52north exploring horizons

Sensor Web Enablement

Installation Guide
for
Sensor Observation Service
version 3.1.1

Document Change Control

Revision Number	*Date Of Issue*	*Author(s)*	*Brief Description Of Changes*
1.0	2008-02-11	Stephan Künster	Added architecture and adapted howto to SOS version 3.0.0
1.1	2009-01-26	Julian Kuhlmann Carsten Hollmann	Adapted the howto to SOS version 3.0.1
1.2	2009-03-04	Carsten Hollmann	Changed from Ant to Maven build tool.
1.3	2009-08-17	Stephan Künster Alexander Strotmann	Adapted the HowTo to SOS version 3.1.0
1.4	2010-03-30	Carsten Hollmann	Adapted the HowTo to SOS version 3.1.1

Editors

Alexander C. Walkowski

Email: walkowski@52north.org

Christoph Stasch

Email: staschc@52north.org

Stephan Künster

Email: skuenster@uni-muenster.de

Julian Kuhlmann

Email: juliankuhlmann@uni-muenster.de

Carsten Hollmann

Email: carsten.hollmann@uni-muenster.de

Alexander Strotmann

Email: strotmann@uni-muenster.de

Table of contents

1 Introduction

1.1 Scope

This document describes the architecture and the install process of the 52 North Sensor Observation Service (SOS).

1.2 What you are doing following the installation procedure?

You will check out the SOS package from the SVN repository or you might also use the sources provided in the downloaded zip file from the 52°North homepage. The package contains an implementation of the Sensor Observation Service (SOS) as a Java Servlet. Executing the installation steps [build] will deploy this service in your Apache Jakarta Tomcat web container as an web application (webapp).

1.3 Some words on the SOS

The Sensor Observation Service (Na & Priest 2007) 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. Four profiles are defined within the SOS specification: core, transactional, enhanced, and entire.

The current release implements the core profile comprising of the mandatory operations:

- *GetCapabilities*, for requesting a self-description of the service .
- *GetObservation*, for requesting the pure sensor data encoded in Observation&Measurements (O&M).
- *DescribeSensor*, for requesting information about the sensor itself, encoded in a Sensor Model Language (SensorML) instance document.

The recent development snapshot implements also the following optional operations:

- *GetFeatureOfInterest*, for requesting the GML encoded representation of the feature that is the target of the observation.
- *GetResult*, for periodically polling of sensor data.
- *RegisterSensor*, for registering new sensors.
- *InsertObservation*, for inserting new observations.

1.4 Architecture

The design of the 52°North Sensor Observation Service is based on a 4-tier web architecture as shown in figure 1 below.

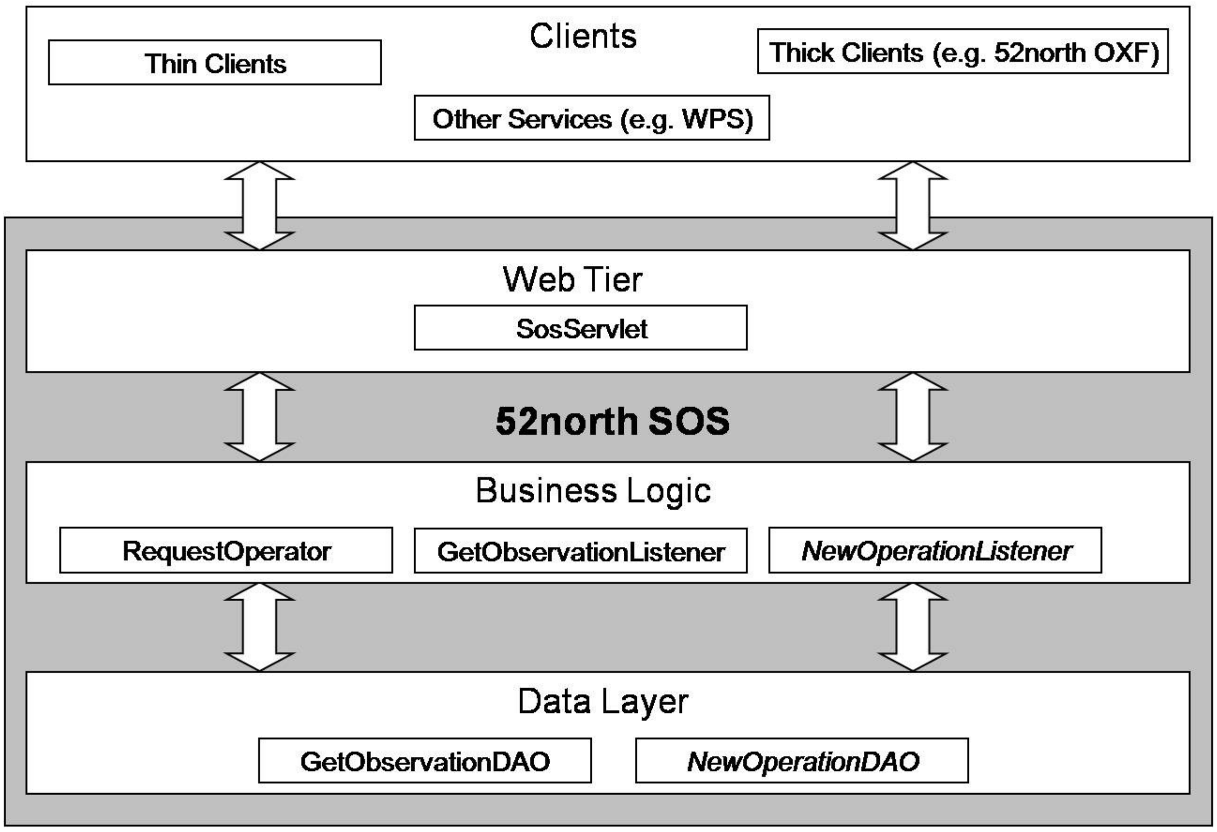


Figure 1: SOS Architecture

The lowest layer encapsulates the access to databases and/or -sources. The sources of sensor data are very heterogeneous and range from simple text files to very complex data models. To enable the user to use different data sources, the data access for each operation is implemented using the Data Access Object (DAO) pattern. This enables the user to easily adjust the 52° North SOS to already existing sensor databases or -sources through a new or changed implementation of the DAO implementations. By default the 52°North SOS uses as PostGIS database to store the observation values and corresponding meta data.

The central component of the Buiness Logic Layer is the RequestOperator. It receives requests from the Presentation Layer, validates the request and forwards the request to the appropriate OperationListener. The 52°North SOS contains Listeners for each supported operation, which are defined in an external config file (see section 3.5.1 for more informations). All Listeners implement a common interface. If you want to support an additional operation, all you have to do is to implement the Operation Listener, implement the corresponding DAO and add the Listener in the config file. The Business Logic Tier contains several other components, e.g. components for parsing/encoding responses.

The Web Tier of the 52°North SOS consists simply of a Servlet, which handles HTTP requests and responses. If you want to support other protocols, you have to replace this Servlet with another class for communication.

There are multiple clients, which can use the data of the 52°North SOS. Based upon the 52°North OX-Framework (see http://www.52north.org/oxf) both thin clients and thick clients could be developed for your sensor application.

2 Requirements

- Windows 2000 or higher [tested with Windows XP SP3]
- JRE/JDK 1.5 [1.6.0]
- Apache Jakarta Tomcat 5.5 or higher [6.0.xx]
- PostreSQL Version [8.4.x]
- PostGIS Version [1.4.x or 1.5]
- SVN-Client (if you want to download SOS package from the SVN repository)
- Apache Maven [2.2.1]

3 Installation Procedure

3.1 Get the programms

- Download Apache Jakarta Tomcat from:

  http://jakarta.apache.org/tomcat

  Follow the installation instructions given on the Apache website to install the Apache Jakarta Tomcat.
- Download PostgreSQL from:

  http://www.postgresql.org/download/

  Install PostgreSQL as descibed in the PostgreSQL manual available at

  http://www.postgresql.org/docs/manuals/
- Download PostGIS from:

  http://www.postgis.org/download/windows/

  or use the PostgreSQL → Application Stack Builder

  Install PostGIS as descibed in the PostGIS documentation available at

  http://postgis.org/documentation/
- or use the Application Stack Builder.

3.1.1 Configure Maven

- Edit the *settings.xml* from your conf folder located under the Maven install folder. Make the following additions.

Under the <*settings*> tag insert the path to your local repository:

```
<localRepository>
    [YourPathTo]\.m2\repository
</localRepository>
```

Check in your file browser if the path in correct!
Under the <*profiles*> tag insert the following profile:

```
<profile>
    <id>52n-start</id>
    <repositories>
        <repository>
                <id>n52-releases</id>
                <name>52n Releases</name>
            <url>http://52north.org/maven/repo/releases</url>
            <releases>
                <enabled>true</enabled>
            </releases>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
        <repository>
            <id>geotools</id>
            <name>Geotools repository</name>
            <url>http://maven.geotools.fr/repository</url>
        </repository>
        <repository>
            <id>Refractions</id>
            <name>Refractions repository</name>
            <url>http://lists.refractions.net/m2</url>
        </repository>
        <repository>
            <id>Apache</id>
            <name>Apache repository</name>
            <url>http://repo1.maven.org/maven2</url>
        </repository>
    </repositories>
</profile>
```

And after the <*profiles*> Section insert the following active profile:

```
<activeProfiles>
    <activeProfile>52n-start</activeProfile>
</activeProfiles>
```

3.2 Get the sources

- Check out the sources from 52°North-SVN-Repository, or use the sources provided by the 52N-SOS-3.1.1.zip from the SOS homepage at 52°North:

Host:	`http://52north.org/svn`
Repository:	`/swe/main/SOS/Service/trunk/SOS/52n-sos`
Module:	`SOS`
Branch:	`main`
Tag:	`HEAD`
User:	`anonymous`
Password:	`no password required – leave it blank`

3.3 Directory structure

- The directory structure of the 52N-SOS-3.1.1 is as follows (see Figure 2):

• `52n-sos-coding:`	source files
• `52n-sos-core:`	source files
• `52n-sos-dao-postgis:`	source files
• `52n-sos-ogc:`	source files
• `52n-sos-services:`	source files
• `db:`	sql scripts for the DB schema
• `doc:`	installation guide
• `jmeter:`	Apache Jmeter test files

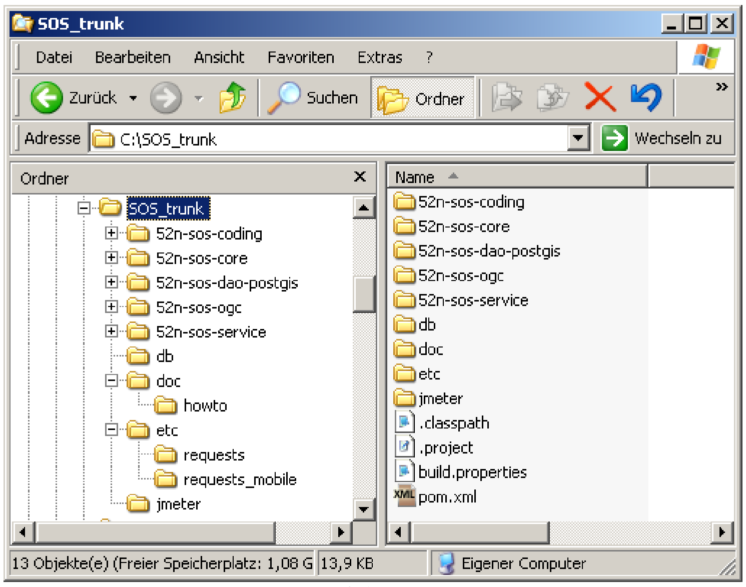


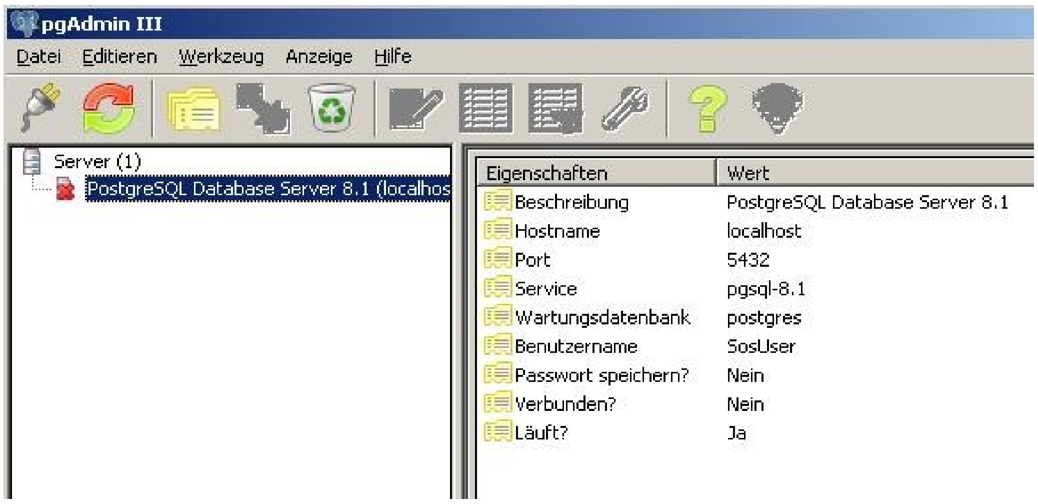
Figure 2: Directory structure of the SOS repository

3.4 Install and Create Database

- Install PostgreSQL
- Install PostGIS

3.4.1 Create database

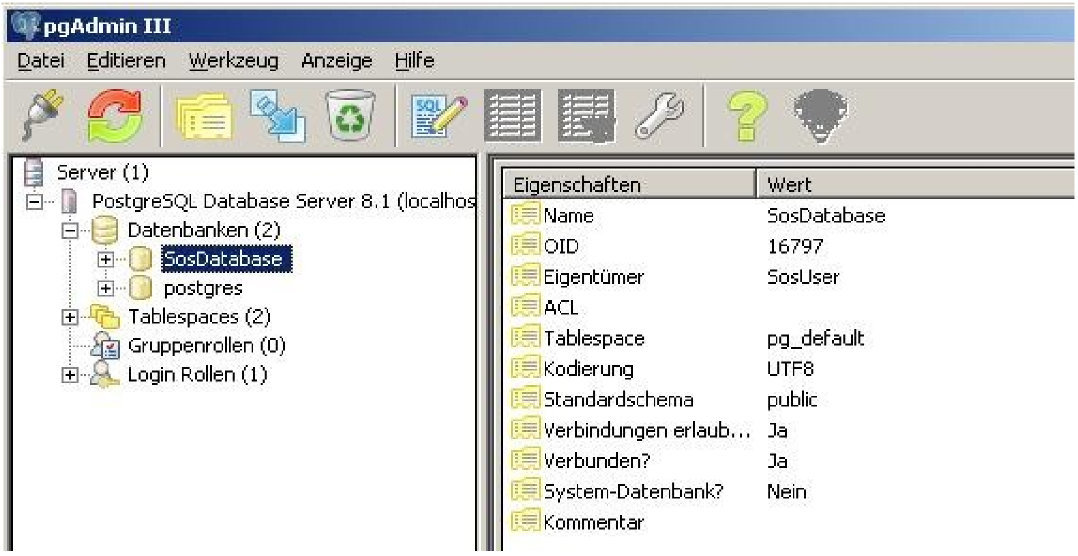
- Use the pgAdmin III – Tool. Start and connect to the database with pgAdmin (*in Windows XP: Start>Programms>PostgreSQL 8.x >pgAdmin III*). The following window appears:


pgAdmin III
Datei Editieren Werkzeug Anzeige Hilfe
Server (1)
PostgreSQL Database Server 8.1 (localhos

Eigenschaften	Wert
Beschreibung	PostgreSQL Database Server 8.1
Hostname	localhost
Port	5432
Service	pgsql-8.1
Wartungsdatenbank	postgres
Benutzername	SosUser
Passwort speichern?	Nein
Verbunden?	Nein
Läuft?	Ja

Figure 3: Start Window of the pgAdmin III

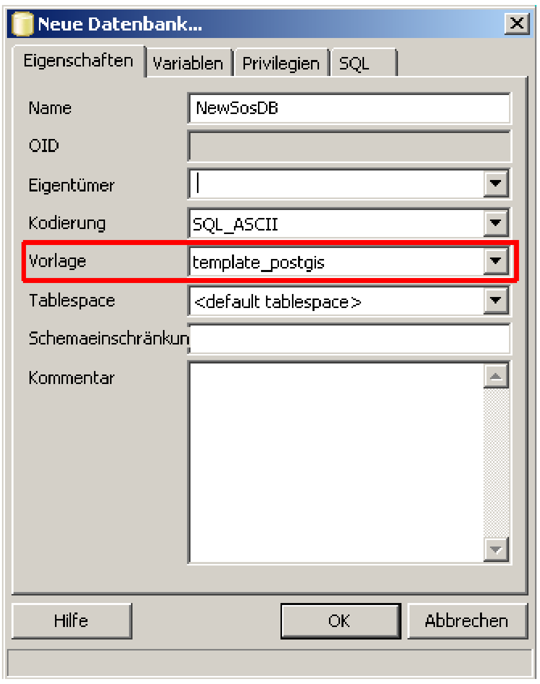
You find your „PostgresSQL Database Server 8.x“ in the upper left corner of the window. Right click on the „PostgresSQL Database Server 8.x“ and choose „Connect“ in the pop up menu. Type in the name and password of the superuser you have chosen in the installation steps. Now the red cross over the icon of the server disappears and you are connected to the server.



Eigenschaften	Wert
Name	SosDatabase
OID	16797
Eigentümer	SosUser
ACL	
Tablespace	pg_default
Kodierung	UTF8
Standardschema	public
Verbindungen erlaub…	Ja
Verbunden?	Ja
System-Datenbank?	Nein
Kommentar	

Figure 4: PostgreSQL Database Server 8.1 view in the pgAdmin III main window

- When you right click on “databases” choose “new databases”. In the opening window (Figure 4) you can create the new database for your SOS (e.g. SosDatabase) typing the needed information and set the “template” field to “template_postgis”.

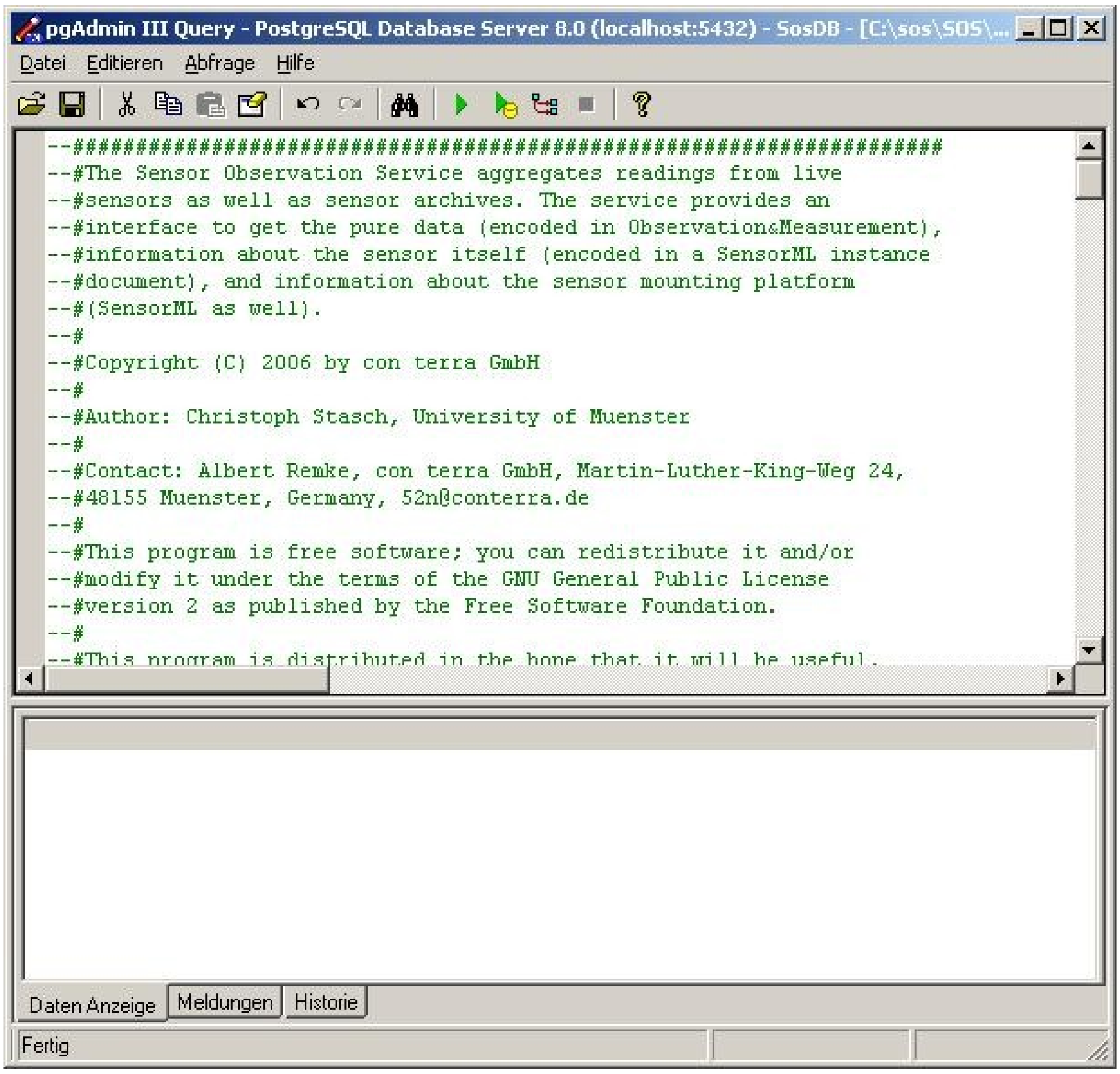


Name	NewSosDB
OID	
Eigentümer	
Kodierung	SQL_ASCII
Vorlage	template_postgis
Tablespace	<default tablespace>
Schemaeinschränkun	
Kommentar	

Figure 5: Create new database window.

3.4.2 Create the table structure of the database

- Start the pgAdmin (*see former chapter*).
- Now click on the button „Execute common SQL query“ in the toolbar. The query window of pgAdmin III appears.
- Click the „Open file“ button and navigate to the db-folder of your zip file. Open the file `datamodel_postgres83.sql`. The SQL statements now appear in the query field.


pgAdmin III Query - PostgreSQL Database Server 8.0 (localhost:5432) - SosDB - [C:\sos\SOS\...
Datei Editieren Abfrage Hilfe
Daten Anzeige Meldungen Historie
Fertig

```
--##########################################################################
--#The Sensor Observation Service aggregates readings from live
--#sensors as well as sensor archives. The service provides an
--#interface to get the pure data (encoded in Observation&Measurement),
--#information about the sensor itself (encoded in a SensorML instance
--#document), and information about the sensor mounting platform
--#(SensorML as well).
--#
--#Copyright (C) 2006 by con terra GmbH
--#
--#Author: Christoph Stasch, University of Muenster
--#
--#Contact: Albert Remke, con terra GmbH, Martin-Luther-King-Weg 24,
--#48155 Muenster, Germany, 52n@conterra.de
--#
--#This program is free software; you can redistribute it and/or
--#modify it under the terms of the GNU General Public License
--#version 2 as published by the Free Software Foundation.
--#
--#This program is distributed in the hope that it will be useful,
```

Figure 6: Query window of the pgAdmin III with opened datamodel_postgres82.sql file

- Now click the „Execute query“ button to execute the table. In the lower field appears the message „Query was succesfull.“ The tables are created now.
- Close the pgAdmin.

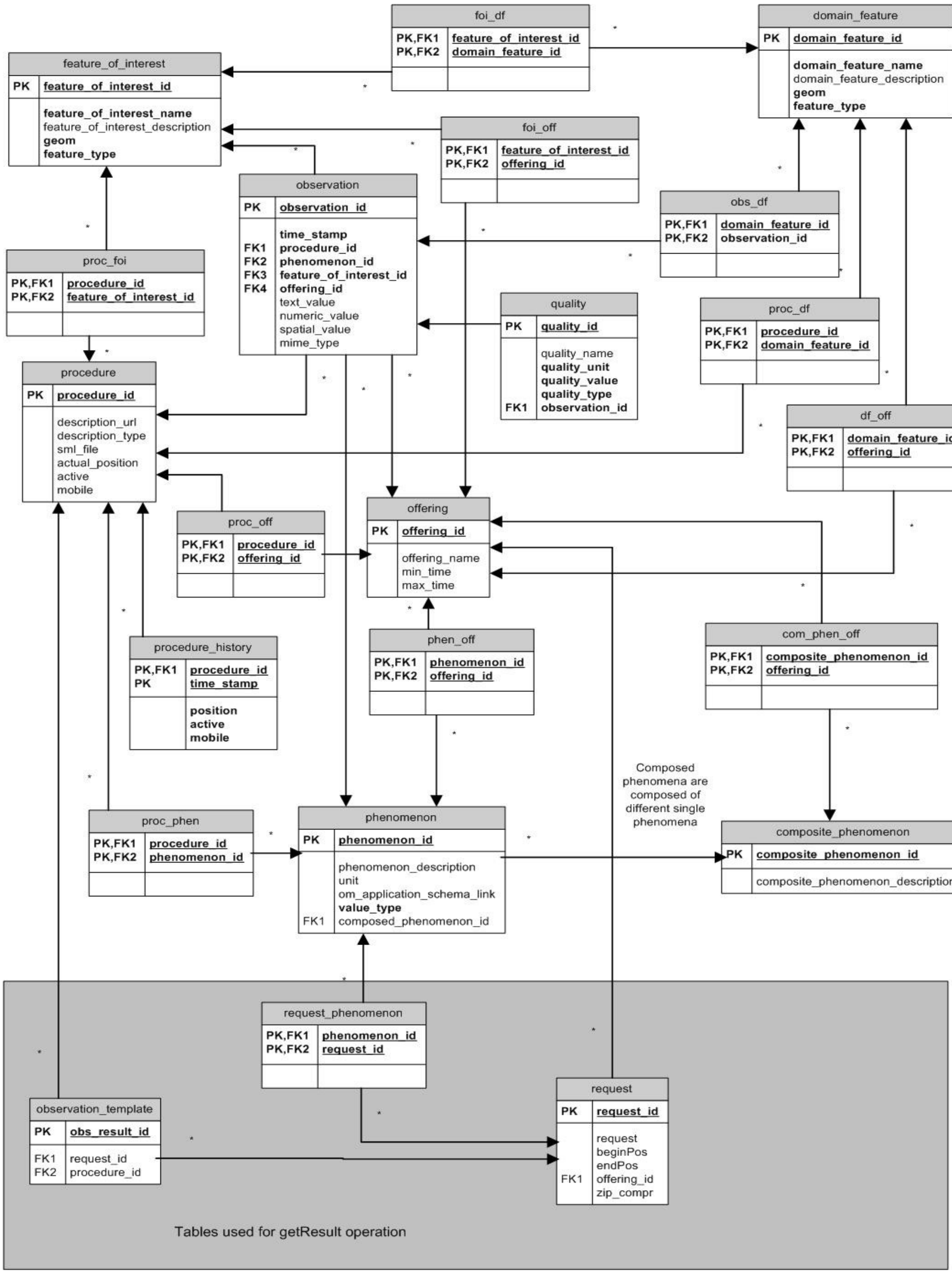


Figure 7: Sensor Observation Service database schema

3.4.3 Populate table structure

Figure 7 depicts the table structure of the database. The tables in the grey box of the figure will be automatically used by the SOS to store request parameters for a later getResult operation request.

The rest of the figure shows the „data“ tables of the SOS database. The SOS will use the tables above to answer incoming requests or to update the values .

The following tables are contained:

1. *feature_of_interest table* - the feature_of_interest table stores data about the feature of interest. The geom column holds the geometry of the feature_of_interest and is of the PostGIS type geometry.
2. *foi_off table* – the foi_off table realizes the many-to-many relationship between offerings and features of interest. Remember to insert the relationships if you have inserted new offerings and/or new features of interest!
3. *foi_df table* – the foi_df table realizes the many-to-many relationship between domain features and features of interest. Remember to insert the relationships if you have inserted new domain features and/or new features of interest!
4. *proc_foi table* – the proc_foi table realizes the many-to-many relationship between procedures and features of interest. Remember to insert the relationships if you have inserted new procedures and/or new features of interest!
5. *observation table* – the observation table aggregates the data of an observation event like time, procedure (sensor or group of sensors), the feature of interest and the observation value, which is stored in a seperate table. Note that the columns observation_id, feature_of_interest_id, and procedure_id are foreign keys. You have to ensure that the values you want to insert in this columns are contained in the tables they reference on.
6. *quality table* – the quality table stores quality attributes for an observation. Qualities are optional and have not to be set. In this case set 'conf.sos.supportsQualit'y in config.properties file to 'false'.
7. *procedure table* – the procedure table stores data about the procedure. Only the procedure_id which should be the URN of the procedure as specified by the OGC must be contained.
8. *proc_off table* – the proc_off table realizes the many-to-many relationship between procedures and offerings. Remember to insert the relationships if you have inserted new procedures and/or new offerings!
9. *offering table* – the offering table stores each offering of this SOS. This table is only used when the SOS is initialized to read in the offerings of this SOS

(e.g. gauge height) and the phenomena which are related to each offering. When you not use the RegisterSensor-Operation you have to note that if you have inserted new offerings, you have to restart your SOS to enable the changes.

10. *phen_off table* - the phen_off table is created to represent the many-to-many relationship between offerings and phenomena.

11. *composite_phenomenon table* – the composite_phenomenon stores composite phenomena.

12. *com_phen_off table* – the com_phen_off table realizes the many-to-many relationship between composite phenomena and offerings. Remember to insert the relationships if you have inserted new composite phenomena and/or new offerings!

13. *phenomenon table* – the phenomenon table represents phenomena. In the context of the new SOS specification phenomena are also called observedProperties. Only the phenomenon_id and value_type are required. The phenomenon_id should contain the URN of the phenomenon as specified by the OGC. The possible values of the value_type column are:

    - **integerType, doubleType**, **floatType** for numerical values
    - **textType** for textual (categorical) values

1. *proc_phen table* – the proc_phen table realizes the many-to-many relationship between procedures and phenomena. Remember to insert the relationships if you have inserted new procedures and/or new phenomena!

2. *domain_feature table* – the domain_feature table stores domain features for the SOS. It represents an area which contains many sensors. In other words it is the investigation area.

3. *df_off table* - the df_off table realizes the many-to-many relationship between domain features and offerings. Remember to insert the relationships if you have inserted new domain features and/or new offerings!

4. *proc_df table* - the proc_df table realizes the many-to-many relationship between procedures and domain features. Remember to insert the relationships if you have inserted new procedures and/or new domain features!

5. *proc_off table* – the proc_off table realizes the many-to-many relationship between procedures and offerings. Remember to insert the relationships if you have inserted new procedures and/or new offerings!

6. *procedure_history table* – the procedure history is needed for the mobile enabled SOS. In this table currently old positions of the registered sensors are stored.

7. *request table* - the request table is only used for the getResult operation. It realizes the many-to-many relationship between procedures and requests.
8. *request_phenomenon table* - the request_phenomenon table is only used for the getResult operation. It realizes the many-to-many relationship between phenomenon and requests.
9. *request_composite_phenomenon table* - the request_composite_phenomenon table is only used for the getResult operation. It stores request composite phenomena.
10. *observation_template table* - the observation_template table is only used for the getResult operation. It realizes the many-to-many relationship between procedures and requests.

- If you want to **insert example data** which can be requested in the test section (chapter 3.6) open and execute the `test.sql` file.

3.5 Configure the properties

3.5.1 Configure pom.xml

- Open the `pom.xml` file and edit the properties followed by

```
<!-- ##############################################
##SOME OF THE FOLLOWING PROPERTIES HAVE TO BE CHANGED!!! ##
#################################################-->
```

The properties followed by

```
<!-- ##########################################################
##FOLLOWING PROPERTIES ONLY TO BE CHANGED FOR ADVANCED USERS!!! ##
#################################################-->
```

are for advanced users.
In the profiles section of the pom.xml move to profile with id 'with-deploy'. There are three more properties.
The properties in the pom.xml look like the following example:

```
<Property>value</Property>
```

In Table 1 all changeable properties from the pom.xml are listed. You find the different properties listed in Table 1. At the beginning of the table, you find properties you have to change (MANDATORY) or you can change (OPTIONAL). After this section the properties for advanced users are described.

Table 1: SOS configuration properties (pom.xml)

Property	Explanation
The following properties must (MANDATORY), can (OPTIONAL) be changed.	
conf.sos.name	In the **profiles section**. Profile with id **with-deploy**. The name of the web application (which is 52nSOSv3 by default). If you prefer another name you can change the name. (OPTIONAL)
deploy.target.host	DNS name or IP of the target host (OPTIONAL if necessary)
deploy.target.port	Port of the target host (OPTIONAL if necessary)
deploy.tomcat.manager.url	URL pointing to Tomcat Manager (NO CHANGES)
deploy.tomcat.manager.username	Tomcat Manager username (MANDATORY)
deploy.tomcat.manager.password	Tomcat Manager password (MANDATORY)
deploy.tomcat.home	installation directory of the tomcat servlet engine (MANDATORY)
conf.sos.ds.connectionstring	The connection string to your database. (e.g. jdbc:postgresql://localhost:5432/SosDB for PostgreSQL) (MANDATORY)
conf.sos.ds.user	The user name for your access to the database server (MANDATORY).
conf.sos.ds.password	The password for your access to the database server (MANDATORY).
The following properties are for advanced users	
conf.sos.capabilitiesCacheController	Class of capabilities cache controller (ADVANCED)
conf.sos.capabilitiesCacheUpdateIntervall	Capabilities Cache Update Interval in minutes (0 = no automatic update) (ADVANCED)
conf.sos.ds.daofactory	In the **profiles section**. Profile with id **with-deploy**. Class name of the data access object factory, including the package name (e.g. org.n52.sos.ds.pgsql.PGSQLDAOFactory). The SOS realizes the data access object pattern and therefore uses a factory for the data access objects. Change this only if you are an advanced user and you are using another data source! (ADVANCED)
conf.sos.ds.driver	The database jdbc driver (e.g. for PostgreSQL org.postgresql.Driver) Change this if you use another database system. (ADVANCED)
conf.sos.ds.initcon	Initial number of connections of the connection pool the SOS uses. It is not recommended to change this. (OPTIONAL)
conf.sos.ds.maxcon	Maximal number of connections of the connection pool the SOS uses. It is not recommended to change this. (OPTIONAL)

Property	Explanation
conf.sos.listeners	Comma separated list of the request listeners which are implemented (without white space!). Change this only if you have implemented further request listeners to support further operations. Following listeners are implemented GetCapabilitiesListener GetObservationListener GetObservationByIdListener DescribeSensorListener DescribeFeatureTypeListener DescribeObservationTypeListener GetResultListener GetFeatureOfInterestListener GetFeatureOfInterestTimeListener InsertObservationListener RegisterSensorListener (ADVANCED)
conf.sos.skeletonfile	Absolute path and name of the skeleton file for the capabilities document. Change this only, if you want to store the file in another directory. (ADVANCED)
conf.sos.skeletonfilemobile	Absolute path and name of the mobile skeleton file for the capabilities document. Change this only, if you want to store the file in another directory. (ADVANCED)
conf.sos.sensordir	The directory where the SensorML documents for each sensor (procedure) are stored. Change this only, if you want to store them in another directory. (ADVANCED)
conf.sos.omEncoder	Implementation of IOMEncoder used to encode observations (has to be reimplemented, if new observation types should be supported) (ADVANCED)
conf.sos.gmlEncoder	GMLEncoder implementation (implementation of IGMLEncoder) (ADVANCED)
conf.sos.postRequestDecoder	HttpPostRequestDecoder implementation (implementation of IhttpPostRequestDecoder) (ADVANCED)
conf.sos.getRequestDecoder	HttpGetRequestDecoder implementation (implementation of IhttpGetRequestDecoder) (ADVANCED)
conf.sos.getResponseEncoder	ResponseEncoder implementation (implementation of IresponseEncoder) (ADVANCED)
conf.sos.sensorMLEncoder	SensorMLEncoder implementation (implementation of IsensorMLEncoder) (ADVANCED)

Property	Explanation
conf.sos.loglevel	The level which determines which log messages will be written into the log file. The standard is INFO. Below is a listing which levels are possible. It is not recommended to change the level. SEVERE (highest value) WARNING INFO CONFIG FINE FINER FINEST (lowest value) (OPTIONAL)
conf.sos.mobileEnabled	Property indicates, whether SOS supports mobile requests ; (default := true) (ADVANCED)
conf.sos.gmlDateFormat	gml date format: yyyy-MM-dd'T'HH:mm:ssZ (ADVANCED)
conf.sos.characterEncoding	Character encoding for response documents (ADVANCED)
conf.sos.srs.prefix	prefix URN for the spatial reference system (ADVANCED)
conf.sos.supportsQuality	Property indicates, whether SOS supports quality informations in observations or not; (default := false) (ADVANCED)
conf.sos.switchCoordinatesForEPSG	property keeps a list of all EPSG codes for which the SOS has to switch coordinates from long/lat to lat/long; PostgreSQL users please read the important note in section 4! (ADVANCED)
conf.sos.foiEncodedInObservation	Property indicates, whether SOS encodes the complete FOI-instance within an Observation instance or just the FOI id; (default := true) (ADVANCED)
conf.sos.logdir	The directory where the log file will be stored. (ADVANCED)
conf.sos.result.lease	Time of lease for result template in getResult operation (in minutes) (ADVANCED)
conf.sos.result.tokenseperator	Token seperator in result element (ADVANCED)
conf.sos.result.tupleseperator	Tuple seperator in result element (ADVANCED)
conf.sos.result.decimalSeperator	Decimal separator in result element (ADVANCED)
conf.sos.result.nodatavalue	No data value for result string containing the values in common observation and getResult response (ADVANCED)
conf.sos.serviceversion	The version of this SOS. (DO NOT CHANGE!)
conf.sos.service.url	URL of SOS web application
dssos.config.file.name	In the **profiles section**. Profile with id **with-deploy**. Sets the config file for database connection and tables. (DO NOT CHANGE!)

- Save changes
- Build the web application

In the following subsections we will define the capabilities skeleton and the sensor descriptions. As final step we will create the web application.

The static information (information about the service provider) of the capabilities document is defined in the capabilities skeleton. The dynamic part (e.g. the time range, the listings of procedure Ids and phenomena IDs) will be retrieved from the database after receiving a GetCapabilities request. The accommodation of the capabilities skeleton file is described in the section 3.5.2 .

Upon receiving a DescribeSensor request, the SOS will response with an SensorML encoded description of the platform sensor. The sensor descriptions will be defined in section 3.5.32 .

The deployment of the web application is automated by the Apache Maven build tool. The deployment is described in section 3.5.43 .

3.5.2 Adjust the capabilities skeleton

- Navigate to the `52n-sos-service/src/main/webapp/WEB-INF/conf/capabilities`-directory of your SOS-directory. Open the `capabilities_skeleton.xml` file with an editor and change the following sections:
    - *ServiceIdentification*: change the title of the SOS.
    - *ServiceProvider*: change all data of the service provider.
- Save your changes.
- The time ranges, list of procedure Ids as well as phenomenon Ids and the whole contents section will be updated automatically by the SOS.

3.5.3 Provide Sensor-Descriptions

If you use the RegisterSensor-Operations for inserting new Sensors, you can step to the next clause.

- Create an SensorML instance document for each sensor. Refere to the SensorML specification 1.0.1 (Botts 2007) for further information on SensorML.
- Store the instance documents in the `52n-sos-service/src/main/webapp/WEB-INF/conf/sensors` directory of your SOS directory.
- Make sure that the names of the documents are the same as the last part of their URNs. These URNs correspond to the procedure Ids in our data model (e.g. Name the SensorML file „`ifgi-sensor-1a.xml`“ if the URN is

„`urn:ogc:def:procedure:ifgi-sensor-1a`“).

- The maven-script will copy the sensor descriptions to the [Tomcat Home]/webapps/[webapp.name]/WEB-INF/conf/sensors directory. If the sensor description has to be modified, change the SensorML document in the [Tomcat Home]/webapps/[webapp.name]/WEB-INF/conf/sensors directory.
- `Ifgi-sensor-1.xml` is an example sensor description for a simple water level station.

3.5.4 Deploy the web application

- Make sure that your Tomcat and your Postgres are started.
- Assume your main directory of your local SOS repository ist `C:\SOS`
- Open a command line and go to the project's folder
- Type the following command line expression:
  - `mvn -Pwith-deploy install`
  - if the building fails, proceed to the troubleshooting section in chapter 5.
- The SOS is now available.

3.6 Tests

3.6.1 Example for GetCapabilities request

The GetCapabilities-Request is the only request which is possible to be sent via HTTP GET- and HTTP POST-Request. The SOS-Webclient (see below) is attached to enable POST Request to the SOS. Make sure that you have executed the file test.sql in the pgAdmin as described in chapter 3.4 .

3.6.1.1 Capabilities-Request via HTTP GET

- Open your browser. Type in

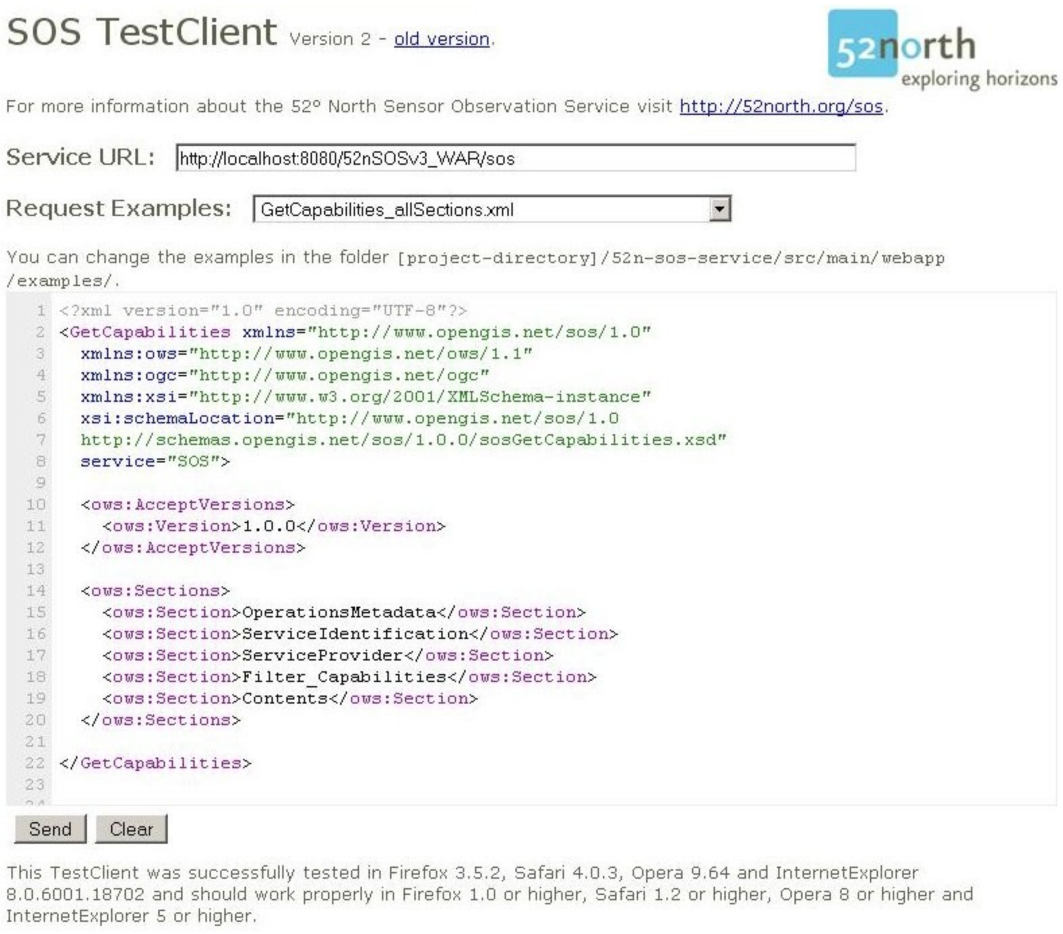
http://localhost:8080/52nSOSv3/sos?REQUEST=GetCapabilities&SERVICE=SOS&ACCEPTVERSIONS=1.0.0

- The capabilities response document is now shown in your browser.
- You can extend the request above through appending the optional parmeters SECTIONS, ACCEPTFORMATS and UPDATESEQUENCE. Look at the SOS specification for more details.

3.6.1.2 Capabilities Request via HTTP POST:

- Open your Internet Browser (e.g. Mozilla Firefox). You can find the SOS Test Client under http://localhost:8080/52nSOSv3/testClient-v2.html. (Make sure that your tomcat is running)

- Chose a request from the drop down list named 'Request Examples'.


SOS TestClient Version 2 - old version.
52north exploring horizons
Send
Clear

For more information about the 52° North Sensor Observation Service visit http://52north.org/sos.

Service URL: http://localhost:8080/52nSOSv3_WAR/sos

Request Examples: GetCapabilities_allSections.xml

You can change the examples in the folder [project-directory]/52n-sos-service/src/main/webapp/examples/.

```
<?xml version="1.0" encoding="UTF-8"?>
<GetCapabilities xmlns="http://www.opengis.net/sos/1.0"
  xmlns:ows="http://www.opengis.net/ows/1.1"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/sos/1.0
  http://schemas.opengis.net/sos/1.0.0/sosGetCapabilities.xsd"
  service="SOS">

  <ows:AcceptVersions>
    <ows:Version>1.0.0</ows:Version>
  </ows:AcceptVersions>

  <ows:Sections>
    <ows:Section>OperationsMetadata</ows:Section>
    <ows:Section>ServiceIdentification</ows:Section>
    <ows:Section>ServiceProvider</ows:Section>
    <ows:Section>Filter_Capabilities</ows:Section>
    <ows:Section>Contents</ows:Section>
  </ows:Sections>

</GetCapabilities>
```

This TestClient was successfully tested in Firefox 3.5.2, Safari 4.0.3, Opera 9.64 and InternetExplorer 8.0.6001.18702 and should work properly in Firefox 1.0 or higher, Safari 1.2 or higher, Opera 8 or higher and InternetExplorer 5 or higher.

Figure 8: Screenshot of the SOS test client 2.0 for capabilities request

- Click the „Send“-Button. The capabilities response xml-document is now shown in your browser.

3.6.2 Other tests

- You can try the other xml request files in the drop down list analogous to the HTTP Post based GetCapabilities request.

4 Insert Data

There are two ways to insert data into the database. You can use the SOSFeeder Framework or you can use the transactional function of the SOS.

Important note for users of PostgreSQL:

Regardless if you are using the Feeder Framework or the Transactional Profile to insert data into the SOS database the ordering of coordinates alsways has to be long/lat, even if you are using a EPSG code for a reference system which defines lat/long as default ordering. The SOS is able to switch coordinates on its own and will return coordinates in the right order.

4.1 SOSFeeder Framework

The SOS Feeder is a framework which can be used to insert data into the standard SOS database (PostgreSQL 8.1 or higher). The **FeederServlet** acts as interface for communication with the SOS DB Feeder.
For more information and how to use:

•You can download the SOSFeeder from the same SVN as the SOS. Only the Repository-Path has to be changed to `/swe/main/SOS/Feeder/SOSFeeder).`

•In the doc-Folder you can find the installation guide.

4.2 SOS Transactional Profile

The transactional profile should enable data producers to register new sensors to the SOS using the RegisterSensor operation and afterwards to insert new observations using the InsertObservation operation. Both operations are very generic due to the reason that the RegisterSensor operation request contains a SensorML description as parameter and the InsertObservation operation request needs a om:Observation as parameter. So currently only numeric values for phenomena could be inserted into the 52°North SOS implementation.

In the Folder xml_mobile you can find examples for RegisterSensor (RegisterSensor_mobile.xml) and InsertObservation (InsertObs_mobile.xml).

5 Troubleshooting

Deploying the web application as described in 3.5.43 fails.

Make sure that:

- you have set the correct path of your Tomcat 6.0 Installation in the pom.xml file.
- Tomcat is running.
- You are in the 52n-sos path and typed "mvn -Pwith-deploy install" in the command line to deploy.

Where do I find the administrator username and password of my Tomcat?

The login datas are stored in the *tomcat-users.xml* in *your Tomcat 6.0/conf* Folder.

If you have any question concerning the installation process, feel free to contact the 52°North SWE mailing list at swe@52north.org.

6 Appendix

6.1 Service Exception codes

Table 2: Exception code listing (Source: Whiteside, A. (2005): OWS Common Implementation Specification, p.34)

ExceptionCode value	Meaning of code	„locator" value
OperationNotSupported	Request is for an operation by this server.	Name of operation not supported
MissingParameterValue	Operation request does not include a parameter value, and this server did not declare a default for that parameter.	Name of missing parameter
InvalidParameterValue	Operation request contains an invalid parameter value [a]	Name of parameter with invalid value
VersionNegotationFailed	List of versions in „Accept Versions" parameter value in GetCapabilities operation request did not include any version supported by this server.	None, omit „locator" parameter

InvalidUpdateSequence	Value of (optional) update Sequence parameter in GetCapabilities operation request is greater than current value of service metadata updateSequence number.	None, omit „locator“ parameter
NoApplicableCode	No other exceptionCode specified by this service and server applies to this exception	None, omit „locator“ parameter
a When an invalid parameter values is received, it seems desirable to place the invalid value(s) in ExceptionText string(s) associated with the InvalidParameterValue value.		

6.2 Known Issues

6.2.1 Maximal size of response document

The Sensor Observation Service was tested at a Server with the following configuration:

- Intel Pentium 4 2.66 Ghz
- 1.5 GB RAM

The O&M response size is limited up to 6.04 MB, if the standard configuration of initial and maximum java memory pool is not changed (tomcat default configuration). This size accords to a number of about 6000 observations in the O&M document. You may have to adjust the tomcat configuration in order to enable larger responses.

6.2.2 Version of schema files

The SOS is an adopted OGC specification. There may occure minor changes in the SOS schemas in future. As those potential changes are not yet known, they could not be addressed in this release. The xml beans libraries are generated from the schema files of version 1.0.0 (state of 2008-02-11).

6.2.3 OM reponse document

The 52N-SOS-3.0.0 always returns an ObservationCollection as response. If the procedure represents a procedure package, in the specification is stated that the response should be a ComplexObservation.

6.3 References

- Botts, M. (2007): Sensor Model Language (SensorML) Implementation Specification, Best Practice Paper, Version 1.0.0, OGC document 07-000
- Priest, M. & Na, A. (2007): Sensor Observation Service, Draft Implementation Specification, Version 0.1.2b, OGC document 06-009r4
- Whiteside, A. (2007): OWS Common Implementation Specification, Implementation Specification, Version 1.1.0, OGC document 06-121r3

- Cox, S. (2007): Observations and Measurements (O&M) Implementation Specification, Version 1.0, OGC document 07-022r1