# cleanup non-running images
docker rm -v $(docker ps -a -q -f status=exited)
docker rmi $(docker images -f "dangling=true" -q)
```

Docker-compose. https://docs.docker.com/compose/install. Easiest via `pip`.

```
$ pip install docker-compose
```

See also CLI utils for `docker-compose`: https://docs.docker.com/v1.5/compose/cli/

Docker utils.

```
docker ps -a

# go into docker image named docker_iotacpp_1 to bash prompt
docker exec -it docker_iotacpp_1 bash
```

12.4.2 Install FIWARE

Installing the FIWARE Docker components to realize IoT setup: IoT Agent, Orion CB, Short term History (STH) with MongoDB persistence. Intro: http://www.slideshare.net/dmoranj/iot-agents-introduction

Take docker-compose for fiware-IoTAgent-Cplusplus as starting point: https://github.com/telefonicaid/fiware-IoTAgent-Cplusplus/tree/develop/docker. All our local development related to Docker can be found here: https://github.com/Geonovum/sospilot/tree/master/src/fiware/docker .

Steps. Follow: https://github.com/telefonicaid/fiware-IoTAgent-Cplusplus/blob/develop/docker/readme.md

```
mkdir -p /opt/fiware/iotagent
cd /opt/fiware/iotagent
git clone https://github.com/telefonicaid/fiware-IoTAgent-Cplusplus iotacpp

#
cd /opt/fiware/iotagent/iotacpp/docker
```

Networking from outside to docker containers. See http://blog.oddbit.com/2014/08/11/four-ways-to-connect-a-docker. Make two utilities, `docker-pid` and `docker-ip` in `/opt/bin`.

```
#!/bin/sh

exec docker inspect --format '{{ .State.Pid }}' "$@"
```

```
#!/bin/sh

exec docker inspect --format '{{ .NetworkSettings.IPAddress }}' "$@"
```

But simpler is to follow: https://docs.docker.com/userguide/dockerlinks/ and even easier via `docker-compose iota.yaml`: https://docs.docker.com/compose/yml. Use the `ports` property: "Expose ports. Either specify both ports (HOST:CONTAINER), or just the container port (a random host port will be chosen)." So our `iotarush.yml` (derived from iota.yaml) becomes:

```
#mongodbdata:
#    image: mongo:2.6
#    volumes:
#        - /data/db
#    restart: "no"
#    command: /bin/echo "Data-only container for mongodb."

#mongodb:
#    image: mongo:2.6
#    volumes_from:
#        - mongodbdata
#    expose:
#        - "27017"
#    command: --smallfiles

mongodb:
    image: mongo:2.6

    # Port mapping: allow only access on local docker host
    ports:
      - "127.0.0.1:27017:27017"
    expose:
      - "27017"

    # Use storage on host file system
    volumes:
      - /var/lib/mongodb_docker:/data/db
    command: --smallfiles

    # restart: always

orion:
    image: geonovum/orionrush
    links:
        - mongodb
    ports:
        - "1026:1026"

sthcomet:
    image: geonovum/sthcomet
    log_driver: "syslog"
    links:
```

```
        - mongodb
        - orion
    ports:
        - "8666:8666"

iotacpp:
    image: telefonicaiot/iotacpp
    links:
        - mongodb
        - orion
    ports:
        - "185.21.189.59:8000:8080"
        - "185.21.189.59:8081:8081"
        - "185.21.189.59:1883:1883"
```

Now start.

```
# Start containers (-d iotacpp option not required?)
$ docker-compose -f iotarush.yaml up

# Stopping
$ docker-compose -f iotarush.yaml stop

# check
$ docker images
REPOSITORY              TAG                 IMAGE ID            CREATED
↪VIRTUAL SIZE
geonovum/orionrush      latest              d097edbfaa4d        2 days ago
↪535.8 MB
geonovum/sthcomet       latest              778b09dbecc9        3 days ago
↪686.4 MB
fiware/orion            latest              a5f228ae72c3        5 weeks ago
↪277.1 MB
telefonicaiot/iotacpp   latest              583fbe68b08e        5 weeks ago         2.
↪092 GB
mongo                   2.6                 dd4b3c1d1e51        5 weeks ago
↪392.8 MB

$ docker ps
CONTAINER ID        IMAGE                   COMMAND                  CREATED
↪    STATUS              PORTS
↪                           NAMES
d3e35f677462        telefonicaiot/iotacpp   "/docker-entrypoint.s"   47 hours ago
↪    Up About an hour    185.21.189.59:1883->1883/tcp, 185.21.189.59:8081->8081/tcp,
↪185.21.189.59:8000->8080/tcp   docker_iotacpp_1
d60d467d4e18        geonovum/sthcomet       "npm start"              47 hours ago
↪    Up About an hour    0.0.0.0:8666->8666/tcp
↪                           docker_sthcomet_1
425dd1179b71        geonovum/orionrush      "/docker-entrypoint.s"   47 hours ago
↪    Up About an hour    0.0.0.0:1026->1026/tcp
↪                           docker_orion_1
ad755ed4d28f        mongo:2.6               "/entrypoint.sh --sma"   47 hours ago
↪    Up About an hour    27017/tcp
↪                           docker_mongodb_1

# get into a container with bash
docker exec -it docker_orion_1 bash
[root@1c9dceec8ec8 /]# ps -elf
```

```
F S UID        PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
4 S root         1     0  0  80   0 - 50812 hrtime 15:19 ?        00:00:00 /usr/bin/
↪contextBroker -fg -multiservice
4 S root       976     0  0  80   0 -  3374 wait   15:35 ?        00:00:00 bash
0 R root      1021   976  0  80   0 -  3846 -      15:36 ?        00:00:00 ps -elf

[root@1c9dceec8ec8 /]# cat /etc/hosts
172.17.0.41     1c9dceec8ec8
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.40     mongodb 41028cff906b docker_mongodb_1
172.17.0.40     mongodb_1 41028cff906b docker_mongodb_1
172.17.0.40     docker_mongodb_1 41028cff906b
172.17.0.40     docker_mongodb_1.bridge
172.17.0.41     docker_orion_1.bridge
172.17.0.41     docker_orion_1
172.17.0.40     docker_mongodb_1
172.17.0.42     docker_iotacpp_1
172.17.0.42     docker_iotacpp_1.bridge

# Check Orion
$ curl 172.17.0.41:1026/statistics
<orion>
        <versionRequests>0</versionRequests>
        <statisticsRequests>1</statisticsRequests>
        <uptime_in_secs>1472</uptime_in_secs>
        <measuring_interval_in_secs>1472</measuring_interval_in_secs>
        <subCacheRefreshs>3</subCacheRefreshs>
        <subCacheInserts>0</subCacheInserts>
        <subCacheRemoves>0</subCacheRemoves>
        <subCacheUpdates>0</subCacheUpdates>
        <subCacheItems>0</subCacheItems>
</orion>

# Check iot agent iotacpp
$ docker exec -it docker_iotacpp_1 bash
[root@8e317c6b9405 /]# ps -elf
F S UID        PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
4 S root         1     0  0  80   0 -  4821 poll_s 15:49 ?        00:00:00 /sbin/init
1 S root        74     1  0  80   0 -  2673 poll_s 15:49 ?        00:00:00 /sbin/
↪udevd -d
5 S iotagent   292     1  0  80   0 - 12087 poll_s 15:49 ?        00:00:00 /usr/sbin/
↪mosquitto -d -c /etc/iot/mosquitto.conf
0 S iotagent   312     1  0  80   0 - 186499 futex_ 15:49 ?       00:00:00 /usr/local/
↪iot/bin/iotagent -n IoTPlatform -v ERROR -i 0.0.0.0 -p 8080 -d /usr/local/iot/lib -
↪c /etc/iot/config.json
0 S root       365     1  0  80   0 -  1028 hrtime 15:50 ?        00:00:00 /sbin/
↪mingetty /dev/tty[1-6]
4 S root       366     0  1  80   0 - 27087 wait   15:50 ?        00:00:00 bash
0 R root       378   366  0  80   0 - 27557 -      15:50 ?        00:00:00 ps -elf
[root@8e317c6b9405 /]# curl -g -X GET http://127.0.0.1:8080/iot/about
Welcome to IoTAgents  identifier:IoTPlatform:8080  1.3.0 commit 128.g14ee433 in Oct␣
↪28 2015
[root@8e317c6b9405 /]#
```

```
# and from outside
curl -g -X GET http://sensors.geonovum.nl:8000/iot/about
Welcome to IoTAgents  identifier:IoTPlatform:8080  1.3.0 commit 128.g14ee433 in Oct␣
↪28 2015

# Get log output
# See https://docs.docker.com/v1.5/compose/cli
$ docker-compose -f iotarush.yaml logs
```

Finally a custom Dockerfile was created for OCB to include Rush (and Redis) as an extension of the `fiware/orion` Docker file to support HTTPS notifications for OCB subscriptions (see WireCloud below). This is the content of the `geonovum/orionrush` Dockerfile:

```
FROM fiware/orion
MAINTAINER Just van den Broecke for Geonovum  www.geonovum.nl

ENV MONGODB_HOST mongodb
ENV REDIS_VERSION 2.8.3
ENV RUSH_VERSION 1.8.3
ENV RUSH_LISTEN_PORT 5001

WORKDIR /opt

RUN \
        # Dependencies
        yum -y install npm  && \
        # yum -y install redis   (gave errors see below)
        yum -y install wget  && \
        wget http://download.redis.io/releases/redis-${REDIS_VERSION}.tar.gz  && \
        tar xzvf redis-${REDIS_VERSION}.tar.gz  && \
        cd redis-${REDIS_VERSION} && \
        make && \
        make install && \
        # /etc/init.d/redis start  (no daemon installed)
        # Rush
        cd /opt && \
        curl https://codeload.github.com/telefonicaid/Rush/tar.gz/${RUSH_VERSION} >␣
↪Rush-${RUSH_VERSION}.tar.gz  && \
        tar xzvf Rush-${RUSH_VERSION}.tar.gz  && \
        cd Rush-${RUSH_VERSION} && \
        npm install --production
        # set mongodb host as linked in docker-compose iota.yml
        # export RUSH_GEN_MONGO=${MONGODB_HOST}
        # bin/listener
        # bin/consumer

WORKDIR /

COPY docker-entrypoint.sh /
COPY server.key /
COPY server.crt /
COPY server.csr /

RUN chmod +x /docker-entrypoint.sh
ENTRYPOINT ["/docker-entrypoint.sh"]

EXPOSE 1026
```

The entry point was changed to :

```
#!/bin/bash

REDIS_HOME=/opt/redis-${REDIS_VERSION}
RUSH_HOME=/opt/Rush-${RUSH_VERSION}

cd ${REDIS_HOME}
redis-server redis.conf > /var/log/redis.log 2>&1 &

cd ${RUSH_HOME}
export RUSH_GEN_MONGO=${MONGODB_HOST}
mv /server.key /server.crt /server.csr ${RUSH_HOME}/utils

bin/listener > /var/log/rush-listener.log 2>&1 &
bin/consumer > /var/log/rush-consumer.log 2>&1 &

/usr/bin/contextBroker -dbhost ${MONGODB_HOST} -fg -multiservice -rush localhost:5001␣
↪-logDir /var/log/contextBroker
```

To run the entire Fiware suite with *IoTAgentCpp*, *OrionRush*, *STH* and *MongoDB*, a new docker-compose file was created, `iotarush.yaml` :

```
#mongodbdata:
#    image: mongo:2.6
#    volumes:
#        - /data/db
#    restart: "no"
#    command: /bin/echo "Data-only container for mongodb."

#mongodb:
#    image: mongo:2.6
#    volumes_from:
#        - mongodbdata
#    expose:
#        - "27017"
#    command: --smallfiles

mongodb:
    image: mongo:2.6

    # Port mapping: allow only access on local docker host
    ports:
      - "127.0.0.1:27017:27017"
    expose:
      - "27017"

    # Use storage on host file system
    volumes:
      - /var/lib/mongodb_docker:/data/db
    command: --smallfiles

    # restart: always

orion:
    image: geonovum/orionrush
    links:
```

```
        - mongodb
    ports:
        - "1026:1026"

sthcomet:
    image: geonovum/sthcomet
    log_driver: "syslog"
    links:
        - mongodb
        - orion
    ports:
        - "8666:8666"

iotacpp:
    image: telefonicaiot/iotacpp
    links:
        - mongodb
        - orion
    ports:
        - "185.21.189.59:8000:8080"
        - "185.21.189.59:8081:8081"
        - "185.21.189.59:1883:1883"
```

To start/stop all fresh the following init.d script was created:

```
#!/bin/sh
#
# init.d script with LSB support for entire FIWARE stack via Docker.
#
# Copyright (c) 2015 Just van den Broecke for Geonovum - <just@justobjects.nl>
#
# This is free software; you may redistribute it and/or modify
# it under the terms of the GNU General Public License as
# published by the Free Software Foundation; either version 2,
# or (at your option) any later version.
#
# This is distributed in the hope that it will be useful, but
# WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License with
# the Debian operating system, in /usr/share/common-licenses/GPL;  if
# not, write to the Free Software Foundation, Inc., 59 Temple Place,
# Suite 330, Boston, MA 02111-1307 USA
#
### BEGIN INIT INFO
# Provides:          iotarush
# Required-Start:    $network $local_fs $remote_fs
# Required-Stop:     $network $local_fs $remote_fs
# Should-Start:      $named
# Should-Stop:
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: IoT stack for FIWARE: IotAgent, Orion with MongoDB and Rush/
↪Redis via Docker(-compose)
# Description:       Manages run cycle for a set of Docker containers together␣
↪providing a FIWARE IoT stack.
```

```
### END INIT INFO

PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

NAME=iotarush
DESC=docker_composition
# Docker-compose file
CONF=/opt/geonovum/sospilot/git/src/fiware/docker/iotarush.yaml
SERVICE_CONTAINERS="docker_iotacpp_1 docker_orion_1 docker_sthcomet_1 docker_mongodb_1
↪"

. /lib/lsb/init-functions

set -e

purge_services() {
        docker rm -v ${SERVICE_CONTAINERS}
    return 0
}

purge_all() {
        # Removes all containers including volume-data for mongodb!
        docker rm -v $(docker ps -a -q -f status=exited)
    return 0
}

start_containers() {
        docker-compose -f ${CONF} up -d
        docker ps
    return 0
}

stop_containers() {
        docker-compose -f ${CONF} stop
        docker ps
    return 0
}

case "$1" in
  start)
        log_daemon_msg "Starting $DESC" "$NAME"
        start_containers
        ;;

  stop)
         log_daemon_msg "Stopping $DESC" "$NAME"
         stop_containers
         purge_all
         ;;

  restart)
        log_daemon_msg "Restarting $DESC" "$NAME"
                $0 stop
                $0 start
        ;;

  status)
        log_daemon_msg "Checking status of $DESC" "$NAME"
```

```
        docker ps
        ;;

  reinit)
        log_warning_msg "Reinit of $NAME ; all mongodb data will be cleared!!!"
                stop_containers
                purge_all
                /bin/rm -rf /var/lib/mongodb_docker/*
                $0 start
        ;;

  *)
        N=/etc/init.d/$NAME
        echo "Usage: $N {start|stop|restart|reinit|status}" >&2
        exit 1
        ;;
esac

exit 0
```

Activating our `iotarush` init.d script and showing commands

```
$ cp iotarush /etc/init.d
$ update-rc.d iotarush  defaults

# Start
service iotarush start

# Status
service iotarush status

# Stop without loosing data
service iotarush stop

# Restart without loosing data
service iotarush restart

# Restart with purging all (mongodb) data
service iotarush reinit
```

12.5 Testing the FIWARE Installation

Using test clients at See test clients at https://github.com/Geonovum/sospilot/tree/master/src/fiware/client and the WireCloud Mashup.

12.5.1 Testing IoTAgent-Orion

Simple scenario, using the UltraLight (UL2.0) IoT protocol.:

1. create IoT service via IoTAgent (IDAS)
2. create IoT device via IoTAgent (IDAS)
3. observe new Entity in Orion
4. send temperature via IDAS

5. observe changed Entity via Orion

See test clients at https://github.com/Geonovum/sospilot/tree/master/src/fiware/client/python/UL20

See tutorial at: http://www.slideshare.net/FI-WARE/fiware-iotidasintroul20v2. Documentation https://fiware-orion.readthedocs.org/en/develop (OCB).

Sending commands from local system using FIWARE FIGWAY (python-IDAS4): https://github.com/Geonovum/sospilot/tree/master/src/fiware/client/python/UL20. These are a set of Python commands for most common REST services for both the OCB and IoTAgent/Manager.

Prepare the right `config.ini` used by all Python commands:

```
[user]
# Please, configure here your username at FIWARE Cloud and a valid Oauth2.0 TOKEN for␣
↪your user (you can use get_token.py to obtain a valid TOKEN).
username=<your username>
token=<your token>

[contextbroker]
# host=130.206.80.40
host=sensors.geonovum.nl
port=1026
OAuth=no
# Here you need to specify the ContextBroker database you are querying.
# Leave it blank if you want the general database or the IDAS service if you are␣
↪looking for IoT devices connected by you.
fiware_service=fiwareiot
# fiware_service=bus_auto
fiware-service-path=/

[idas]
host=sensors.geonovum.nl
# host=130.206.80.40
#adminport=5371
#ul20port=5371
adminport=443
ul20port=443
OAuth=no
# Here you need to configure the IDAS service your devices will be sending data to.
# By default the OpenIoT service is provided.
#
fiware-service=fiwareiot
# fiware-service=workshop
# fiware-service=bus_auto
fiware-service-path=/
#apikey=4jggokgpepnvsb2uv4s40d59ov
apikey=4jggokgpepnvsb2uv4s40d59ov

[local]
#Choose here your System type. Examples: RaspberryPI, MACOSX, Linux, ...
host_type=MACOSX
# Here please add a unique identifier for you. Suggestion: the 3 lower hexa bytes of␣
↪your Ethernet MAC. E.g. 79:ed:af
# Also you may use your e-mail address.
host_id=a0:11:00
```

Create device template (called `GEONOVUM_TEMP`) under https://github.com/telefonicaid/fiware-figway/tree/master/python-IDAS4/Sensors_UL20/devices :

```
        {
         "devices": [
            { "device_id": "DEV_ID",
              "entity_name": "ENTITY_ID",
              "entity_type": "thing",
              "protocol": "PDI-IoTA-UltraLight",
              "timezone": "Europe/Amsterdam",
        "attributes": [
                {
                  "object_id": "temp",
                  "name": "temperature",
                  "type": "int"
                },
            {
                "object_id": "pos",
            "name": "position",
            "type": "coords",
            "metadatas": [
                {
                    "name": "location",
                    "type": "string",
                    "value": "WGS84"
                }
              ]
            }
              ],
         "static_attributes": [
                { "name": "organization",
                  "type": "string",
                  "value": "Geonovum"
                }
               ]
              }
             ]
            }
```

Create service, then a device and send an observation using Python code under https://github.com/Geonovum/sospilot/tree/master/src/fiware/client/python/UL20 (IoTAgent with UL protocol) and https://github.com/Geonovum/sospilot/tree/master/src/fiware/client/python/ContextBroker (OCB).

Watch that the related Entity is created in the OCB and that it gets an attribute value when sending an Observation to the IoTAgent.

```
    # IoTAgent: List devices (none present)
    python ListDevices.py
    * Asking to http://sensors.geonovum.nl:8000/iot/devices
    * Headers: {'Fiware-Service': 'fiwareiot', 'content-type': 'application/json',
↪'Fiware-ServicePath': '/', 'X-Auth-Token': 'NULL'}
    ...

    * Status Code: 200
    * Response:
    { "count": 0,"devices": []}

    # OCB: Show Entities ot OCB (none present)
    $ python GetEntities.py ALL
    * Asking to http://sensors.geonovum.nl:1026/ngsi10/queryContext
    * Headers: {'Fiware-Service': 'fiwareiot', 'content-type': 'application/json',
↪'accept': 'application/json', 'X-Auth-Token': 'NULL'}
```

```
    * Sending PAYLOAD:
    {
        "entities": [
            {
                "type": "",
                "id": ".*",
                "isPattern": "true"
            }
        ],
        "attributes": []
    }

    # IoTAgent: Create an IoT service
    $ python CreateService.py fiwareiot 4jggokgpepnvsb2uv4s40d59ov 185.21.189.59 1026
    * Asking to http://sensors.geonovum.nl:8000/iot/services
    * Headers: {'Fiware-Service': 'fiwareiot', 'content-type': 'application/json',
↪'Fiware-ServicePath': '/', 'X-Auth-Token': 'NULL'}
    * Sending PAYLOAD:
    {
        "services": [
            {
                "token": "token2",
                "apikey": "4jggokgpepnvsb2uv4s40d59ov",
                "resource": "/iot/d",
                "entity_type": "thing",
                "cbroker": "http://185.21.189.59:1026"
            }
        ]
    }

    # IoTAgent: Register a Device passing related Entity name
    $ python RegisterDevice.py OTTERLO_TEMP NexusProDev WeatherOtterloEnt
    * opening: ./devices/OTTERLO_TEMP
    * Asking to http://sensors.geonovum.nl:8000/iot/devices
    * Headers: {'Fiware-Service': 'fiwareiot', 'content-type': 'application/json',
↪'Fiware-ServicePath': '/', 'X-Auth-Token': 'NULL'}
    * Sending PAYLOAD:
    {
        "devices": [
            {
                "protocol": "PDI-IoTA-UltraLight",
                "entity_name": "WeatherOtterloEnt",
                "entity_type": "thing",
                "static_attributes": [
                    {
                        "type": "string",
                        "name": "location",
                        "value": "BosHut"
                    }
                ],
                "timezone": "Europe/Amsterdam",
                "attributes": [
                    {
                        "type": "int",
                        "name": "temperature",
                        "object_id": "ot"
                    }
                ],
```

```
                "device_id": "NexusProDev"
            }
        ]
    }

    ...

    * Status Code: 201

# IoTAgent: List the newly added device
    $ python ListDevices.py
    * Asking to http://sensors.geonovum.nl:8000/iot/devices
    * Headers: {'Fiware-Service': 'fiwareiot', 'content-type': 'application/json',
↪'Fiware-ServicePath': '/', 'X-Auth-Token': 'NULL'}
    ...

    * Status Code: 200
    * Response:
    { "count": 1,"devices": [{ "device_id" : "NexusProDev" }]}

    # OCB: Show related Entity in OCB
    $ python GetEntities.py ALL
    * Asking to http://sensors.geonovum.nl:1026/ngsi10/queryContext
    * Headers: {'Fiware-Service': 'fiwareiot', 'content-type': 'application/json',
↪'accept': 'application/json', 'X-Auth-Token': 'NULL'}
    * Sending PAYLOAD:
    {
        "entities": [
            {
                "type": "",
                "id": ".*",
                "isPattern": "true"
            }
        ],
        "attributes": []
    }

    ...

    * Status Code: 200
    ***** Number of Entity Types: 1

    ***** List of Entity Types
    <entityId type="thing" isPattern="false"> : 1

    **** Number of Entity IDs: 1

    **** List of Entity IDs
    <id>WeatherOtterloEnt< : 1

    Do you want me to print all Entities? (yes/no)yes
    <queryContextResponse>
      <contextResponseList>
        <contextElementResponse>
          <contextElement>
            <entityId type="thing" isPattern="false">
              <id>WeatherOtterloEnt</id>
            </entityId>
```

```
            <contextAttributeList>
              <contextAttribute>
                <name>TimeInstant</name>
                <type>ISO8601</type>
                <contextValue>2015-10-30T20:21:17.557970</contextValue>
              </contextAttribute>
              <contextAttribute>
                <name>location</name>
                <type>string</type>
                <contextValue>BosHut</contextValue>
                <metadata>
                  <contextMetadata>
                    <name>TimeInstant</name>
                    <type>ISO8601</type>
                    <value>2015-10-30T20:21:17.558093</value>
                  </contextMetadata>
                </metadata>
              </contextAttribute>
            </contextAttributeList>
          </contextElement>
          <statusCode>
            <code>200</code>
            <reasonPhrase>OK</reasonPhrase>
          </statusCode>
        </contextElementResponse>
      </contextResponseList>
    </queryContextResponse>

    # IoTAgent: Send an Observation to device
    $ python SendObservation.py NexusProDev 'ot|16'
    * Asking to http://sensors.geonovum.nl:8000/iot/d?k=4jggokgpepnvsb2uv4s40d59ov&
↪i=NexusProDev
    * Headers: {'Fiware-Service': 'fiwareiot', 'content-type': 'application/json',
↪'Fiware-ServicePath': '/', 'X-Auth-Token': 'NULL'}
    * Sending PAYLOAD:
    ot|16

    ...

    * Status Code: 200
    * Response:

    # OCB: See value in Entity
python GetEntities.py ALL
* Asking to http://sensors.geonovum.nl:1026/ngsi10/queryContext
* Headers: {'Fiware-Service': 'fiwareiot', 'content-type': 'application/json', 'accept
↪': 'application/json', 'X-Auth-Token': 'NULL'}
* Sending PAYLOAD:
{
    "entities": [
        {
            "type": "",
            "id": ".*",
            "isPattern": "true"
        }
    ],
    "attributes": []
}
```

```
...

* Status Code: 200
***** Number of Entity Types: 1

***** List of Entity Types
<entityId type="thing" isPattern="false"> : 1

**** Number of Entity IDs: 1

**** List of Entity IDs
<id>WeatherOtterloEnt< : 1

Do you want me to print all Entities? (yes/no)yes
<queryContextResponse>
  <contextResponseList>
    <contextElementResponse>
      <contextElement>
        <entityId type="thing" isPattern="false">
          <id>WeatherOtterloEnt</id>
        </entityId>
        <contextAttributeList>
          <contextAttribute>
            <name>TimeInstant</name>
            <type>ISO8601</type>
            <contextValue>2015-10-30T21:04:13.770532</contextValue>
          </contextAttribute>
          <contextAttribute>
            <name>location</name>
            <type>string</type>
            <contextValue>BosHut</contextValue>
            <metadata>
              <contextMetadata>
                <name>TimeInstant</name>
                <type>ISO8601</type>
                <value>2015-10-30T21:04:13.770563</value>
              </contextMetadata>
            </metadata>
          </contextAttribute>
          <contextAttribute>
            <name>temperature</name>
            <type>int</type>
            <contextValue>16</contextValue>
            <metadata>
              <contextMetadata>
                <name>TimeInstant</name>
                <type>ISO8601</type>
                <value>2015-10-30T21:04:13.770532</value>
              </contextMetadata>
            </metadata>
          </contextAttribute>
        </contextAttributeList>
      </contextElement>
      <statusCode>
        <code>200</code>
        <reasonPhrase>OK</reasonPhrase>
      </statusCode>
```

```
    </contextElementResponse>
  </contextResponseList>
</queryContextResponse>

# Get Context Types, note: --header 'Fiware-Service: fiwareiot' needs to be present!!
    $ curl sensors.geonovum.nl:1026/v1/contextTypes -S --header 'Accept: application/
↪json' --header 'Fiware-Service: fiwareiot'
    {
      "types" : [
        {
          "name" : "thing",
          "attributes" : [
            "temperature",
            "location",
            "TimeInstant"
          ]
        }
      ],
      "statusCode" : {
        "code" : "200",
        "reasonPhrase" : "OK"
      }
    }

    # Get Context Entities, note: --header 'Fiware-Service: fiwareiot' needs to be␣
↪present!!
    $ curl sensors.geonovum.nl:1026/v1/contextEntities -S --header 'Accept:␣
↪application/json' --header 'Fiware-Service: fiwareiot'
    {
      "contextResponses" : [
        {
          "contextElement" : {
            "type" : "thing",
            "isPattern" : "false",
            "id" : "WeatherOtterloEnt",
            "attributes" : [
              {
                "name" : "TimeInstant",
                "type" : "ISO8601",
                "value" : "2015-10-31T12:55:28.157330"
              },
              {
                "name" : "location",
                "type" : "string",
                "value" : "BosHut",
                "metadatas" : [
                  {
                    "name" : "TimeInstant",
                    "type" : "ISO8601",
                    "value" : "2015-10-31T12:55:28.157371"
                  }
                ]
              },
              {
                "name" : "temperature",
                "type" : "int",
                "value" : "11",
                "metadatas" : [
```

```
                {
                  "name" : "TimeInstant",
                  "type" : "ISO8601",
                  "value" : "2015-10-31T12:55:28.157330"
                }
              ]
            }
          ]
        },
        "statusCode" : {
          "code" : "200",
          "reasonPhrase" : "OK"
        }
      }
    ]
  }
```

12.5.2 Testing with MQTT Client

The IoTAgent also supports the MQTT protocol: http://mqtt.org

MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport.

We will use Mosquitto as MQTT-client first:

Mosquitto is an open source (BSD licensed) message broker that implements the MQ Telemetry Transport protocol versions 3.1 and 3.1.1. MQTT provides a lightweight method of carrying out messaging using a publish/subscribe model.

See test clients at https://github.com/Geonovum/sospilot/tree/master/src/fiware/client/python/MQTT

Before observations can be sent a Service needs to be created and a Device(s) registered.

On Mac OSX install Mosquitto via HomeBrew:

```
$ brew install mosquitto
==> Installing dependencies for mosquitto: c-ares, libwebsockets
==> Installing mosquitto dependency: c-ares
==> Downloading https://homebrew.bintray.com/bottles/c-ares-1.10.0.mavericks.bottle.
↪tar.gz
```

Use `mosquitto_pub` as a commandline client http://mosquitto.org/man/mosquitto_pub-1.html for initial tests.

```
$ mosquitto_pub -d -h sensors.geonovum.nl -p 1883 -t sensors/temperature -m
↪"1266193804 32"
Client mosqpub/18773-sunda sending CONNECT
Client mosqpub/18773-sunda received CONNACK
Client mosqpub/18773-sunda sending PUBLISH (d0, q0, r0, m1, 'sensors/temperature', ...
↪ (13 bytes))
Client mosqpub/18773-sunda sending DISCONNECT
```

See `setup.sh` for full example with Service/Device creation and observation publication via MQTT :

```
#!/bin/bash

# see https://github.com/telefonicaid/fiware-IoTAgent-Cplusplus/blob/develop/doc/MQTT_
↪protocol.md
```

```
# provisioning API: https://github.com/telefonicaid/fiware-IoTAgent-Cplusplus/blob/
↪develop/doc/north_api.md

DEVICE_ID=DavisMQTTDev
ORION_ENTITY=TempGeonovumMQTTEnt
# API_KEY=4jggokgpepnvsb2uv4s40d59ov
API_KEY=7qqa9uvkgketabc8ui4knu1onv
SERVICE_ID=fiwareiot
ORION_HOST=185.21.189.59
ORION_PORT=1026
MQTT_HOST=sensors.geonovum.nl
MQTT_PORT=1883
DEVICE_TEMPLATE=GEONOVUM_MQTT_TEMP

# Create the service
python CreateService.py ${SERVICE_ID} ${API_KEY} ${ORION_HOST} ${ORION_PORT}

# Provision Device
python RegisterDevice.py ${DEVICE_TEMPLATE} ${DEVICE_ID} ${ORION_ENTITY}

# Publish observations
# mosquitto_pub -h $HOST_IOTAGENT_MQTT -t <api_key>/mydevicemqtt/t -m 44.4 -u <api_
↪key>
mosquitto_pub -r -d -h ${MQTT_HOST} -p ${MQTT_PORT} -u ${API_KEY} -t ${API_KEY}/$
↪{DEVICE_ID}/temp -m 11
mosquitto_pub -r -d -h ${MQTT_HOST} -p ${MQTT_PORT} -u ${API_KEY} -t ${API_KEY}/$
↪{DEVICE_ID}/pos -m '52.152435,5.37241'
```

and a sample device file:

```
        {
         "devices": [
            { "device_id": "DEV_ID",
              "entity_name": "ENTITY_ID",
              "entity_type": "thing",
              "protocol": "PDI-IoTA-MQTT-UltraLight",
              "timezone": "Europe/Amsterdam",
        "attributes": [
                {
                  "object_id": "temp",
                  "name": "temperature",
                  "type": "int"
                },
            {
                "object_id": "pos",
            "name": "position",
            "type": "coords",
            "metadatas": [
                {
                    "name": "location",
                    "type": "string",
                    "value": "WGS84"
                }
              ]
            }
              ],
         "static_attributes": [
                { "name": "organization",
```

```
                "type": "string",
                "value": "Geonovum"
              }
            ]
          }
        ]
      }
```

The Service creation and Device provisioning uses the Admin API of the IotAgentCpp server. MQTT is only used to send observations. See also http://fiware-iot-stack.readthedocs.org/en/latest/device_api/index.html

The helper .py programs are ported from FIWARE FIGWAY Sensors_UL20 code.

TWO VERY IMPORTANT DIFFERENCES WITH UL20:

- in the payload when creating the Service the following field needs to be set: `"resource": "/iot/mqtt"` otherwise the Device cannot be registered (*"protocol is not correct"* error).
- Also in the Device Template (see here under devices/) the "protocol": `"PDI-IoTA-MQTT-UltraLight"` needs to be present.

See also this issue: https://github.com/telefonicaid/fiware-IoTAgent-Cplusplus/issues/254

12.5.3 Inspect data in MongoDB

Within the `mongodb` Docker container we can inspect the persisted data in the Mongo shell : https://docs.mongodb.org/manual/reference/mongo-shell.

```
$ docker exec -it docker_mongodb_1 bash
root@43fd245b67ca:/# mongo
MongoDB shell version: 2.6.11
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
> show dbs
admin             (empty)
iot               0.031GB
local             0.031GB
orion             0.031GB
orion-fiwareiot   0.031GB
sth_fiwareiot     0.031GB

> use iot
switched to db iot
> show collections
COMMAND
DEVICE
PROTOCOL
SERVICE
SERVICE_MGMT
system.indexes
> use orion
switched to db orion
> show collections
```

```
entities
system.indexes
> db.entities.find()
> use orion-fiwareiot
switched to db orion-fiwareiot
> db.entities.find()
{ "_id" : { "id" : "WeatherOtterloEnt", "type" : "thing", "servicePath" : "/" },
"attrNames" : [ "TimeInstant", "location", "temperature" ],
"attrs" : { "TimeInstant" : { "value" : "2015-10-31T20:41:28.654329", "type" :
↪"ISO8601", "creDate" : 1446324088, "modDate" : 1446324088 },
"location" : { "value" : "BosHut", "type" : "string", "md" : [ { "name" : "TimeInstant
↪", "type" : "ISO8601", "value" : "2015-10-31T20:41:28.654370" } ],
"creDate" : 1446324088, "modDate" : 1446324088 },
"temperature" : { "value" : "11", "type" : "int", "md" : [ { "name" : "TimeInstant",
↪"type" : "ISO8601", "value" : "2015-10-31T20:41:28.654329" } ],
"creDate" : 1446324088, "modDate" : 1446324088 } }, "creDate" : 1446324088, "modDate"␣
↪: 1446324088 }

# timeseries storage
> use sth_fiwareiot
switched to db sth_fiwareiot
> show collections
sth_/_dust13ent_thing
sth_/_dust13ent_thing.aggr
sth_/_nexusent1_thing
sth_/_nexusent1_thing.aggr
sth_/_tempgeonovument_thing
sth_/_tempgeonovument_thing.aggr
sth_/_thing:dust13_thing
sth_/_thing:dust13_thing.aggr
system.indexes
```

12.5.4 Display Values with WireCloud

WireCloud http://conwet.fi.upm.es/wirecloud/ is a Mashup framework within FIWARE with instance at FIWARE Lab: https://mashup.lab.fiware.org

Here we can create Widgets to get data from the Orion CB, so indirectly observations sent to the IoTAgent from our devices.

First Steps

Trying simple the NGSI Browser, but did not succeed (help mail sent to `fiware-lab-help@lists.fiware.org`:

```
Trying to connect to my OCB  which has entities created via IDAS. Both are of latest␣
↪Docker version.
Works fine using the FIGWAY Python scripts.

But using any Mashup widget that does requests to the OCB like the NGSI Browser the␣
↪widget remans blanc,
since the OCB sends back:

{
  "errorCode" : {
```

```
    "code" : "404",
    "reasonPhrase" : "No context element found"
  }
}

This reply is also received when querying via curl:
curl <my_ocb_host>:1026/v1/contextEntities -S --header 'Accept: application/json'

But if I add the header  --header 'Fiware-Service: fiwareiot' (which was specified␣
↪when creating the IoT
service w IDAS) then I get expected responses from the OCB.

However the Widgets, Operators in WC have no means to add the 'Fiware-Service' Header.
```

This problem was also posted at StackOverflow.

A quick local solution is to manually add the HTTP header using the browser's Developer Tools to a WireCloud Widget JavaScript (usually `main.js`) where the HTTP request to the Orion NGSI API is created. The WC NGSI connector supports adding extra HTTP headers as per the NGSI documentation. See for example here below where the Chrome developer tools is used to modify the NGSIConnection :

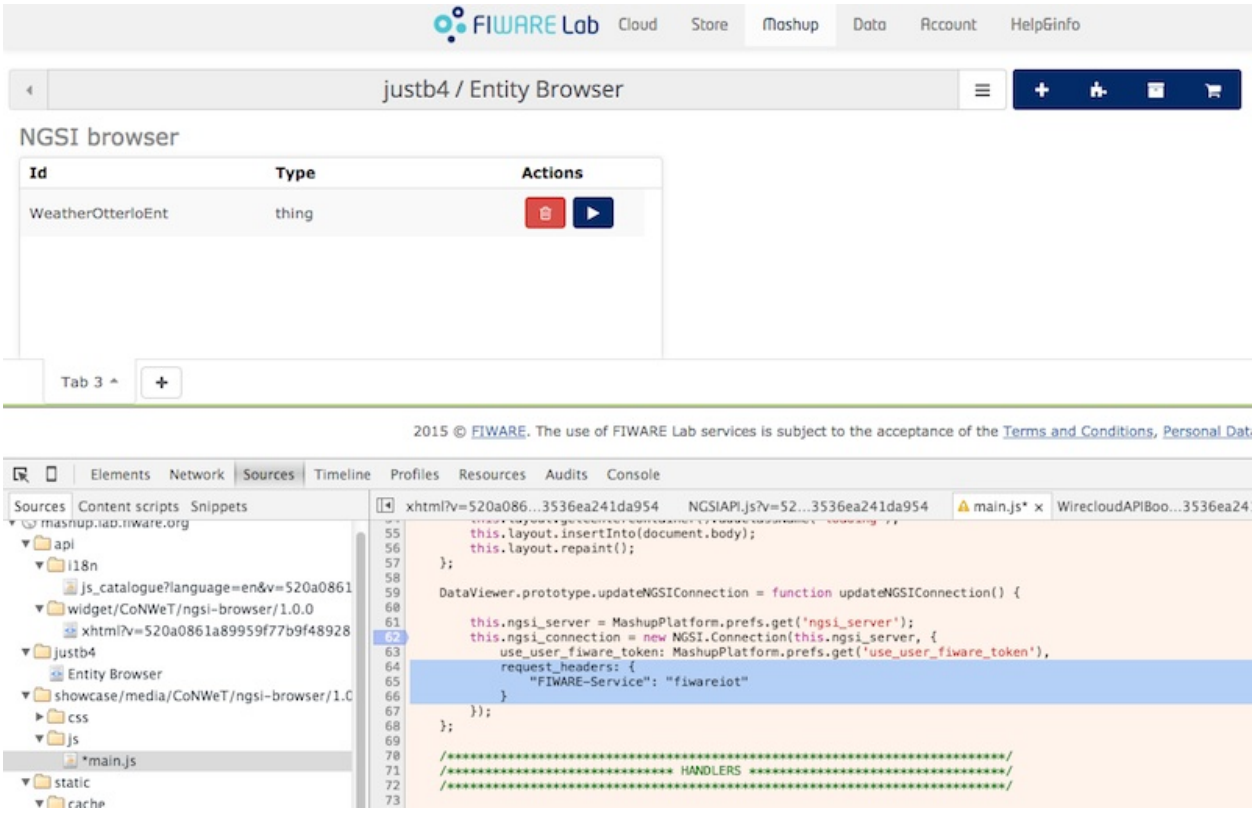


Fig. 12.2: *Quick Fix: modify the HTTP header locally in browser*

Another possibility is to download the widget, modify its config.xml and main.js to support configuring the FIWARE-Service header. This worked as well, but the real fixes should be done within the component source code. The issue affects all WC components (Operators, Widgets) that interact with Orion NSGI. The following issues have been opened:

- https://github.com/wirecloud-fiware/ngsi-browser-widget/issues/1 (fixed)

- https://github.com/wirecloud-fiware/ngsi-source-operator/issues/3
- https://github.com/wirecloud-fiware/ngsi-updater-widget/issues/1
- https://github.com/wirecloud-fiware/ngsi-type-browse-widget/issues/1

As the NGSI Browser Widget was fixed and a new version was available, a first test could be performed.

Temperature in POI on Map

Using the NGSI Browser Widget (fixed v1.0.1) with an NSGI Entity to POI Converter connected to a Map Widget the temperature could be made visible on a map. The result of the mashup is below.

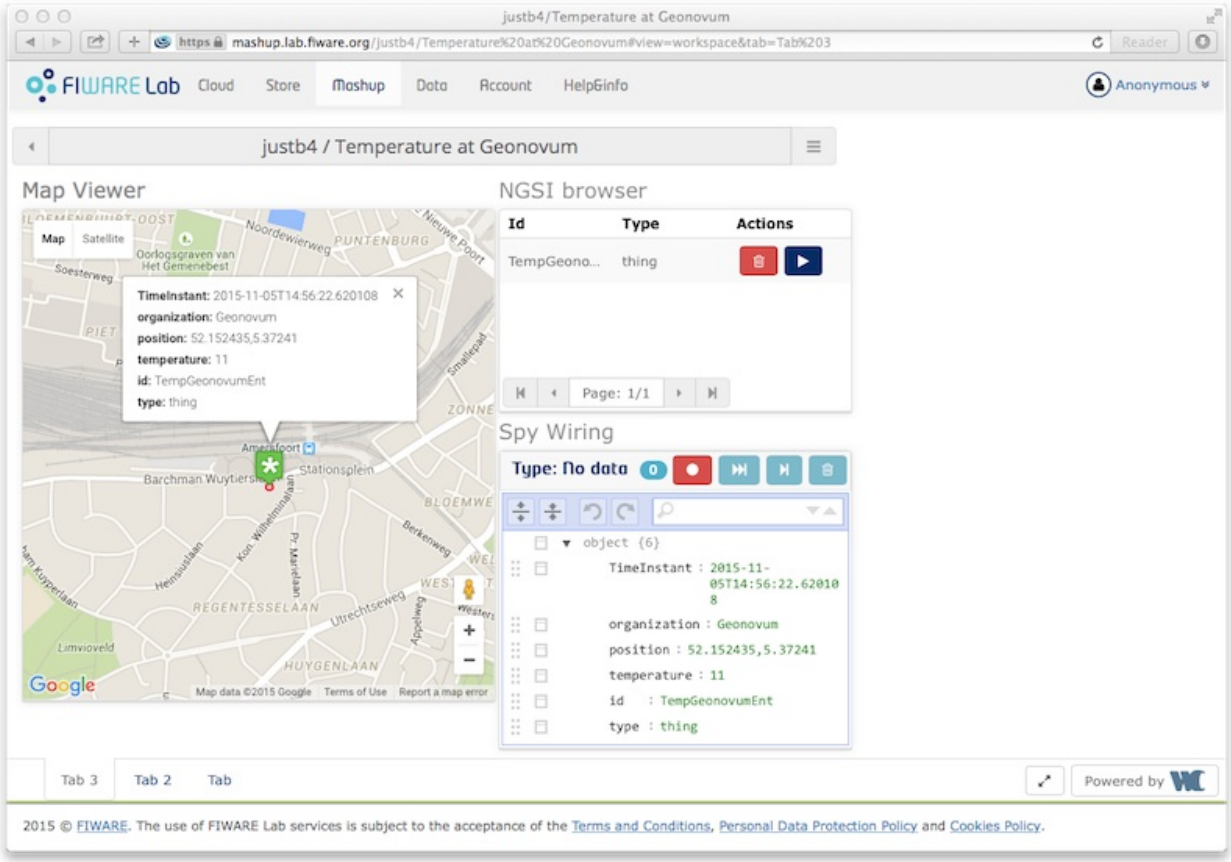


Fig. 12.3: *First WireCloud Mashup: show Temperature from sensor as POI*

The wiring for these components was as depicted below.

Next attempt was to use NGSI Subscriptions such that the widgets receive real-time updates. For this the NGSI Source Operator can be applied i.s.o. the `NGSI Browser Widget` used above. The `NGSI Source Operator` will use NGSI10 Subscriptions to register at an Orion CB using a callback mechanism. The wiring is depicted below.

The `NGSI Source Operator` was first fixed via issue 3 such that the `Fiware-service` HTTP-header could be applied. But since the Orion CB runs without HTTPS within a remote (Docker) service, callbacks via HTTPS did not work. Also using HTTP via proxy http://ngsiproxy.lab.fiware.org did not work because of browser security restrictions. These problems have been documented and discussed in this issue. Polling mode may be another solution but possibly too strenous.

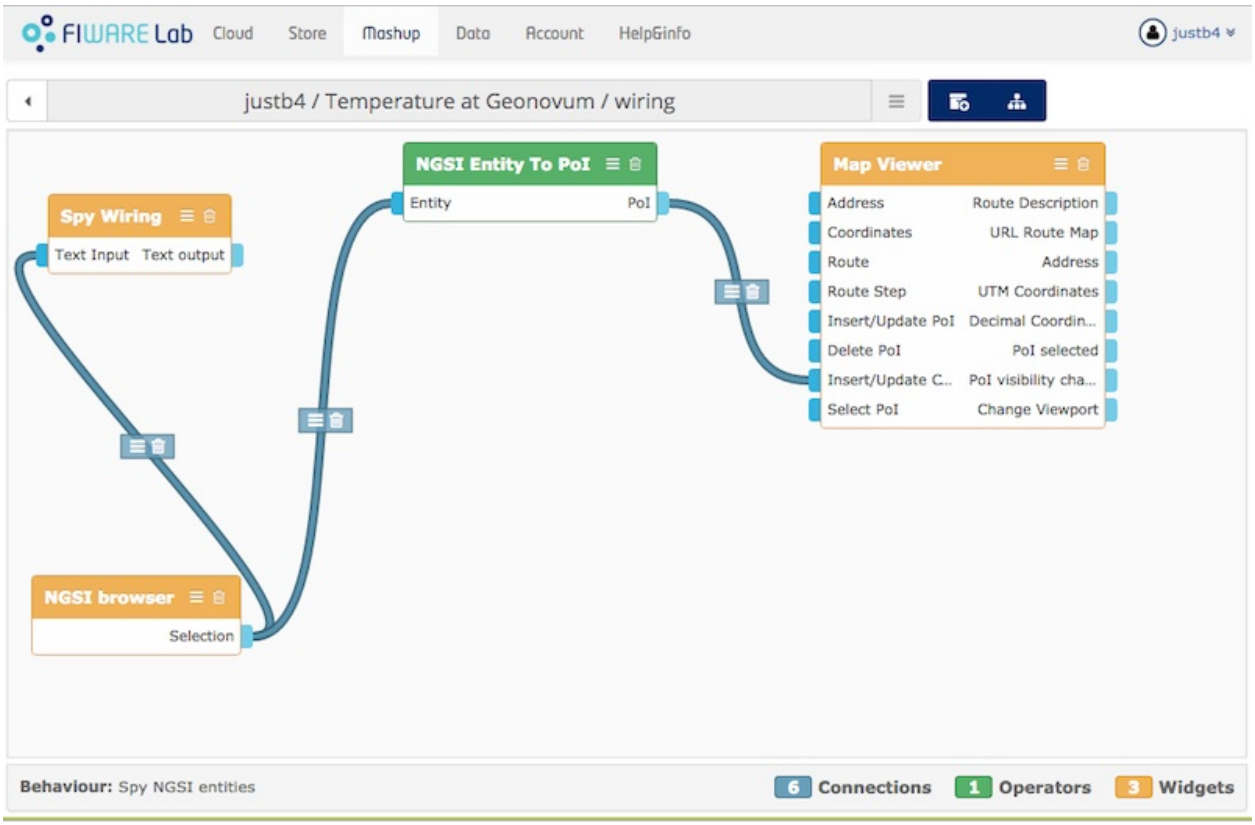


Fig. 12.4: *First WireCloud Mashup: wiring view in editor*

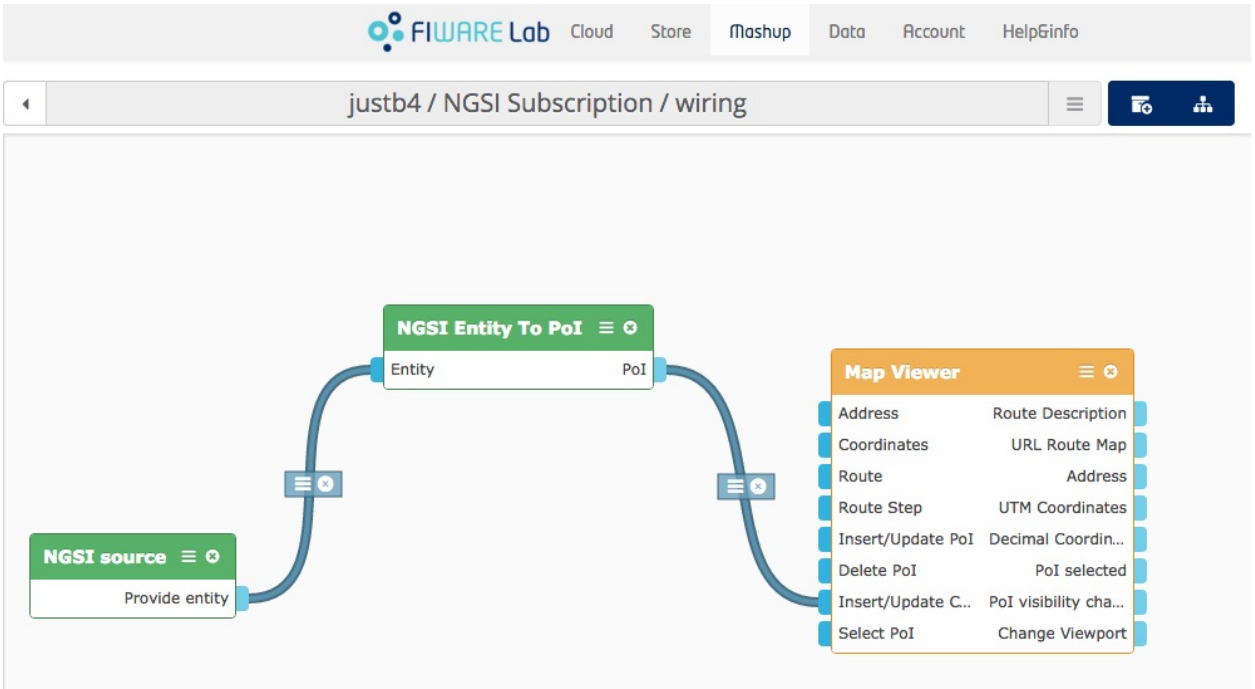


Fig. 12.5: *Second WireCloud Mashup: wiring view in editor*

Using an HTTPS middleman Relayer, Rush, is suggested as a solution. This was tried first. The final range of commands is:

```
# within docker_orion_1 running container

# Dependencies
cd /root
yum -y install npm
# yum -y install redis   (gave errors see below)
yum -y install wget
wget http://download.redis.io/releases/redis-2.8.3.tar.gz
tar xzvf redis-2.8.3.tar.gz
cd redis-2.8.3
make
make install
# /etc/init.d/redis start  (no daemon installed)
redis-server redis.conf &

# Rush
cd /root
curl https://codeload.github.com/telefonicaid/Rush/tar.gz/1.8.3 > Rush-1.8.3.tar.gz
tar xzvf Rush-1.8.3.tar.gz
cd Rush-1.8.3
npm install --production

# set mongodb host as linked in docker-compose iota.yml
export RUSH_GEN_MONGO=mongodb
bin/listener &
bin/consumer &
```

NB *yum -y install redis* installed Redis 2.4 but gave recurring error: `Generic Server Error: Error: ERR unknown command 'evalsha'`. Redis should be higher than v2.4 as stated in the solution here

Activate *Rush* in Orion within *iota.yaml* by setting the `orion` entry (`command` args) to:

```
.
orion:
    image: fiware/orion
    links:
        - mongodb
    ports:
        - "1026:1026"
    command: -dbhost mongodb -rush localhost:5001 -logDir /var/log/contextBroker
.
```

Now notifications are seen immediately on sending events from the UL20 client! See picture below:

12.5.5 Display with OpenLayers/Heron

In order to facilitate viewing data in standard geo-web viewers, an OpenLayers NGSI10 Vector Layer component was developed. This allows viewers like the existing project HeronViewer to show (real-time) sensor data.

Below is a screenshot of the HeronViewer showing in blue dots 2 “bot-sensors” and the CityGIS Dustduino-based sensor. All show a temperature in realtime.

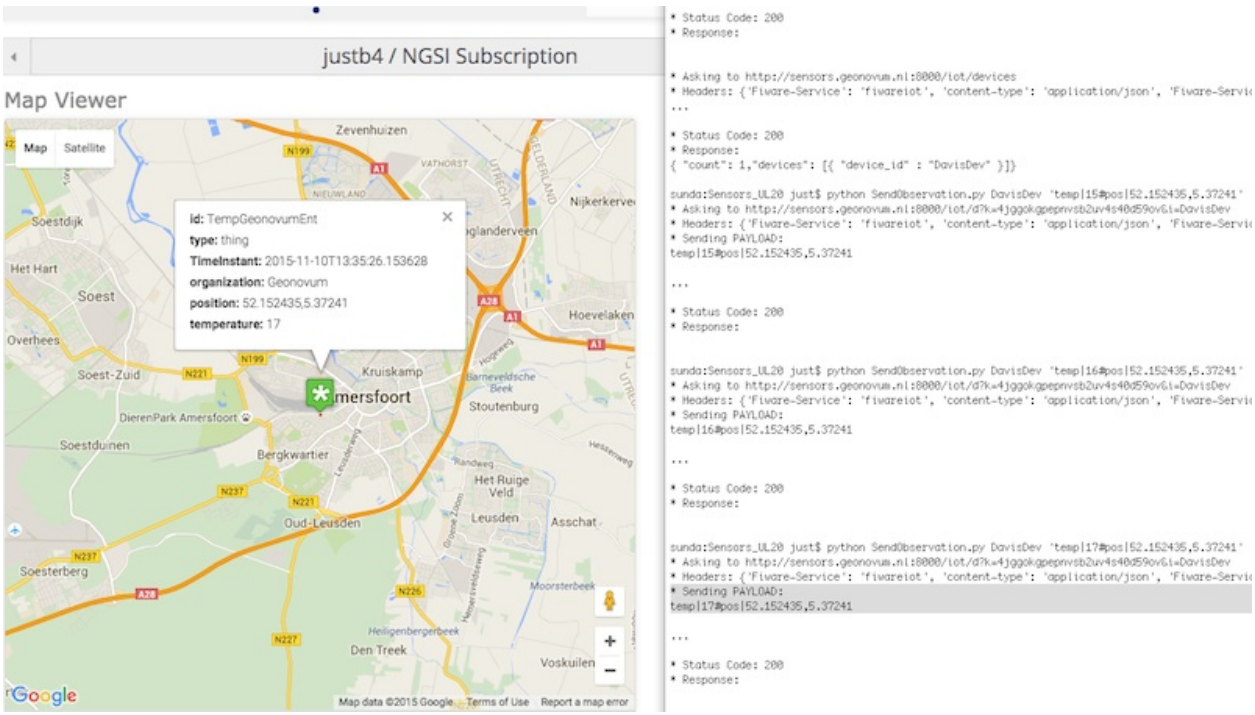


Fig. 12.6: *Second WireCloud Mashup: direct notifications in Map (left) from UL20 client (right)*

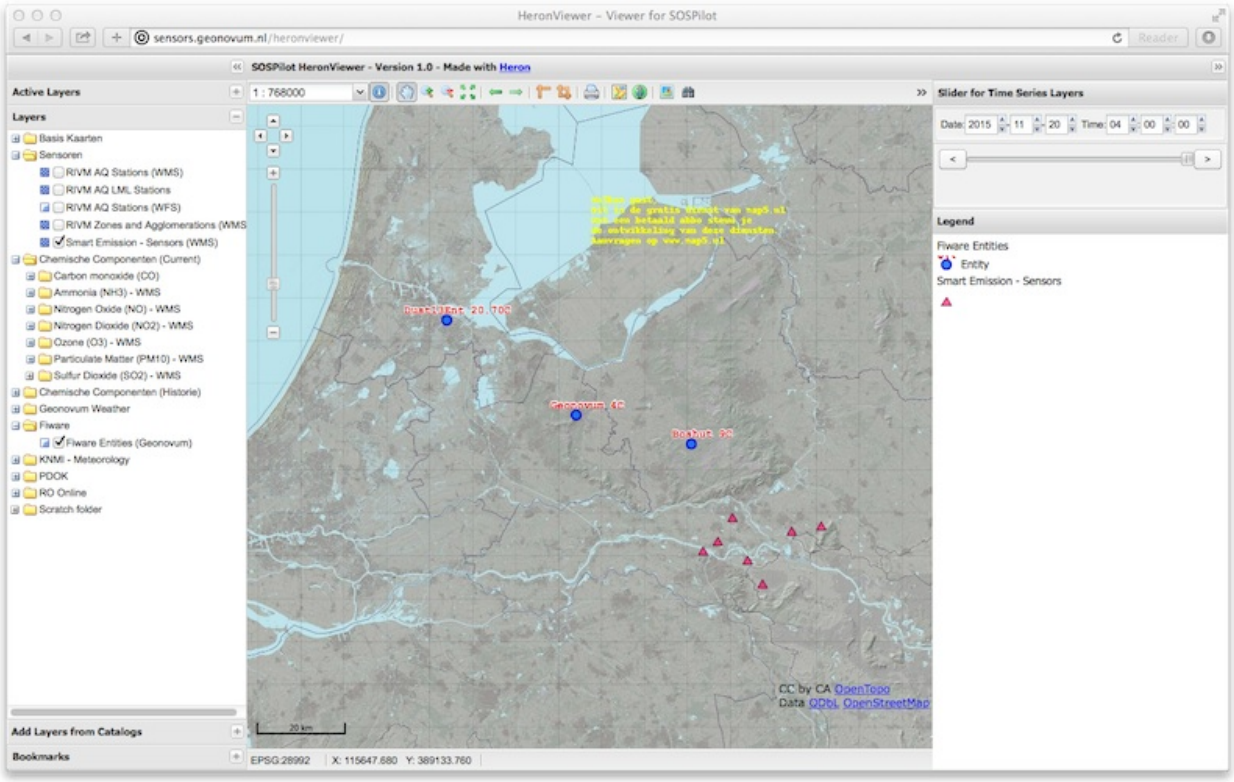


Fig. 12.7: *NGSI10 OpenLayers Vector Layer (blue dots) in HeronViewer*

12.5.6 Timeseries data from STH

The Short Term History (STH) component stores timeseries data. This is achieved by a `subscribeContext` on the Orion CB NGSI10 API. This has to be done explicitly. Documentation can be found here: https://github.com/telefonicaid/fiware-sth-comet

In our setup we fire a `curl` script to trigger a generic subscription on the `thing` entity:

```
#!/bin/bash

#POST /v1/subscribeContext HTTP/1.1
#Host: sensors.geonovum.nl:1026
#origin: https://mashup.lab.fiware.org
#Cookie: _ga=GA1.2.1632625772.1441807083, policy_cookie=on
#Content-Length: 257
#via: 1.1 mashup.lab.fiware.org (Wirecloud-python-Proxy/1.1)
#accept-language: en-US,en;q=0.8,de;q=0.6,fr;q=0.4,nl;q=0.2,it;q=0.2
#accept-encoding: gzip, deflate
#x-forwarded-host: sensors.geonovum.nl:1026
#x-forwarded-for: 82.217.164.50
#fiware-service: fiwareiot
#accept: application/json
#user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML,
↪ like Gecko) Chrome/46.0.2490.86 Safari/537.36
#connection: keep-alive
#x-requested-with: XMLHttpRequest
#referer: https://mashup.lab.fiware.org/justb4/NGSI%20Subscription
#content-type: application/json
#
#
#write error to stdout
#
#130.206.084.011.36914-185.021.189.059.01026:
#{"entities":[{"id":".*","isPattern":"true","type":"thing"}],
#"reference":"https://ngsiproxy.lab.fiware.org:443/callbacks/23:33:47-1:23:33:48-1",
#"duration":"PT3H",
#"notifyConditions":[
#{"type":"ONCHANGE","condValues":["position","temperature","organization"]
#}]}
#
#write error to stdout
#
#185.021.189.059.01026-130.206.084.011.36914: HTTP/1.1 200 OK
#Content-Length: 109
#Content-Type: application/json
#Date: Mon, 30 Nov 2015 21:31:14 GMT
#
#{
#  "subscribeResponse" : {
#    "subscriptionId" : "565cc0222b41bbad4c87656f",
#    "duration" : "PT3H"
#  }
#}

ORION_HOST=sensors.geonovum.nl
ORION_PORT=1026
STH_HOST=sensors.geonovum.nl
STH_PORT=8666
```

```
#  --header 'Fiware-ServicePath: /'
curl ${ORION_HOST}:${ORION_PORT}/v1/subscribeContext -s -S\
 --header 'Content-Type: application/json' \
 --header 'Accept: application/json' \
 --header 'fiware-service: fiwareiot' \
 -d @- <<EOF
{
    "entities": [
        {
            "type": "thing",
            "isPattern": "true",
            "id": ".*"
        }
    ],
    "reference": "http://sensors.geonovum.nl:8666/notify",
    "duration": "P1Y",
    "notifyConditions": [
        {
            "type": "ONCHANGE",
            "condValues": ["temperature", "humidity", "pm10", "pm2_5"]
        }
    ],
    "throttling": "PT5S"
}
EOF
```

This triggers a flow of data from the Orion CB via the STH to MongoDB. Via the STH REST API we can request timeseries data. As we need to send Fiware-HTTP headers we can invoke the STH API using a (Chrome) REST client as follows:

Fig. 12.8: *Request Timeseries Data from STH REST API*

12.6 Installing FIWARE - from Source

Done on the Linux VPS (Ubuntu 14.04).

Abandoned, Orion CB compiled and ran with mongodb, but IoTAgent gave core dump, using Docker now, but kept for reference.

12.6.1 Orion Context Broker (OCB)

Build from source

On 28.okt.2015. Version: `0.24.0-next` (git version: a938e68887fbc7070b544b75873af935d8c596ae). See https://github.com/telefonicaid/fiware-orion/blob/develop/doc/manuals/admin/build_source.md, but later found: https://github.com/telefonicaid/fiware-orion/blob/develop/scripts/bootstrap/ubuntu1404.sh

Install build deps.

```
apt-get  install cmake scons libmicrohttpd-dev libboost-all-dev
# what about libcurl-dev gcc-c++ ?
apt-get -y install make cmake build-essential scons git libmicrohttpd-dev libboost-
↪dev libboost-thread-dev libboost-filesystem-dev libboost-program-options-dev ␣
↪libcurl4-gnutls-dev clang libcunit1-dev mongodb-server python python-flask python-
↪jinja2 curl libxml2 netcat-openbsd mongodb valgrind libxslt1.1 libssl-dev␣
↪libcrypto++-dev

Setting up libboost1.54-dev (1.54.0-4ubuntu3.1) ...
Setting up libboost-dev (1.54.0.1ubuntu1) ...
```

Install the Mongo Driver from source:

```
    mkdir -p /opt/mongodb
cd /opt/mongodb
    wget https://github.com/mongodb/mongo-cxx-driver/archive/legacy-1.0.2.tar.gz
    tar xfvz legacy-1.0.2.tar.gz
    cd mongo-cxx-driver-legacy-1.0.2
    scons                                          # The build/linux2/normal/
↪libmongoclient.a library is generated as outcome
    sudo scons install --prefix=/usr/local         # This puts .h files in /usr/local/
↪include/mongo and libmongoclient.a in /usr/local/lib
```

Install rapidjson from sources:

```
    mkdir -p /opt/rapidjson
cd /opt/rapidjson
    wget https://github.com/miloyip/rapidjson/archive/v1.0.2.tar.gz
    tar xfvz v1.0.2.tar.gz
    sudo mv rapidjson-1.0.2/include/rapidjson/ /usr/local/include
```

Install Google Test/Mock from sources (:

```
    mkdir -p /opt/googlemock
cd /opt/googlemock
    wget http://googlemock.googlecode.com/files/gmock-1.5.0.tar.bz2
    tar xfvj gmock-1.5.0.tar.bz2
    cd gmock-1.5.0
    ./configure
```

```
    make
    sudo make install  # installation puts .h files in /usr/local/include and library␣
↪in /usr/local/lib
    sudo ldconfig      # just in case... it doesn't hurt :)
```

Build Orion itself

```
    mkdir -p /opt/fiware/orion/
cd /opt/fiware/orion/
    git clone https://github.com/telefonicaid/fiware-orion git
cd git

    # Build errors on linking! libboost regex it seems
    [100%] Building CXX object src/app/contextBroker/CMakeFiles/contextBroker.dir/
↪contextBroker.cpp.o
    Linking CXX executable contextBroker
    /usr/local/lib/libmongoclient.a(dbclient.o): In function `boost::basic_regex<char,
↪ boost::regex_traits<char, boost::cpp_regex_traits<char> > >::assign(char const*,␣
↪char const*, unsigned int)':
    /usr/include/boost/regex/v4/basic_regex.hpp:382: undefined reference to␣
↪`boost::basic_regex<char, boost::regex_traits<char, boost::cpp_regex_traits<char> >␣
↪>::do_assign(char const*, char const*, unsigned int)'
    /usr/local/lib/libmongoclient.a(dbclient.o): In function `boost::re_detail::perl_
↪matcher<__gnu_cxx::__normal_iterator<char const*, std::string>, std::allocator
↪<boost::sub_match<__gnu_cxx::__normal_iterator<char const*, std::string> > >,␣
↪boost::regex_traits<char, boost::cpp_regex_traits<char> > >::unwind_extra_
↪block(bool)':
    /usr/include/boost/regex/v4/perl_matcher_non_recursive.hpp:1117: undefined␣
↪reference to `boost::re_detail::put_mem_block(void*)'
    /usr/local/lib/libmongoclient.a(dbclient.o): In function `boost::re_detail::cpp_
↪regex_traits_implementation<char>::error_string(boost::regex_constants::error_type)␣
↪const':
    /usr/include/boost/regex/v4/cpp_regex_traits.hpp:447: undefined reference to␣
↪`boost::re_detail::get_default_error_string(boost::regex_constants::error_type)'
    /usr/local/lib/libmongoclient.a(dbclient.o): In function `void boost::re_
↪detail::raise_error<boost::regex_traits_wrapper<boost::regex_traits<char,␣
↪boost::cpp_regex_traits<char> > > >(boost::regex_traits_wrapper<boost::regex_traits
↪<char, boost::cpp_regex_traits<char> > > const&, boost::regex_constants::error_type)
↪':
    /usr/include/boost/regex/pattern_except.hpp:75: undefined reference to `boost::re_
↪detail::raise_runtime_error(std::runtime_error const&)'
    /usr/local/lib/libmongoclient.a(dbclient.o): In function `boost::re_detail::cpp_
↪regex_traits_implementation<char>::error_string(boost::regex_constants::error_type)␣
↪con

    # appears known problem: https://github.com/telefonicaid/fiware-orion/issues/1162
# DISTRO 14.04.3_LTS var not in add in src/app/contextBroker/CMakeLists.txt
# add
ELSEIF(${DISTRO} STREQUAL "Ubuntu_14.04.3_LTS")
   MESSAGE("contextBroker: Ubuntu ===TEST===== DISTRO: '${DISTRO}'")
   TARGET_LINK_LIBRARIES(contextBroker ${STATIC_LIBS} -lmicrohttpd -lmongoclient -
↪lboost_thread -lboost_filesystem -lboost_system -lboost_regex -lpthread -lssl -
↪lcryp\
to -lgnutls -lgcrypt)
```

(Optional but highly recommended) run unit test. Firstly, you have to install MongoDB (as the unit tests rely on mongod running in localhost).

```
apt-get install mongodb-server
apt-get install pcre            # otherwise, mongodb crashes in CentOS 6.3
service mongodb start
service mongodb status   # to check that mongodb is actually running
make unit_test

[----------] Global test environment tear-down
[==========] 1101 tests from 168 test cases ran. (4985 ms total)
[  PASSED  ] 1096 tests.
[  FAILED  ] 5 tests, listed below:
[  FAILED  ] mongoQueryContextRequest_filters.outsideRange_n
[  FAILED  ] mongoQueryContextRequest_filters.withoutEntityType
[  FAILED  ] mongoQueryContextGeoRequest.queryGeoCircleOut
[  FAILED  ] mongoQueryContextGeoRequest.queryGeoPolygonOut1
[  FAILED  ] mongoQueryContextGeoRequest.queryGeoPolygonOut2

5 FAILED TESTS
YOU HAVE 23 DISABLED TESTS
```

Also need to fix build error in make unit_tests

```
ELSEIF(${DISTRO} STREQUAL "Ubuntu_14.04.3_LTS")
  MESSAGE("contextBroker: Ubuntu ===TEST===== DISTRO: '${DISTRO}'")
  TARGET_LINK_LIBRARIES(unitTest ${STATIC_LIBS} -lmicrohttpd -lmongoclient -lboost_
↪thread -lboost_filesystem -lboost_system -lboost_regex -lpthread -lssl -lcrypto -lg\
nutls -lgcrypt)
```

Install the binary. You can use INSTALL_DIR to set the installation prefix path (default is `/usr`), thus the broker is installed in `$INSTALL_DIR/bin` directory.

```
sudo make install INSTALL_DIR=/usr

#test install
contextBroker --version
0.24.0-next (git version: a938e68887fbc7070b544b75873af935d8c596ae)
Copyright 2013-2015 Telefonica Investigacion y Desarrollo, S.A.U
Orion Context Broker is free software: you can redistribute it and/or
modify it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

Orion Context Broker is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero
General Public License for more details.

Telefonica I+D
```

Running OCB

COnfig in `/etc/default/contextBroker`. As system service:

```
service contextBroker start
# calls /usr/bin/contextBroker
ERROR: start-stop-daemon: user 'orion' not found
```

```
# Found info in DockerFile: https://github.com/Bitergia/fiware-chanchan-docker/blob/
↪master/images/fiware-orion/0.22.0/Dockerfile
# Add user without shell
useradd -s /bin/false -c "Disabled Orion User" orion

mkdir -p /var/log/contextBroker
mkdir -p /var/run/contextBroker
chown orion:orion /var/log/contextBroker
chown orion:orion /var/run/contextBroker
service contextBroker status
# contextBroker is running
```

Test with fiware-figway Python client:

```
sunda:ContextBroker just$ python  GetEntities.py ALL
* Asking to http://sensors.geonovum.nl:1026/ngsi10/queryContext
* Headers: {'Fiware-Service': 'fiwareiot', 'content-type': 'application/json', 'accept
↪': 'application/json', 'X-Auth-Token': 'NULL'}
* Sending PAYLOAD:
{
    "entities": [
        {
            "type": "",
            "id": ".*",
            "isPattern": "true"
        }
    ],
    "attributes": []
}

...

* Status Code: 200
***** Number of Entity Types: 0

***** List of Entity Types

**** Number of Entity IDs: 0

**** List of Entity IDs

Do you want me to print all Entities? (yes/no)yes
<queryContextResponse>
  <errorCode>
    <code>404</code>
    <reasonPhrase>No context element found</reasonPhrase>
  </errorCode>
</queryContextResponse>
```

12.6.2 IoT Agent (CPP version)

From GitHub: https://github.com/telefonicaid/fiware-IoTAgent-Cplusplus, using Docker file https://github.com/telefonicaid/fiware-IoTAgent-Cplusplus/blob/develop/docker/Dockerfile for inspiration.

Build fiware-IoTAgent-Cplusplus

From source.

Steps:

```
    apt-get install -y tar gzip unzip file cpp gcc automake autoconf libtool git␣
↪scons cmake
    apt-get install -y libssl-dev libbz2-dev zlib1g-dev doxygen

    mkdir -p /opt/fiware/iotagent
    git clone https://github.com/telefonicaid/fiware-IoTAgent-Cplusplus iotacpp
    cd /opt/fiware/iotagent/iotacpp

    # Disable and fix  CMakeLists.txt
    cp CMakeLists.txt CMakeLists.txt.backup
    sed -i CMakeLists.txt -e 's|^add_test|#add_test|g' -e 's|^add_subdirectory(tests)|
↪#add_subdirectory(tests)|g' -e 's|^enable_testing|#enable_testing|g' -e
↪'s|git@github.com:mongodb/mongo-cxx-driver.git|https://github.com/mongodb/mongo-cxx-
↪driver|g'

    # Get version string
    $ source tools/get_version_string.sh
    $ get_rpm_version_string | cut -d ' ' -f 1
    1.3.0

    # Build in Release subdir
    mkdir -p ${CMAKE_CURRENT_SOURCE_DIR}/build/Release
    cd ${CMAKE_CURRENT_SOURCE_DIR}/build/Release
    cmake -DGIT_VERSION=1.3.0 -DGIT_COMMIT=1.3.0 -DMQTT=ON -DCMAKE_BUILD_TYPE=Release␣
↪ ../../

    # very long build...lots of output...
$ make install
```

Create install scripts from RPM spec files. Need these files https://github.com/telefonicaid/fiware-IoTAgent-Cplusplus/tree/develop/rpm/SOURCES See https://github.com/Geonovum/sospilot/tree/master/src/fiware/iotagent

Running

Config JSON.

```
{
    "ngsi_url": {
        "cbroker": "http://127.0.0.1:1026",
        "updateContext": "/NGSI10/updateContext",
        "registerContext": "/NGSI9/registerContext",
        "queryContext": "/NGSI10/queryContext"
    },
    "timeout": 10,
    "http_proxy": "PUBLIC_PROXY_PORT",
    "public_ip": "8081",
    "dir_log": "/var/log/iot/",
    "timezones": "/etc/iot/date_time_zonespec.csv",
    "storage": {
        "host": "localhost",
```

```
        "type": "mongodb",
        "port": "27017",
        "dbname": "iot"
    },
   "resources": [
        {
            "resource": "/iot/d",
            "options": {
                "FileName": "UL20Service"
            }
        },
        {
            "resource": "/iot/mqtt",
            "options": {
                "ConfigFile" : "/etc/iot/MqttService.xml",
                "FileName": "MqttService"
            }
         }
   ]
}
```

Core dump...gave up.

CHAPTER 13

Some First Ideas

Hieronder wat eerste ideeën. Kept for future reference.

13.1 Project naam

SensorPlus of bestaat er al iets? Denk `SOSPilot`

13.2 Support Technologie

1. website sensors.geonovum.nl: simpele Bootstrap HTML met info + links naar bijv. 52N Client en andere demo's, documentatie etc
2. code en doc: GitHub: account Geonovum?, in ieder geval waar we beiden direct op kunnen committen ipv via PRs? *DONE*
3. documentatie: Sphynx+ReadTheDocs, gebruik bij bijv Stetl en OGG: http://stetl.org, werkt via GH Post Commit, dus bij iedere commit is docu weer synced en online en er is export naar PDF *DONE*
4. ETL: Python, GDAL, SQL, evt Stetl?
5. website: sensors.geonovum.nl, simpele Bootstrap site, auto deploy via: https://gist.github.com/oodavid/1809044

13.3 Inrichten Linux Server

- algemeen: stappen doc http://docs.kademo.nl/project/geolinuxserver.html
- apart admin account die minder rechten heeft als root maar wel Tomcat/ETL etc kan runnen
- backup: inrichten (met CloudVPS backupserver en script)

13.4 ETL Opzet

Denk 3 ETL stappen met 3 stores:

1. Harvesten van brondata uit http://geluid.rivm.nl/sos/ op in DB als XML Blobs met filenaam en start/eind tijd kolom
2. lokale brondata naar intermediate "core" DB: in principe 2 tabellen nodig: Stations en Metingen, 1:1 overzetten uit XML
3. "Core DB" naar 52N SOS DB, evt later naar IPR/INSPIRE XML

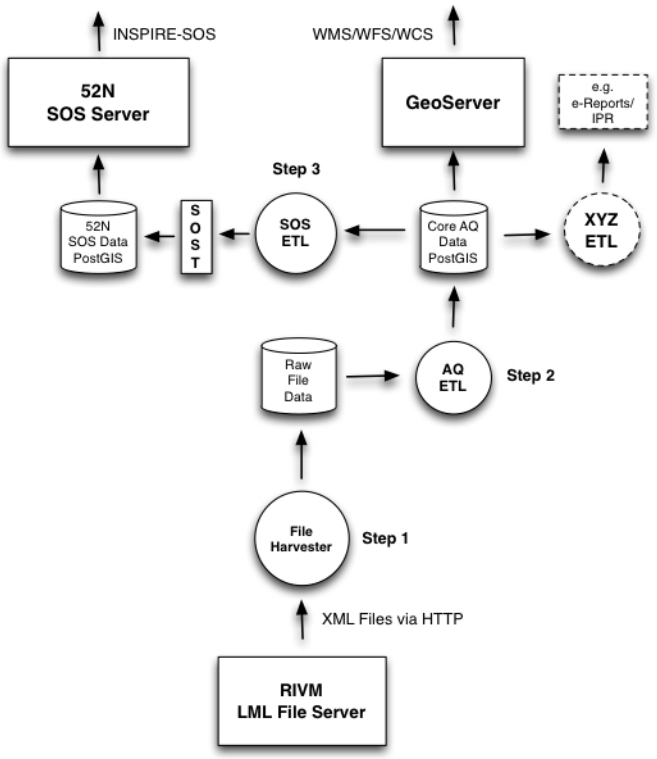


Fig. 13.1: *Figure 1 - Overall Architecture*

De pijlen geven de dataflow weer. Processen zijn cirkels. De flow is als volgt:

1. De File Harvester haalt steeds XML files met AQ/LML metingen op van de RIVM server
2. De File Harvester stopt deze files als XML blobs 1-1 in de database, met filenaam+datum kolommen
3. Het AQ ETL proces leest steeds de file blobs uit de Raw File Data DB en zet deze om naar een Core AQ DB
4. De Core AQ DB bevat de metingen + stations in reguliere tabellen 1-1 met de oorspronkelijke data, met ook Time kolommen
5. De Core AQ DB kan gebruikt worden om OWS (WMS/WFS) services via GeoServer te bieden
6. Het SOS ETL proces zet de core AQ data om naar de 52North SOS DB schema of evt via REST publicatie (TODO)
7. De drie processen (File Harvester, AQ ETL en SOS ETL) bewaren steeds een "last ETL time" timestamp waardoor ze op elk moment "weten waar ze zijn" en kunnen hervatten

Deze opzet is vergelijkbaar met die van BAG in PDOK, daar vinden de volgende stappen plaats:

1. BAG XML downloaden (via Atom).
2. BAG XML inlezen 1:1 model, in Core BAG DB.
3. Vanaf Core BAG DB transformatie naar andere formats/DBs: GeoCoder, INSPIRE AD, BagViewer etc.

De 3 ETL-processen (bijv via cron) houden steeds hun laatste sync-time bij in DB.

Voordelen van deze opzet:

- backups van de brondata mogelijk
- bij wijzigingen/fouten altijd de "tijd terugzetten" en opnieuw draaien
- simpeler ETL scripts dan "alles-in-1", bijv van "Core AQ DB" naar "52N SOS DB" kan evt in SQL
- migratie bij wijzigingen 52N SOS DB schema simpeler
- voorbereid op IPR/INSPIRE ETL (bron is dan Core OM DB)
- OWS server (WMS/WFS evt WCS) aansluiten op Core OM DB (via VIEW evt, zie onder)

13.5 OWS Inrichting

GeoServer draait reeds op http://sensors.geonovum.nl/gs

- GeoServer (?, handig bijv voor WMS-T en brede WFS en INSPIRE support)
- Met een VIEW op de "Core OM DB" kan een DB voor WMS(-Time) / WFS evt WCS ingeregeld (join tabel op stations/metingen).

13.6 OWS Client

- WFS Filter Client met Download: voorbeeld: http://lib.heron-mc.org/heron/latest/examples/multisearchcenternl/ ("Build your own searches" optie)
- TimeSlider (WMS-Time) Heron voorbeeld: http://lib.heron-mc.org/heron/latest/examples/timeslider
- EEA voorbeeld: http://www.eea.europa.eu/themes/air/interactive/pm10-interpolated-maps

13.7 SOS Inrichting

52N SOS draait reeds op http://sensors.geonovum.nl/sos

- 52N SOS server (Simon) inspire versie, zie action point
- DB in PG schema, niet in "public" (ivm backup/restore andere systemen)

13.8 SOS Client Inrichting

- 52N (Simon) zie action point

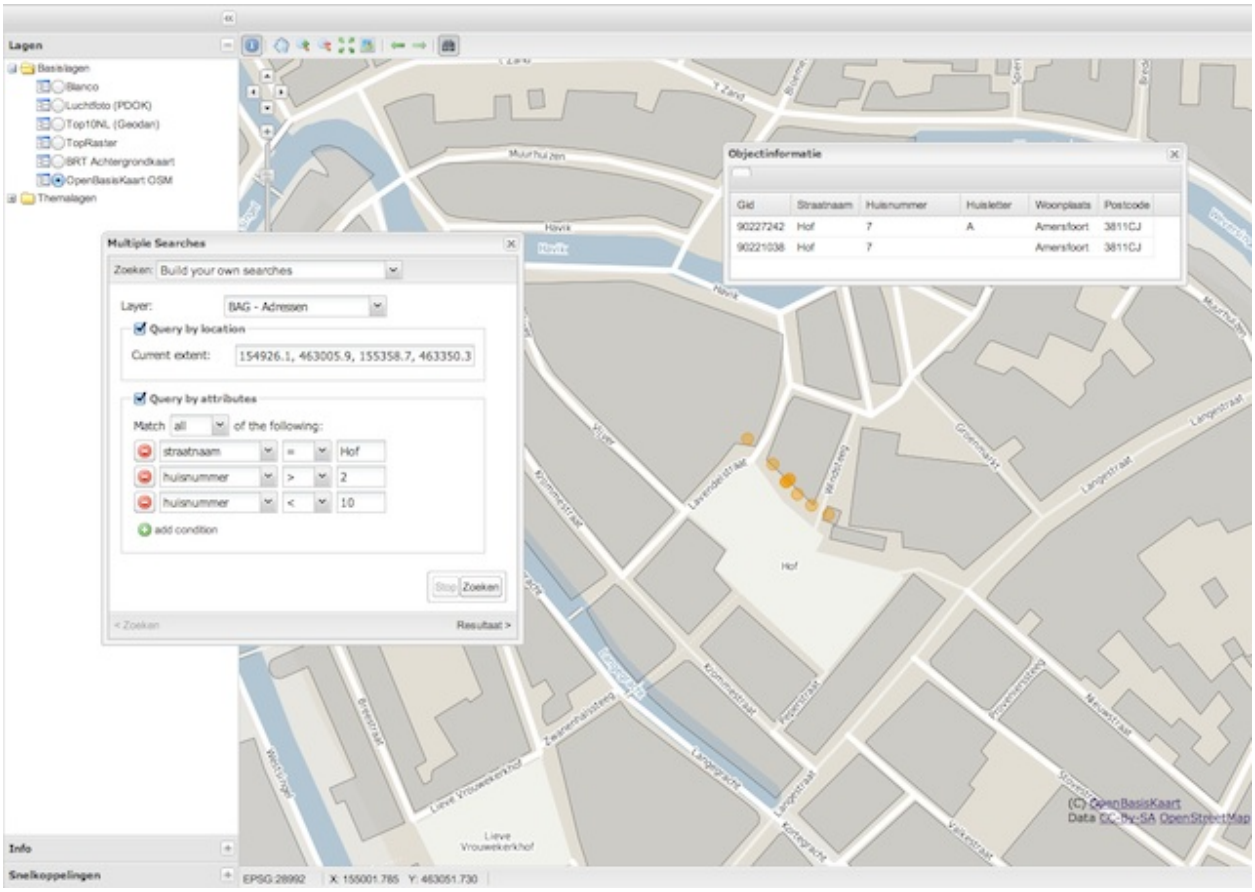


Fig. 13.2: *Figure - Heron WFS Query Builder*

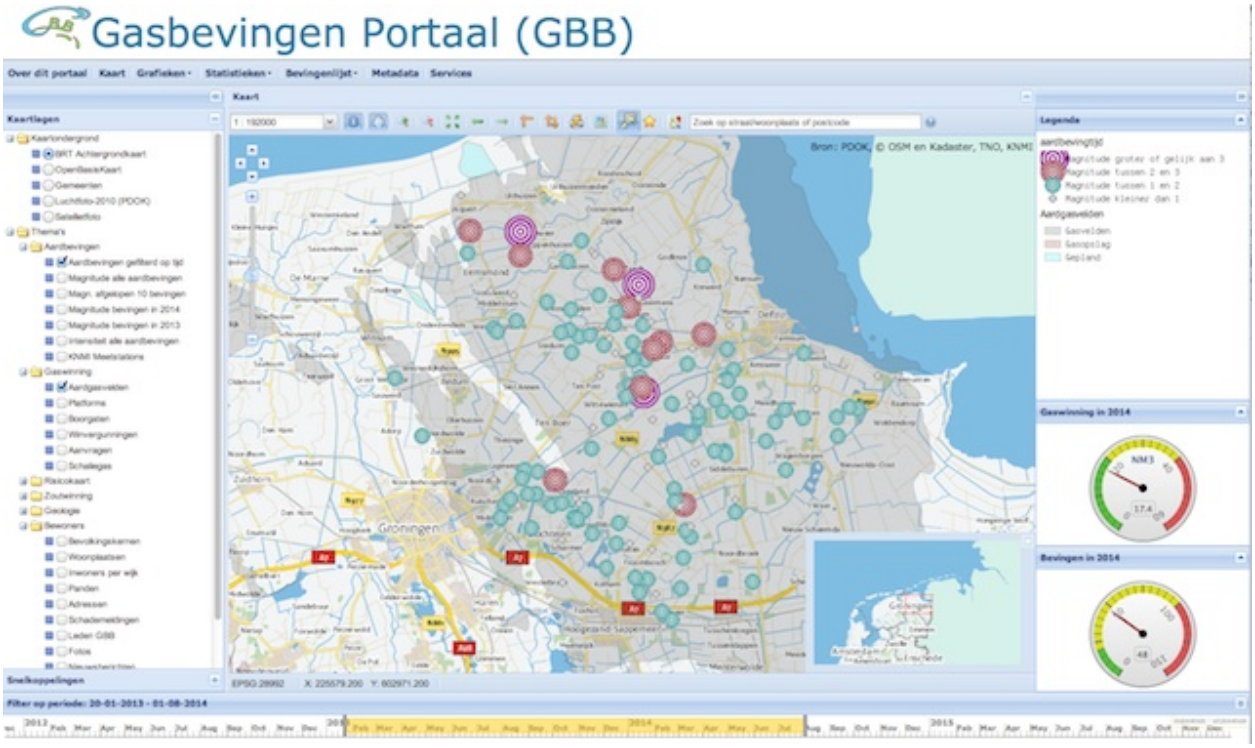


Fig. 13.3: *Figure - Heron Timeslider in Gasbevingen Portaal*

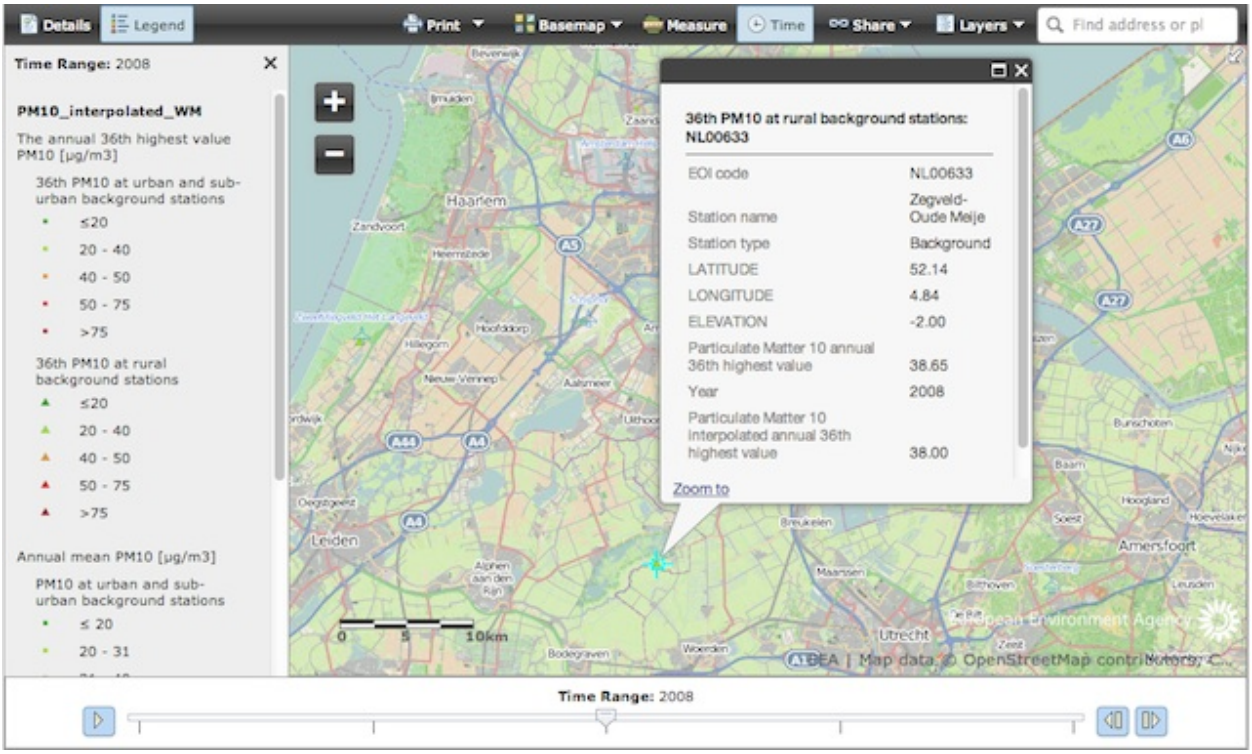


Fig. 13.4: *Figure - eea.europa.eu pm10-interpolated-map*

CHAPTER 14

Dissemination

This chapter collects various presentations and documents related to dissemination events like workshops and presentations. All "raw" documents can be found in the project GitHub repository at https://github.com/Geonovum/sospilot/tree/master/docs/_static/dissemination.

14.1 Presentation 27 okt 2014

Presentation at RIVM on the SOSPilot project in particular LML to SOS and EU Air Quality Reporting.

- SOSPilot Presentation Slides

14.2 Workshop 17 dec 2015

In the context of the *Smart Emission project <http://www.ru.nl/gpm/onderzoek/research-projects/smart-emission>* a workshop was held at Geonovum on Dec 17, 2015. The main subjects were:

- presenting OGC standards in general (WMS, WFS, CSW) and in particular Sensor-related (SWE, SOS)
- hands-on demo's for several clients and visualisations (QGIS, ArcGIS and Heron)

14.2.1 Slides

The agenda and presentation slides, all in PDF, can be found here:

- Agenda
- OGC Standards+INSPIRE - Thijs Brentjens
- OGC SWE/SOS - Just van den Broecke
- Visualisation in ArcGIS of LML data - Graduation Presentation - Freek Thuis
- QGIS - Matthijs Kastelijns

- Heron Viewer - Just van den Broecke

14.2.2 Links

More detailed tutorials on OGC standards at the OGC School in particular the entire PDF document (18MB!).

Some interactive visualisation examples "via Google" were also shown like made with D3JS:

- Visualization of Beijing Air Pollution: http://vis.pku.edu.cn/course/Visualization_2012F/assignment2/Chengye
- More can be found here http://vis.pku.edu.cn/wiki/public_course/visclass_f12/assignment/a02/websites/start (click "Alive Version" for each)

CHAPTER 15

Contact

The website sensors.geonovum.nl is the main entry point for this project.

All development is done via GitHub: see https://github.com/geonovum/sospilot.

Developers are Thijs Brentjens and Just van den Broecke

Contact Michel Grothe at Geonovum via email at michel AT geonovum.nl

CHAPTER 16

Links

Below links relevant to the project.

16.1 Education

- Geonovum Course Sensor_Web_Enablement (SWE) - http://geostandards.geonovum.nl/index.php?title=5_Sensor_Web_Enablement
- OGC SWE Architecture: http://docs.opengeospatial.org/wp/07-165r1/
- Wikipedia Sensor Web http://en.wikipedia.org/wiki/Sensor_web

16.2 Data

16.3 Linux Server

16.4 Demos and Servers

16.5 Europe

16.6 Tools and Libs

16.7 Hardware

CHAPTER 17

Indices and tables

- genindex
- modindex
- search

Bibliography

[LML] Landelijk Meetnet Luchtkwaliteit http://www.lml.rivm.nl/

[LMLXML] Ruwe LML XML data RIVM: http://geluid.rivm.nl/sos

[IPRXML] Download XML in IPR format: http://www.regione.piemonte.it/ambiente/aria/rilev/ariaday-test/xmlexport/read?startDate=&endDate=&action=selection

[UbuntuGeo] Algemene handleiding Ubuntu FOSS Geo inrichting: http://docs.kademo.nl/project/geolinuxserver.html

[SOS52N] SOS 52N Project Server: http://sensors.geonovum.nl/sos

[OWSGS] WMS/WFS Project Server: http://sensors.geonovum.nl/gs

[EEA] European Environment Agency (EEA) http://www.eea.europa.eu/themes/air

[Eionet] Eionet AQ Portal: http://www.eionet.europa.eu/aqportal/

[ECID] Directives 2004/107/EC and 2008/50/EC Implementing Decision - http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2011:335:0086:0106:EN:PDF

[AirQualityReportingXSD] AirQualityReporting.xsd - http://dd.eionet.europa.eu/schema/id2011850eu-1.0/AirQualityReporting.xsd/view

[GDALOGR] GDAL/OGR, http://gdal.org

[Stetl] Stetl, Open Source ETL in Python, http://www.stetl.org

[PythonBeautifulSoup] Python Beautiful Soup : om HTML docs (bijv index.html Apache) uit te parsen: http://www.crummy.com/software/BeautifulSoup

[lxml] lxml, http://lxml.de

[Deegree] Deegree, http://www.deegree.org

[INSPIRE] INSPIRE, http://inspire.ec.europa.eu/

[INSPIREFOSS] INSPIRE FOSS project, http://inspire-foss.org

[PostGIS] PostGIS/PostgreSQL, http://postgis.refractions.net

[DAVISVP] Davis Vantage Pro2 Weather Station, http://www.davisnet.com/weather/products/vantage-pro-professional-weather-stations.asp