

Implementing a Sensor Observation Service for Wi-Fi tracking data

Matthijs Bon
Xander den Duijn
Balazs Dukai



Implementing a Sensor Observation Service for Wi-Fi tracking data

by

Matthijs Bon
Xander den Duijn
Balazs Dukai

**GEO1007 course during the MSc programme Geomatics for the built
environment
at the Delft University of Technology,**

Project duration: April 16, 2016 – June 9, 2016

to be presented on Friday May 20, 2016.

Preface

Matthijs Bon
Xander den Duijn
Balazs Dukai
Delft, June 2016

Contents

1	Introduction	1
1.1	Problem description	1
1.2	Objectives.	1
1.3	Methods	2
2	Functionality	5
2.1	52North application.	5
2.1.1	Database model	5
3	Technical Description	7
3.1	Preprocessing (of the wifilog data)	7
3.2	Filling the tables	7
3.2.1	'Simple' tables	7
3.2.2	FeatureOfInterest	7
3.2.3	Series	7
3.2.4	Observation	7
3.2.5	ComplexValue	7
4	Status of the application	9
5	Conclusions and recommendations	11
A	Appendix: Sample requests	13

Introduction

This report describes the process and results of the assignment for the GEO1007 course in the MSc Geomatics programme. For this assignment, the authors chose one of seven topics. The chosen topic focusses on implementing a Sensor Observation Service to publish the tracking data derived from the geomatics synthesis project on a server and test it in a SOS client, such as the 52 North application.

A Sensor Observation Service (SOS) is a service standardized by the Open Geospatial Consortium (OGC). The standard defines a web service interface for the discovery and retrieval of real time or archived data produced by all kinds of sensors like mobile or stationary as well as in-situ or remote sensors (OGC, 2016). A sensor can measure multiple things, e.g. a meteorological sensor can observe properties such as temperature, wind speed, humidity. The SOS standard focusses mainly on the observations of these sensors.

In this project, the focus lies on the Wi-Fi network of the TU Delft campus. The wireless access points (AP) used in the Wi-Fi network can be seen as sensors, and the devices registered by the APs can be seen as the measurement. Each device can be identified by its unique mac address, but also other properties can be measured, i.e. the received signal strength and the signal to noise ratio (SNR). These are all observations that can be retrieved using a SOS. In the next sections the research question, objectives, methods and tools will be discussed.

1.1. Problem description

Implementing a SOS from Wi-Fi tracking data is a topic that has little to no scientific literature. So far sensor observations services are most commonly used with sensors that observe natural or meteorological properties, such as temperature. Using it for publishing Wi-Fi tracking data is uncommon and will require much trial and error.

During this project, the research will be guided by the following question:

How can the 52North web application be used to publish and visualize the WiFi tracking data from the TU Delft eduroam network?

This research question can be divided into the following subjects which need to be addressed:

- Research the 52North database model for a SOS
- Setting up the SOS server
- Testing the SOS client

These questions can be answered once the goals and objectives have been established.

1.2. Objectives

To answer the research question and sub questions, the purpose of the research has to be clearly defined. The purpose of the research is captured into two main objectives:

MUST	SHOULD
Automatically transform the raw WiFi-logs from the PostgreSQL database to an SOS-compliant data model.	Functionality to download raw WiFi-logs or trajectories.
Time series tracking data (subset WiFi-logs with time range) in the client	
COULD	WONT
	Push notification to the user when new data is available
	Push new data to the user. When subscribing, the user can decide to either receive the raw WiFi-logs or the trajectories.

Table 1.1: MoSCoW Rules

- To provide a method for publishing WiFi-based tracking data through an SOS service and visualize the data in a client. The user can view and subset the data in the client and eventually download it. This allows a quick, preliminary data filtering which speeds up the data analysis work flow.
- To set up a SES service which pushes newly added tracking data to the client/user.

To dive deeper into the steps that need to be taken to answer the research question, the objectives can be divided into sub-goals:

- Automatically transform the raw WiFi-logs from the PostgreSQL database to an SOS-compliant data model.
- Functionality to download raw WiFi-logs or trajectories.
- Time series tracking data (subset WiFi-logs with time range) in the client

Finally, to structure the objectives and goals and to define the scope of the project, the objectives are grouped using the MoSCoW rules (see Table 1.1). To achieve these goals, implementation of the SOS Core Profile is necessary, comprising the mandatory operations: GetCapabilities, DescribeSensor, GetObservation.

1.3. Methods

Publishing the Wi-Fi tracking data using the SOS can be done using different methods. For this project multiple tools are used, these are listed below.

- PostgreSQL
- Python
- GeoServer
- Apache Tomcat
- 52North Application

The 52North application was used as a guide for the implementation of the SOS on the server. This application is very well documented on their website. An work flow combining Python and PostgreSQL is used to automate as much of the SQL that is needed for the implementation i.e. limiting the manual work.

Occupancy vs Movement

This project aligns with the Synthesis project of the MSc Geomatics programme. During that project, we focus on movement in and between buildings. The first idea was to publish this movement data using the SOS, but this soon became rather difficult. Because the SOS uses time series to organize the data, there are limitations on the use of the data. When consider the access points as sensors in the SOS, one observation can include a mac address, SNR and RSSI. The time series organize these measurements in a way that one access point can have multiple observations, but one observation (mac address in this case) can not be tracked. Another way of looking at this would be to consider each mac address as sensor and each observation is the access point that it connects to. The first method will consider the data as occupancy information, whereas the second method will consider it movement data, in which tracking one person over time is possible. Because the time in this project is limited, the decision was made to see the access points as sensors, i.e. considering not movement but occupancy.

2

Functionality

The following chapter describes the OGC SOS standard implementation by the 52°North organization. The OGC SOS 2.0 standard was adopted in 2012 which serves the basis of the 52°North SOS 4.x implementation. Firstly the functionalities of the implementation are detailed, then the underlying data model is described.

2.1. 52North application

The 52°North SOS 4.x supports the requirements of the SOS 2.0 specification, implementing all of its extensions and their operations:

1. **Core:** GetCapabilities, GetObservation, DescribeSensor
2. **Enhanced:** GetFeatureOfInterest, GetObservationById
3. **Transactional:** InsertSensor, UpdateSensorDescription, DeleteSensor, InsertObservation
4. **Result Handling:** InsertResultTemplate, InsertResult, GetResultTemplate, GetResult

Additionally, 52°North SOS 4.x offers the following main features:

- SOS API
- Persistence framework to easily change the underlying database management system and database model (*Hibernate* and *Hibernatespatial*)
- Administration GUI
- Installer GUI
- Bundle including SWC REST-API and JavaScript SOS Client
- RESTful binding extension

2.1.1. Database model

The 52°North SOS 4.x database model is divided into two major profile, **Core** and **Transactional** profile. The Core database model implements the OGC SOS 2.0 *Core* and *Extended* profiles, while the Transactional database model extends the Core model to implement the OGC SOS 2.0 *Transactional* and *Result Handling* profiles. In the Core database model, the following tables (Table 2.1) are relevant for disseminating Wi-Fi tracking data. In the Transactional database model, the following tables (Table 2.2) are relevant for disseminating Wi-Fi tracking data.

Table name	Description
<i>codespace</i>	contains the {codespace} and {codespacename} for the identifiers of Procedure, ObservableProperty, FeatureOfInterest and Offering
<i>featureOfInterest</i>	the geometries of the observations
<i>featureOfInterestType</i>	the type of the geometries of the observations
<i>observableProperty</i>	parameters that are observed
<i>series</i>	describes a series, a combination of featureOfInterest, observableProperty and procedure
<i>observation</i>	contains the observations
<i>observationHasOffering</i>	relates the offerings to the observations
<i>offering</i>	needed for structuring the data in a SOS server
<i>procedure</i>	the processes thorough which the observed values were generated
<i>procedureDescriptionFormat</i>	the format in which procedures shall be described
<i>unit</i>	units of a measurement
<i>textvalue</i>	contains the measured values, which are of type OM_TextObservation
<i>observationConstellation</i>	look-up table to check if the observation type of the InsertObservation, request is valid for the constellation procedure, observableProperty,,and offering
<i>observationType</i>	stores the observation types

Table 2.1: Tables in the Core database model

Table name	Description
<i>featureRelation</i>	hierarchy of features
<i>offeringAllowedFeatureType</i>	look-up table for the feature types which are valid for the offering
<i>offeringAllowedObservationType</i>	look-up table for the observation types which are valid for the offering
<i>resultTemplate</i>	holds the result template information which are necessary for the result handling operations
<i>validProcedureTime</i>	stores the procedure description

Table 2.2: Tables in the Transactional database model

3

Technical Description

3.1. Preprocessing (of the wifilog data)

3.2. Filling the tables

3.2.1. 'Simple' tables

codespace

`FeatureOfInterest::codeSpace` is the codespace attribute for the identifier (*gml:identifier*) of the access points. As the GML 3.2. schema defines the *gml:identifier* is a “special identifier is assigned to an object by the maintaining authority with the intention that it is used in references to the object.” Furthermore, the attribute *codeSpace* is of type *anyURI*. In the present case the Technical University Delft is the maintaining authority of the campus WLAN and with it the access points. Because there is no official repository that contains the identifiers of the access points, we defined the codespace as *tudelft-wlan*.

name and codespacename

Codespacename refers to the codespace for the *name* of the *featureOfInterest*. *FeatureOfInterest:name* refers to *gml:name* in the GML 3.2 schema. In the case of the TU Delft WLAN the access point names equal to the access point identifiers, thus they have the same codespace as well.

hibernatediscriminator

According to the SensorObservationService documentation provided by 52North's wiki page, four tables require the attribute 'hibernatediscriminator'. The wiki page describes this attribute as 'only needed for internal purposes', but this is rather unclear. Because there is no clear description on the value of this attribute, the value 'F' is chosen, as indicator for a 'False' value.

3.2.2. FeatureOfInterest

3.2.3. Series

3.2.4. Observation

3.2.5. ComplexValue

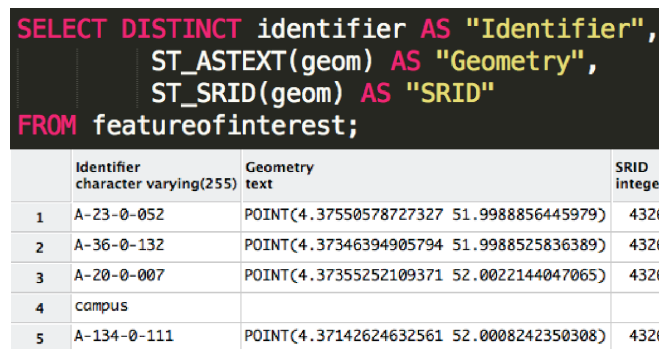
4

Status of the application

The application that was created during the course of this project is almost finished. The steps that needed to be taken to get from the raw Wi-Fi data to the SOS on the geoserver have all been taken, but some debugging is still to be done to have a final working application.

Setting up the geoserver was the easy part. The webapp that 52°North provided could be imported directly into the geoserver running on the local machines. The configuration of the 52°North SOS was slightly harder. The original Wi-Fi tracking data is on a remote server on the TU Delft campus, but the encoding for that server is in 'LATIN1', which is unsupported by the 52°North SOS. It required a new database with UTF-8 encoding to get the SOS tables in the database. Filling the tables with the tracking data took most of the time, but using the 52°North application as a guide ensured that the correct attributes were used in the tables.

Finally, the 52°North webapp could be used to do sample queries on the database to check if the database model was correct. This worked and showed that the database model was correct. Figure 4.1 shows such a sample query and the result from the database.



The image shows a screenshot of a database client interface. At the top, a SQL query is entered in a text area: `SELECT DISTINCT identifier AS "Identifier", ST_ASTEXT(geom) AS "Geometry", ST_SRID(geom) AS "SRID" FROM featureofinterest;`. Below the query, the results are displayed in a table with four columns: an index, 'Identifier' (character varying(255)), 'Geometry' (text), and 'SRID' (integer). The table contains five rows of data.

	Identifier character varying(255)	Geometry text	SRID integer
1	A-23-0-052	POINT(4.37550578727327 51.9988856445979)	4326
2	A-36-0-132	POINT(4.37346394905794 51.9988525836389)	4326
3	A-20-0-007	POINT(4.37355252109371 52.0022144047065)	4326
4	campus		
5	A-134-0-111	POINT(4.37142624632561 52.0008242350308)	4326

Figure 4.1: Test queries on the database

Additionally, sample requests can also be sent. These sample requests can be generated by the 52°North webapp test client. Such requests are *GetCapabilities* and *GetFeatureOfInterest*. The screenshots of these requests are depicted in appendix A.

From the bindings that are available to send the request (JSON, KVP, SOAP and POX), only the KVP (Key Value Pairs) seemed to work. Unfortunately the other bindings would result in errors. Without deeper understanding about SOS and these bindings, we were not able to debug the errors and create valid requests.

Furthermore, we were able to retrieve the observations with KVP binding and *GetObservation* procedure, using the extension *MergeObservationsIntoDataArray*. This procedure returns the *description*, *type*, *phenomenonStartTime*, *phenomenonEndTime*, *resultTime*, *procedure identifier and name*, *observedProperty identifier and name*, *result* and *the type, name, identifier, coordinates of the featureOfInterest*. See the request below and a screenshot of the result in the appendix.

`http://localhost:8080/52n-sos-webapp/service?service=SOS&version=2.0.0&request=GetObservation&MergeObservationsIntoDataArray=true&procedure=wifi_access_point`

However, the same request without using the extension returns an empty result, no error message. See the url below.

`http://localhost:8080/52n-sos-webapp/service?service=SOS&version=2.0.0&request=GetObservation&procedure=wifi_access_point`

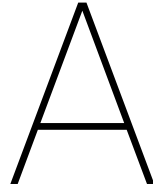
5

Conclusions and recommendations

Conclusions

From current status of the application, it can be concluded that to some extent the 52°North web app is suitable for publishing Wi-Fi tracking data. However, not every part of the implementation currently works as it should. Creating a database compliant with the SOS database model is very feasible, if one takes time to understand the database model and has understanding on how to populate the tables in the database. As discussed in chapter 2, mapping the correct attribute values to the correct tables is very important and should be done carefully.

Recommendations For this project research into the implementation of Sensor Observation Services for Wi-Fi tracking data was conducted, but other services were disregarded. There was discussion about whether or not WFS would be a better service to publish Wi-Fi tracking data, but no in depth research was conducted. For better assessment, WFS should be investigated. Furthermore, this project focusses SOS only, but with SOS as starting point, a Sensor Event Service (SES) could serve as a standard to push notifications to users when there is new data available. In further research, this could be a useful addition when creating an application to publish and monitor Wi-Fi tracking data. Additionally,



Appendix: Sample requests

```
<?xml version='1.0' encoding='UTF-8'>
<sos:GetFeatureOfInterestResponse xmlns:sos="http://www.opengis.net/sos/2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:sams="http://www.opengis.net/samplingSpatial/2.0" xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:sf="http://www.opengis.net/sampling/2.0"
  xmlns:xlink="http://www.w3.org/1999/xlink" xsi:schemaLocation="http://www.opengis.net/sos/2.0 http://schemas.opengis.net/sos/2.0/sosGetFeatureOfInterest.xsd
  http://www.opengis.net/sampling/2.0 http://schemas.opengis.net/sampling/2.0/samplingFeature.xsd http://www.opengis.net/samplingSpatial/2.0
  http://schemas.opengis.net/samplingSpatial/2.0/spatialSamplingFeature.xsd http://www.opengis.net/gml/3.2 http://schemas.opengis.net/gml/3.2.1/gml.xsd">
  <sos:featureMember>
    <sams:SpatialSamplingFeature gml:id="sf_F840C4618DFE357BC7480B91F003BA9CC9AEDD08">
      <gml:description>System Campus > 08-BK-City > 2e Verdieping</gml:description>
      <gml:identifier codeSpace="tudelft-wlan">A-08-B-202</gml:identifier>
      <gml:name codeSpace="tudelft-wlan">A-08-B-202</gml:name>
      <sf:type xlink:href="http://www.opengis.net/def/samplingFeatureType/OGC-ON/2.0/SF_SamplingPoint"/>
      <sf:sampledFeature xlink:href="campus"/>
    </sams:SpatialSamplingFeature>
    <sams:Shape>
      <ns:Point xmlns:ns="http://www.opengis.net/gml/3.2" ns:id="point_sf_F840C4618DFE357BC7480B91F003BA9CC9AEDD08">
        <ns:pos srsName="http://www.opengis.net/def/crs/EPSSG/0/4326">52.00565620980589 4.370536981524358</ns:pos>
      </ns:Point>
    </sams:Shape>
    </sams:SpatialSamplingFeature>
  </sos:featureMember>
  <sos:featureMember>
    <sams:SpatialSamplingFeature gml:id="sf_5C44885A08D61A0A979395141915E93170CA8B19">
      <gml:description>System Campus > 08-BK-City > 2e Verdieping</gml:description>
      <gml:identifier codeSpace="tudelft-wlan">A-08-B-203</gml:identifier>
      <gml:name codeSpace="tudelft-wlan">A-08-B-203</gml:name>
      <sf:type xlink:href="http://www.opengis.net/def/samplingFeatureType/OGC-ON/2.0/SF_SamplingPoint"/>
      <sf:sampledFeature xlink:href="campus"/>
    </sams:SpatialSamplingFeature>
    <sams:Shape>
      <ns:Point xmlns:ns="http://www.opengis.net/gml/3.2" ns:id="point_sf_5C44885A08D61A0A979395141915E93170CA8B19">
        <ns:pos srsName="http://www.opengis.net/def/crs/EPSSG/0/4326">52.00565620980589 4.370536981524358</ns:pos>
      </ns:Point>
    </sams:Shape>
    </sams:SpatialSamplingFeature>
  </sos:featureMember>
</sos:GetFeatureOfInterestResponse>
```

Figure A.1: Sample request: GetFeatureOfInterest

Figure A.1 shows a sample request. The request that was sent to the SOS was
<http://localhost:8080/52n-sos-webapp/service?service=SOS&request=GetFeatureOfInterest&version=2.0.0>

```
<?xml version='1.0' encoding='UTF-8'>
<sos:Capabilities xmlns:sos="http://www.opengis.net/sos/2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ows="http://www.opengis.net/ows/1.1"
  xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:fes="http://www.opengis.net/fes/2.0" version="2.0.0" xsi:schemaLocation="http://www.opengis.net/fes/2.0
  http://schemas.opengis.net/fes/2.0/fes.xsd http://www.opengis.net/sos/2.0 http://schemas.opengis.net/sos/2.0/sosGetCapabilities.xsd
  http://www.opengis.net/ows/1.1 http://schemas.opengis.net/ows/1.1/owsAll.xsd">
  <ows:ServiceIdentification>
    <ows:Title xml:lang="eng">52N SOS</ows:Title>
    <ows:Abstract xml:lang="eng">
      52North Sensor Observation Service - Data Access for the Sensor Web
    </ows:Abstract>
    <ows:ServiceType>OGC:SOS</ows:ServiceType>
    <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
    <ows:ServiceTypeVersion>2.0.0</ows:ServiceTypeVersion>
  </ows:ServiceIdentification>
  <ows:Profile>
    http://www.opengis.net/extension/SOSDO/1.0/observationDeletion
  </ows:Profile>
  <ows:Profile>
    http://www.opengis.net/spec/OMXML/1.0/conf/categoryObservation
  </ows:Profile>
  <ows:Profile>
    http://www.opengis.net/spec/OMXML/1.0/conf/countObservation
  </ows:Profile>
  <ows:Profile>
    http://www.opengis.net/spec/OMXML/1.0/conf/geometryObservation
  </ows:Profile>
  <ows:Profile>
    http://www.opengis.net/spec/OMXML/1.0/conf/measurement
  </ows:Profile>
  <ows:Profile>
    http://www.opengis.net/spec/OMXML/1.0/conf/textObservation
  </ows:Profile>
  <ows:Profile>
    http://www.opengis.net/spec/OMXML/1.0/conf/truthObservation
  </ows:Profile>
  <ows:Profile>
    http://www.opengis.net/spec/OMXML/2.0/conf/categoryObservation
  </ows:Profile>
</sos:Capabilities>
```

Figure A.2: Sample request: GetCapabilities

Figure A.2 shows a sample request. The request that was sent to the SOS was
<http://localhost:8080/52n-sos-webapp/service?service=SOS&request=GetFeatureOfInterest&version=2.0.0>