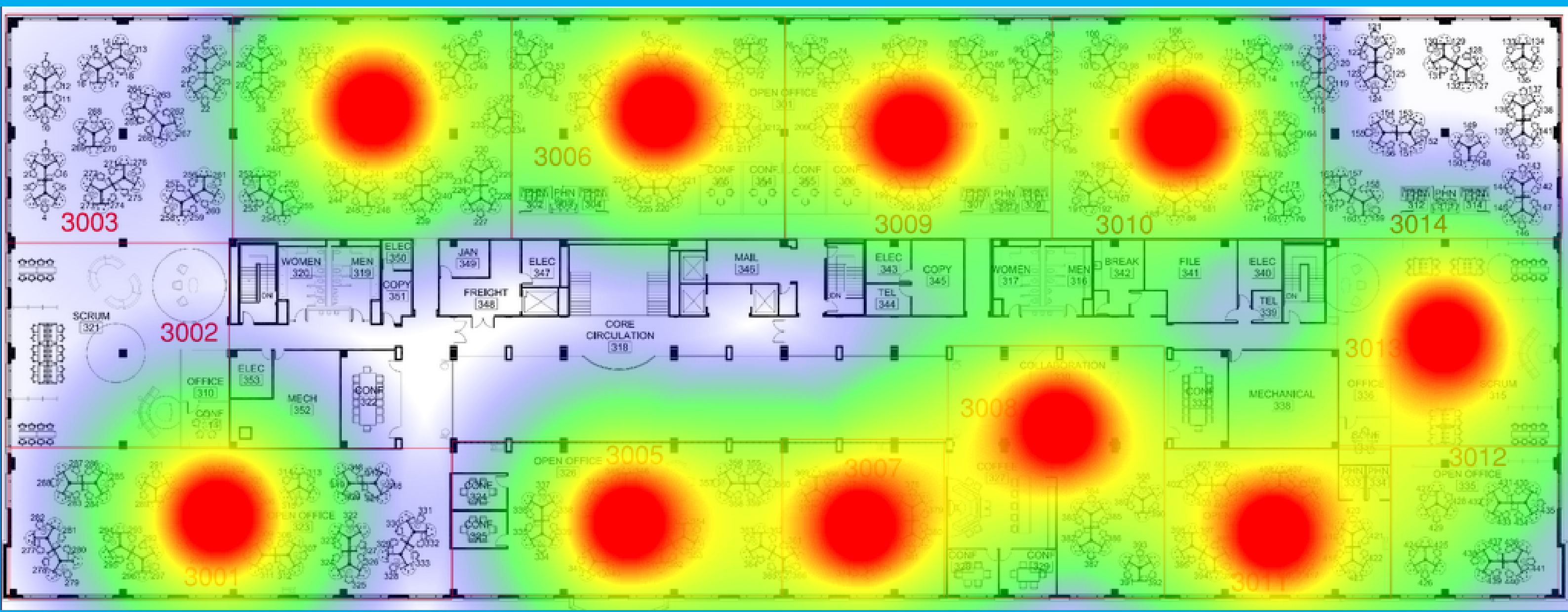
Implementing a Sensor Observation Service for Wi-Fi tracking data

Matthijs Bon
Xander den Duijn
Balazs Dukai



TU Delft

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

TU Delft

Preface

Matthijs Bon
Xander den Duijn
Balazs Dukai
Delft, June 2016

Contents

1

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 devices 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. Time series tracking data (subset WiFi-logs with time range) in the client	Functionality to download raw WiFi-logs or trajectories.
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 implementaion 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 requriements of the SOS 2.0 specification, implementing all of its extensions and their operations:

1. **Core**: GetCapabilities, GetObservation, DescribeSensor
2. **Enhanced**: GetFeatureOfInterest, GetObservaitonById
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**
codespace	contains the {codespace} and {codespacename} for the identifiers of Procedure, ObservableProperty, FeatureOfInterest and Offering
featureOfInterest	the geometries of the observations
featureOfInterestType	the type of the geometries of the observations
observableProperty	parameters that are observed
series	describes a series, a combination of featureOfInterest, observableProperty and procedure
observation	contains the observations
observationHasOffering	relates the offerings to the observations
offering	needed for structuring the data in a SOS server
procedure	the processes thorough which the observed values were generated
procedureDescriptionFormat	the format in which procedures shall be described
unit	units of a measurement
textvalue	contains the measured values, which are of type OM_TextObservation
observationConstellation	look-up table to check if the observation type of the InsertObservation, request is valid for the constellation procedure, observableProperty,,and offering
observationType	stores the observation types

Table 2.1: Tables in the Core database model

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

Table 2.2: Tables in the Transactional database model

3

Implementation

3.1. Preprocessing

The first step, before inserting the data into the tables, would be to create copy of (a part of) the wifilog datasource, and make it part of the SOS datamodel. Each record in the raw wifilog datasource can be seen as an observation of a device by an access point at a certain location. The building of which the access point is located in is known, but the exact location in the building is not known. For this reason the access points in a building are snapped to one point (approximately the center of the building). This has as a result that for instance, all access points in the Faculty of Architecture will have the same location. The table containing each access point, including the geometry. The geometrical location of each building on the campus and the raw wifilog datasource are used to create the access_points table. This table contains each and every access points including the geometry. The location of the access points are required for the sensor observation service.

3.2. Filling the tables

The data is inserted into the tables using a combination of Python and SQL scripts (see code document). This subsection will describes our implementation of the wifilog datasource into the 52North SOS datamodel.

The codeSpace (see Table 3.1) is the attribute for the identifier (gml:identifier) of the access points. As the GML 3.2. schema defines the gml:identifier is a “special identifier assigned to an object by the maintaining authority with the intention that it is used in references to the object.” 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.

Both the identifier and name are corresponding to the name of the measurement station. In our case this is the name of the access point (apname) as it is stated in the wifilog database.

In our case the featureOfInterest (see Table 3.3) is a Wi-Fi access point, with their concrete locations of measurements represented through sampling points (the featureofinteresttype).

We could distinguish four different observable properties (phenomena, see Table 3.4); mac, netid, rssi and snr. In our implementation the mac address, which is a unique identifier assigned to the device, is used, since complex values are only supported for SOS version 4.4.

Table 3.5 describes a serie, a combination of a featureOfInterest, observableProperty and procedure. The number of series is equal to the number sensors, the Wi-Fi access points. This means that every feature of interest contains one single unique serie.

codespaceid	1
codespace	tudelft-wlan

Table 3.1: codeSpace table

featureOfInterestId	id of the featureOfInterest
hibernatediscriminator	F
featureofinteresttypeid	1 (there is only one type of featureofinterest)
identifier	the name of the measurement station (the Wi-Fi access point, i.e. A-12-0-005)
codespace	1; the (only) entry in the codespace table
name	the name of the measurement station (the Wi-Fi access point, i.e. A-12-0-005)
codespaceName	1; the (only) entry in the codespace table
description	the maploc i.e. System Campus; the textual description of this feature 23-CITG Beganegrond
geom	the geometry of the station
descriptionXml	
url	

Table 3.2: FeatureOfInterest table

featureOfInterestTypeId	1; id of the featureOfInterestType
featureOfInterestType	http://www.opengis.net/def/samplingFeatureType/OGC-OM/2.0/ SF_SamplingPoint; featureOfInterestType value

Table 3.3: FeatureOfInterestType table

observablepropertyid	1; the id of the observableProperty (which is the mac address)
hibernatediscriminator	F
identifier	mac_address; the identifier of the observable property (phenomenon)
codespace	1; the (only) entry in the codespace table
name	Mac; the name of the observable property
codespacename	1; the (only) entry in the codespace table
description	Mac addresses scanned by wifi access point; long name of the phenomenon
disabled	F

Table 3.4: ObservableProperty

seriesid	id of the series
featureofinterestid	pointing to the entry in the featureOfInterest table
observablepropertyid	1; pointing to the first entry in the observableProperty table (the mac adress)
procedureid	1; the (only) entry in the procedure table
deleted	F
published	
firsttimestamp	
lasttimestamp	
firstnumericvalue	
lastnumericvalue	
unitid	

Table 3.5: Series table

observationid	id of the observation
seriesid	pointing to the entry in the series table
phenomenontimestart	asstime i.e. 2016-04-04 16:45:13; timestamp for which the measured value applied
phenomenonTimeEnd	asstime+sesdur i.e. 2016-04-04 17:52:00; end of the measurement
resulttime	asstime+sesdur i.e. 2016-04-04 17:52:00; end of the measurement
identifier	
codespace	1; the (only) entry in the codespace table
name	Scan; the name of the observation
codespacename	
description	WiFi Scan; the textual description of this observation
deleted	
validtimestart	
validtimeend	
unitid	1; the (only) entry in the unit table
samplinggeometry	the geometry of the station

Table 3.6: My caption

observationid	**id of the observation**
offeringid	1; the (only) entry in the offering table

Table 3.7: ObservationHasOffering

procedureid	1; the (only) id of the procedure
hibernatediscriminator	F
proceduredescriptionformatid	1; the (only) entry in the proceduredescriptionformatid table
identifier	wifi_access_point, the identifier of the procedure
codespace	1; the (only) entry in the codespace table
name	Scan; the name of the procedure
codespacename	1; the (only) entry in the codespace table
description	Scan; the name of the observation
deleted	F
disabled	F
descriptionfile	
referenceflag	F

Table 3.8: My caption

proceduredescriptionformatid	id of the procedureDescriptionFormat
proceduredescriptionformat	http://www.opengis.net/sensorML/1.0.1; procedureDescriptionFormat value

Table 3.9: My caption

unitid	**1; id of the unit**
unitid	mac adress; unit value representing a unit of measurement

Table 3.10: My caption

observationid	**id of the observation, pointing to the entry in the observation table**
value	the measured value i.e. rhQO/vg4AnEVjXNqgQscJu8Q3bXC+JiBL5e4FR8ZxPE= (hashed mac adress)

Table 3.11: My caption

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.

```
SELECT DISTINCT identifier AS "Identifier",
        ST_ASTEXT(geom) AS "Geometry",
        ST_SRID(geom) AS "SRID"
FROM featureofinterest;
```

	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 the retrieve the observations with KVP binding and *GetObservation* procedure, using the extension *MergeObservationsIntoDataArray*. This procedure returnes 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 we expected. 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.
The company Geonovum created their own SOS pilot project (see website) and they are using SOS Transactions (SOS-T) to populate the tables in the database. In their opinion populating the tables directly is cumbersome and error prone and we can confrim that. Therefore we also recommend to use SOS-T in future applications.

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 on 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. The SES could open up applications in the field of real-time occupation monitoring for facility management organizations.
Visualizing the observation results would provide additional insight into the observed phenomenon. In the case of Wi-Fi tracking data, aggregated measurements, e.g. number of people over time are often of more value than the raw measurements. Therefore future research could investigate the posibilites to publish and visualize aggregated measurements, eventually to provide the option for the user to switch between the two.

A

Appendix: Sample requests


```xml
<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:SF_SpatialSamplingFeature gml:id="ssf_FB40C4618DFE357BC7480B91F003BA9CC9AEDD08">
      <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-OM/2.0/SF_SamplingPoint"/>
      <sf:sampledFeature xlink:href="campus"/>
      <sams:shape>
        <ns:Point xmlns:ns="http://www.opengis.net/gml/3.2" ns:id="point_ssf_FB40C4618DFE357BC7480B91F003BA9CC9AEDD08">
          <ns:pos srsName="http://www.opengis.net/def/crs/EPSG/0/4326">52.00565620980589 4.370536981524358</ns:pos>
        </ns:Point>
      </sams:shape>
    </sams:SF_SpatialSamplingFeature>
  </sos:featureMember>
  <sos:featureMember>
    <sams:SF_SpatialSamplingFeature gml:id="ssf_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-OM/2.0/SF_SamplingPoint"/>
      <sf:sampledFeature xlink:href="campus"/>
      <sams:shape>
        <ns:Point xmlns:ns="http://www.opengis.net/gml/3.2" ns:id="point_ssf_5C44885A08D61A0A979395141915E93170CA8B19">
          <ns:pos srsName="http://www.opengis.net/def/crs/EPSG/0/4326">52.00565620980589 4.370536981524358</ns:pos>
        </ns:Point>
      </sams:shape>
    </sams:SF_SpatialSamplingFeature>
  </sos:featureMember>
```


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
<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/filter/2.0/filterAll.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/1.1.0/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: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>
```


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`


```
-<sos:GetObservationResponse xsi:schemaLocation="http://www.opengis.net/swes/2.0 http://schemas.opengis.net/swes/2.0/swes.xsd
 http://www.opengis.net/sos/2.0 http://schemas.opengis.net/sos/2.0/sosGetObservation.xsd http://www.opengis.net/gml/3.2 http://schemas.opengis.net
 /gml/3.2.1/gml.xsd http://www.opengis.net/om/2.0 http://schemas.opengis.net/om/2.0/observation.xsd">
 -<sos:observationData>
   -<om:OM_Observation gml:id="o_5">
      <gml:description>description</gml:description>
      <om:type xlink:href="http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_TextObservation"/>
    -<om:phenomenonTime>
      -<gml:TimePeriod gml:id="phenomenonTime_5">
          <gml:beginPosition>2016-04-04T18:27:35.000Z</gml:beginPosition>
          <gml:endPosition>2016-04-04T18:32:39.000Z</gml:endPosition>
        </gml:TimePeriod>
      </om:phenomenonTime>
    -<om:resultTime>
      -<gml:TimeInstant gml:id="ti_CFF4300A4BCAF8D0158DA47549B8D875FCE97B70">
          <gml:timePosition>2016-04-04T18:32:39.000Z</gml:timePosition>
        </gml:TimeInstant>
      </om:resultTime>
      <om:procedure xlink:href="wifi_access_point" xlink:title="Scan"/>
    -<om:parameter>
      -<om:NamedValue>
          <om:name xlink:href="http://www.opengis.net/def/param-name/OGC-OM/2.0/samplingGeometry"/>
        -<om:value xsi:type="ns:GeometryPropertyType">
          -<ns:Point ns:id="point_6A58A350F5DF8C544D15D14585BB61095BC21662">
              <ns:pos srsName="http://www.opengis.net/def/crs/EPSG/0/0">4.37096071809295 52.0028218180504</ns:pos>
            </ns:Point>
          </om:value>
        </om:NamedValue>
      </om:parameter>
      <om:observedProperty xlink:href="mac_address" xlink:title="Mac"/>
      <om:featureOfInterest xlink:href="A-30-0-036" xlink:title="A-30-0-036"/>
      <om:result xsi:type="xs:string">hJQOPV5UH1bGueYGNNTzKrmDLzpRRtRpZYY984UUGC0=</om:result>
    </om:OM_Observation>
  </sos:observationData>
```


Figure A.3: GetObservation with KVP binding and MergeObservationsIntoDataArray extension.

Figure A.3 shows a sample request. The request that was sent to the SOS was

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