EU Framework Program for Research and Innovation (SC5-18a-2014 - H2020)

Project Nr: 641538

Coordinating an Observation Network of Networks EnCompassing saTellite and IN-situ to fill the Gaps in European Observations

Deliverable D7.2
Observation networks tutorials and portfolio of best practices

Version 3

Due date of deliverable: 31/11/2016
Actual submission date: 15/12/2016

Document control page				
Title	D7.2 Observation networks tutorials and portfolio of best practices			
Creator	52N			
Editor	52N			
Description	Report on the collection of best practices and tools for observation networks and data exchange.			
Publisher	ConnectinGEO Consortium			
Contributors	ConnectinGEO Partners			
Type	Text			
Format	MS-Word			
Language	EN-GB			
Creation date	31/11/2016			
Version number	0.0.1			
Version date	15/12/2016			
Last modified by				
Rights	Copyright © 2016, ConnectinGEO Consortium			
Dissemination level		CO (confidential, only for members of the consortium)		
	X	PU (public)		
		PP (restricted to other programme participants)		
		RE (restricted to a group specified by the consortium)		
	When restricted, access granted to:			
Nature		R (report)		
		P (prototype)		
		D (demonstrator)		
	X	O (other)		
Review status	X	Draft	*Where applicable:*	
		WP leader accepted		Accepted by the PTB
		PMB quality controlled		Accepted by the PTB as public document
		Coordinator accepted		
Action requested		to be revised by all ConnectinGEO partners		
		for approval of the WP leader		
		for approval of the PMB		
		for approval of the Project Coordinator		
		for approval of the PTB		
Requested deadline				

Revision history

Version	Date	Modified by	Comments
1.0	17-11-2016	MR_52N | SJ_52N | CD_52N	Creation of the content of the web tutorial
2.0	02-12-2016	JM_CREAF | IS_CREAF	Tutorial website reviewed and creation of the first version of the deliverable
3.0	15-12-2016	MR_52N | SJ_52N | CD_52N	Review of the deliverable, executive summary and final version preparation

Table of Contents

0. Executive summary

This document describes a Sensor Web Tutorial and Guide that has been developed within the ConnectinGEO project to strengthen the adoption of interoperable standards such as the Sensor Web technology. It provides an introduction into the relevant standards and the related technology. The tutorial relies on exemplary scenarios to illustrate the different standards and the corresponding approaches how to apply Sensor Web technology. These approaches, which were evaluated as part of ConnectinGEO comprise:

- In-situ data for solar irradiance (in cooperation with Mines Paris Tech)
- Weather data collected by a Sensor Web testbed set-up and maintained by 52°North
- In-situ data collected by mobile sensors as part of the enviroCar platform (e.g. data about fuel consumption of cars)
- Stationary remote sensors (crowd-sources photographs of the FotoQuest platform in cooperation with IIASA)

Within the tutorial especially the following topics are introduced:

- An introduction into the concept of Spatial Data Infrastructures and the Sensor Web
- The OGC Sensor Model Language (SensorML) as a means for providing metadata about measurement processes.
- The ISO/OGC Observations and Measurements (O&M) standard for modelling and encoding observation data.
- Fundamental OGC service interfaces for accessing observation data (OGC Sensor Observation Service) and controlling sensors (OGC Sensor Planning Service)
- Practical guidance on how to design Sensor Web systems including a hand-on part based on a dedicated Docker image of the 52°North SOS.
- A short introduction into typical SOS client applications.
- Information about the relationship between Sensor Web and Internet of Things technologies.

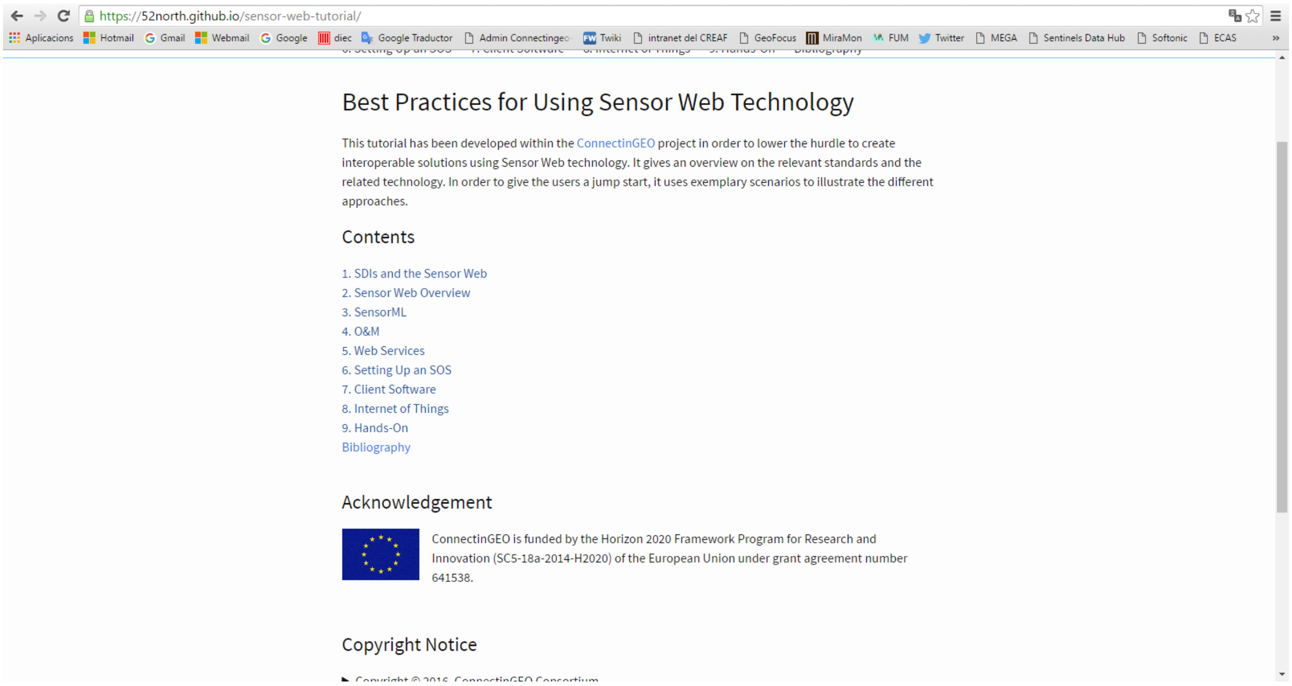
In summary, the presented tutorial offers a comprehensive guide for readers that are interested to share their observation data in an open and interoperable manner.

2. Introduction

This deliverable is based on the tutorial that has been developed within the ConnectinGEO project in order to lower the hurdle to create interoperable solutions using Sensor Web technology https://52north.github.io/sensor-web-tutorial/01_sdis-and-swe.html. It gives an overview on the relevant standards and the related technology. In order to give the users a jump-start, it uses exemplary scenarios to illustrate the different approaches.

The OGC Sensor Web Enablement (SWE) Framework offers open standards for sensor data encoding and exchange as well as several Web services to homogeneously access sensor data from heterogeneous sensors. Within this Best Practice Guide, the central components of the SWE framework are introduced to provide basic knowledge on how to use them.

D7.2 provides an overview of the tutorial based on texts and screenshots from the tutorial website.



3. Standards overview

This section of the tutorial guides the user through several standards that help to overcome issues of data heterogeneity, particularly when dealing with in-situ sensors.

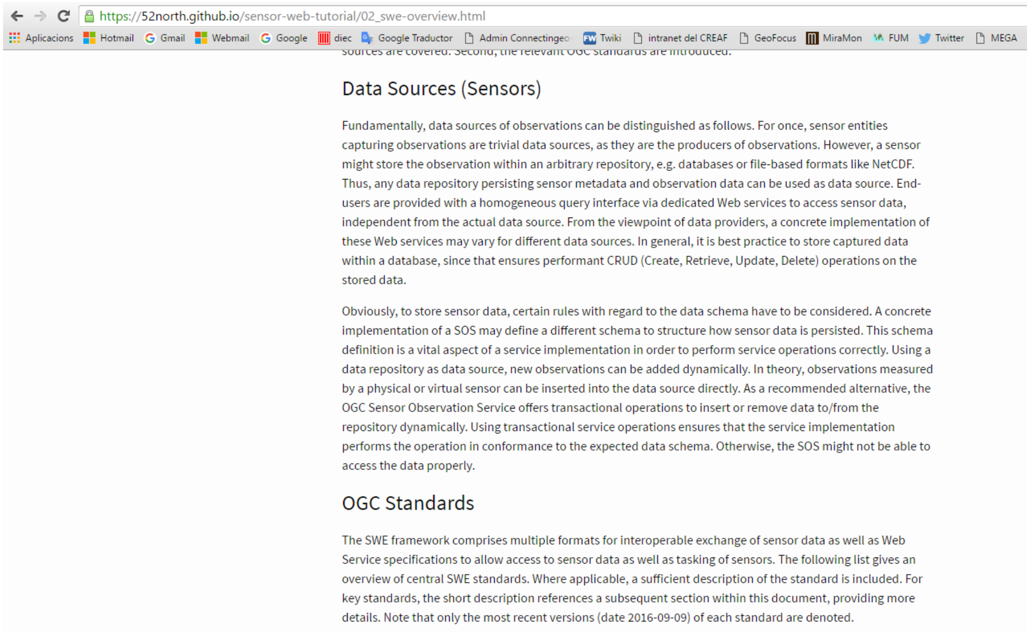
3.1. SDIs and the Sensor Web

A Spatial Data Infrastructure denotes a collection of technologies, policies and institutional arrangements to serve as a basis for discovery, access and processing of geospatial data (Global Spatial Data Infrastructure Association, 2012). It comprises standards, specifications and agreements to ensure interoperability and better access to spatial data. Some examples are the EU Initiative INSPIRE (Infrastructure for Spatial Information in the European Community) or GEOSS (Global Earth Observation System of Systems).

SWE is an acronym for Sensor Web Enablement and comprises standardized formats and Web service interfaces in the fields of sensor data (Bröring et al, 2011 a). Since 2003, the standardization organisation the Open Geospatial Consortium (OGC) follows the vision to make heterogeneous sensor data (such as remote satellite / airborne imaging data, in-situ monitoring sensors or even virtual computations) available for discovery, access and use via interoperable formats and Web services.

3.2. Sensor Web Overview

Within this section, several base components of the SWE framework are denoted and shortly described.



https://52north.github.io/sensor-web-tutorial/02_swe-overview.html

Aplicacions Hotmail Gmail Webmail Google diec Google Traductor Admin Connectingeo Twiki intranet del CREAF GeoFocus MiraMon FUM Twitter MEGA

sources are covered. Second, the relevant OGC standards are introduced.

Data Sources (Sensors)

Fundamentally, data sources of observations can be distinguished as follows. For once, sensor entities capturing observations are trivial data sources, as they are the producers of observations. However, a sensor might store the observation within an arbitrary repository, e.g. databases or file-based formats like NetCDF. Thus, any data repository persisting sensor metadata and observation data can be used as data source. End-users are provided with a homogeneous query interface via dedicated Web services to access sensor data, independent from the actual data source. From the viewpoint of data providers, a concrete implementation of these Web services may vary for different data sources. In general, it is best practice to store captured data within a database, since that ensures performant CRUD (Create, Retrieve, Update, Delete) operations on the stored data.

Obviously, to store sensor data, certain rules with regard to the data schema have to be considered. A concrete implementation of a SOS may define a different schema to structure how sensor data is persisted. This schema definition is a vital aspect of a service implementation in order to perform service operations correctly. Using a data repository as data source, new observations can be added dynamically. In theory, observations measured by a physical or virtual sensor can be inserted into the data source directly. As a recommended alternative, the OGC Sensor Observation Service offers transactional operations to insert or remove data to/from the repository dynamically. Using transactional service operations ensures that the service implementation performs the operation in conformance to the expected data schema. Otherwise, the SOS might not be able to access the data properly.

OGC Standards

The SWE framework comprises multiple formats for interoperable exchange of sensor data as well as Web Service specifications to allow access to sensor data as well as tasking of sensors. The following list gives an overview of central SWE standards. Where applicable, a sufficient description of the standard is included. For key standards, the short description references a subsequent section within this document, providing more details. Note that only the most recent versions (date 2016-09-09) of each standard are denoted.

https://52north.github.io/sensor-web-tutorial/02_swe-overview.html

Standardized **formats**:

- **Observations & Measurements (O&M)**: The O&M standard defines how an Observation is encoded and structured, as elaborated in Section 6.2.
- **PUCK Protocol Standard**: PUCK specifies a protocol for RS232 and Ethernet connected sensor instruments. Via dedicated commands, relevant sensor metadata (e.g. sensor description encoded in SensorML) can be retrieved from the instrument directly. In addition, offered configurations and settings can be remotely applied to an instrument using dedicated PUCK operations. The official specification is available under http://www.opengeospatial.org/standards/puck.
- **Sensor Model Language (SensorML)**: SensorML specifies a homogeneous description of procedure metadata, describing the entity that measures an observable property including details on how it is measured. In consequence, for each different procedure type (e.g. physical sensors or virtual computations), the same set of descriptive elements can be applied to provide an interoperable sensor description. Section 6.1 presents an exemplar overview of how to encode sensor metadata using SensorML.

Standardized **Web Services**:

- **Sensor Observation Service (SOS)**: The SOS is the central Web service within the SWE framework. It allows the retrieval of various sensor data including Sensor descriptions as SensorML, observation data encoded in O&M or specific data like features or certain results. Moreover, certain transactional operations are specified to add/remove new sensors as well as observations. An extensive overview of the available operations and relevant other service aspects are given in the dedicated section.
- **Sensor Planning Service (SPS)**: Via dedicated operations offered by a SPS, the sensors themselves can be tasked to perform certain measurements, as elaborated in the dedicated section.
- **SensorThings API**: As elaborated in section 10, the SensorThings API is targeted to allow homogeneous access to heterogeneous smart sensors within the Web of Things approach. The standard is located at http://docs.opengeospatial.org/is/15-078r6/15-078r6.html.

3.3. SensorML

Sensors in general can be classified into the following categories: stationary, mobile, in-situ and remote. Sensors can be installed at a fixed location (stationary) or be mounted on a mobile device to operate in arbitrary locations. In-situ and remote indicate, whether the sensor performs the measurement using local input from within its direct environment (in-situ) or computes measurements for non-local data (remote). In addition, the process, how sensors collect and/or compute data, may vary dramatically from use case to use case. In consequence, a common generic way to describe sensor instances is necessary. To satisfy these various requirements, the OGC developed the Sensor Model Language (SensorML) standard, which can be retrieved as version 2.0 from the URL:

http://www.opengeospatial.org/standards/sensorml

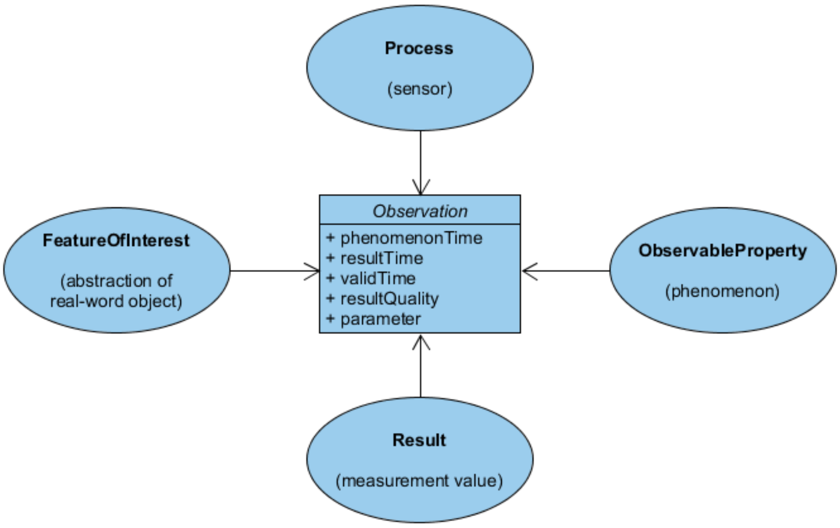
Within the SWE framework, SensorML plays an important role when exchanging metadata about sensors themselves. It represents a uniform description format to describe relevant sensor metadata such as calibration parameters, input/output as well as location definitions, pre- or post-measurement actions/transformations or any other information, which is necessary to correctly interpret and process observations (OGC 2011). Within this scope, a sensor is denoted as a process, which may be a physical or virtual component transforming dedicated input data to one or more outputs via a certain transformation/method. For each degree of complexity, SensorML provides means to describe the whole process metadata in an interoperable way.

More information at: https://52north.github.io/sensor-web-tutorial/03_sensorml.html

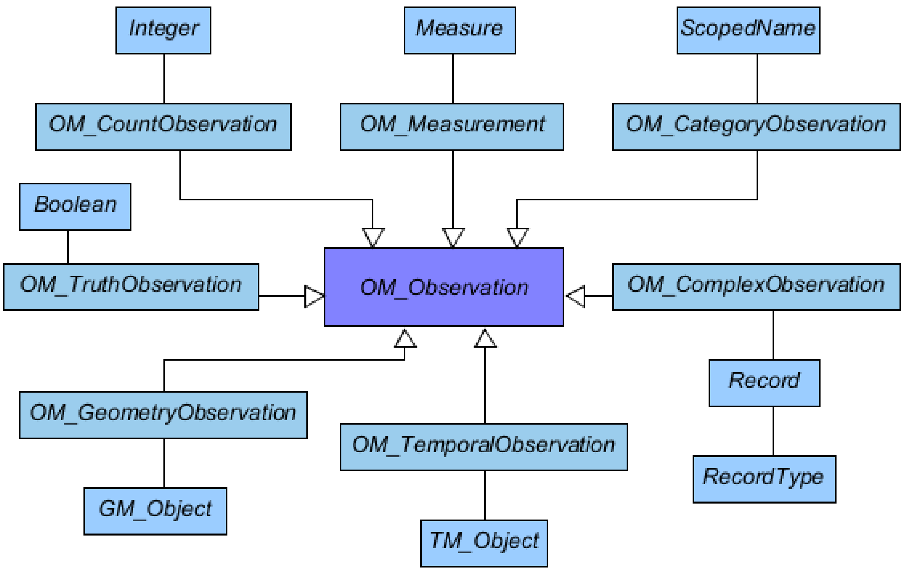
3.4. Observations & Measurements

An observation is a composed object providing relevant observation details. In consequence, an observation belongs to a procedure that measured a certain observedProperty at a dedicated featureOfInterest for a certain phenomenonTime and stores a result. The property resultQuality indicates the quality of the measured result, as each measurement is faulty to a certain extent by nature.

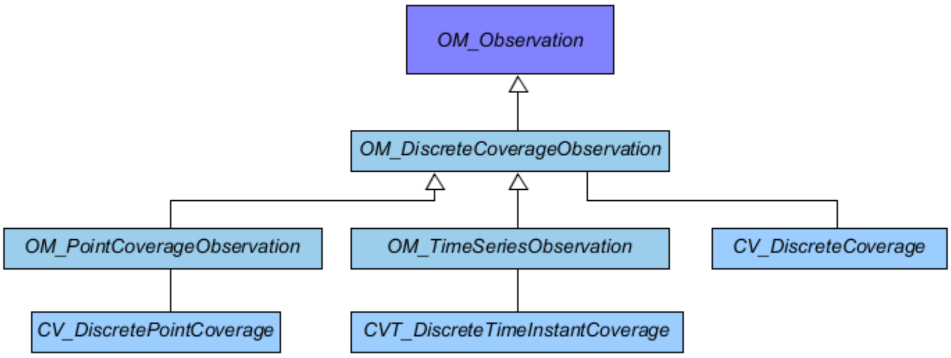
The following figure presents an overview of the components of an observation:



Based on this generic concept, observations are categorized according to their result type. They may be distinguished in discrete and continuous observation. An observation is discrete, when the result represents a spatio-temporal invariant value and applies to the whole area of the referenced featureOfInterest. The possible observation types and their content/result types are illustrated in the below figure. In contrast, a continuous observation measures an observableProperty that may vary in time or for the whole area of a certain featureOfInterest. In that case, the result is not represented by a single value, but by a coverage. To model the coverage, distinct sub-geometries are defined to discretise the whole area of the associated feature. In consequence, a single coverage consists of multiple entries, one per sub-geometry and/or timestamp.



The following figure represents the standardized coverage observation types according to the O&M standard (Cox 2013) in a simplified way. Considering the discretization of an area-based featureOfInterest, the standard introduces two distinct declarations. The whole featureOfInterest as a single object (a spatially distributed entity) is denoted as ultimate feature, whereas each sub-geometry resulting from the discretization process is denoted as a sampling feature. Hence, each concrete measurement value of a coverage observation refers to a sampling feature.



Some examples are given in https://52north.github.io/sensor-web-tutorial/04_o-and-m.html

4. Web Services

4.1. Introduction

The OGC Sensor Observation Service (SOS) is a Web Service that allows users to request observations and associated metadata of sensors. Within the context of the SWE framework, the SOS represents the core service to access sensor data in an interoperable and standardized manner. The service specification defines several mandatory as well as some optional operations. With regard to the core operations (GetCapabilities, DescribeSensor, GetObservation), a description, the service interface (request parameters) and an example is presented. For most optional operations, only a description of the method as well as the request parameters is included. Most of the subsequent information was extracted from the SOS 2.0 standard specification (OGC 2012).

https://52north.github.io/sensor-web-tutorial/05_web-services.html

Parameter Name	Description	Mandatory
service	fixed value "SOS"	no
request	fixed value "GetCapabilities"	yes
version	indicates the service version, e.g. "2.0.0"	yes
extension	specific extension, e.g. "language"	no
acceptVersions	submit accepted versions, e.g. "2.0.0"	no
acceptFormats	preferred response formats	no
updateSequence	service metadata document version, value is "increased" whenever any change is made in complete service metadata document	no
sections	include only relevant sections within the response document and omit the rest	no

A full response document contains several sections offering metadata. The most relevant sections are:

- **serviceIdentification**: metadata about the service itself, such as title, versions, languages
- **serviceProvider**: metadata about the provider/organization, such as contact information
- **operationsMetadata**: metadata about the offered operations, including which operations are offered (like GetCapabilites, DescribeSensor, GetObservation, …), available bindings (e.g. POX, KVP, SOAP, JSON) and a URL endpoint.
- **contents**: metadata about available observation offerings including their associated properties such as identifier, procedure, reponseFormats, temporal and spatial aspects, …
- **filterCapabilities**: metadata about the supported spatial and temporal filter functionalities extension: container for extensions

https://52north.github.io/sensor-web-tutorial/05_web-services.html

DescribeSensor

The `DescribeSensor` operation can be used to retrieve a detailed sensor description about a certain sensor, encoded in a Sensor Model Language (SensorML) version 1.0.1 or 2.0 document. As sensor configuration and calibration might change over time, several different instances of a sensor description for the same sensor can be stored by a SOS, each being valid for a certain amount of time. To request a certain sensor description the following request parameters are offered:

Parameter Name	Description	Mandatory
service	fixed value "SOS"	no
request	fixed value "DescribeSensor"	yes
version	indicates the service version, e.g. "2.0.0"	yes
extension	specific extension, e.g. "language"	no
procedure	reference to the target sensor/procedure	yes
procedureDescriptionFormat	selects the target description format identifier	yes
validTime	Time instant or period for which the sensor description is retrieved	no

For instance, an exemplar DescribeSensor request (using the binding POX) against the URL http://insitu.webservice-energy.org/52n-sos-webapp/service might look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<swes:DescribeSensor xmlns:swes="http://www.opengis.net/swes/2.0" xmlns:gml="http://www.
   <!-- reference to sensor/procedure using its identifier -->
   <swes:procedure>ENTPE-BH-KZ-CH1-QCfull</swes:procedure>
   <!-- definition of sensor description format; here SensorML 1.0.1 -->
   <swes:procedureDescriptionFormat>http://www.opengis.net/sensorML/1.0.1</swes:procedur
   <!-- optional validTime: if not set, latest sensor description is returned -->
```

4.2. Setting-Up an SOS Server

Within this section, concrete recommendations on how to setup a SOS server are given. To offer sensor data in a standardized and interoperable way, an implementation of a SOS according to the OGC specification has to be deployed on a Web server. The implementation itself hereby has to consider several key aspects. First of all, an appropriate data source tier is required in order to persist, manage and retrieve sensor data. Second, a suitable implementation of the core SOS functionalities must be provided.

https://52north.github.io/sensor-web-tutorial/06_sos-setup.html

Data Sources and Storing of Sensor Data

While in theory, an arbitrary data source could be used for persistence and data management, a general recommendation is to employ a database (e.g. PostgreSQL or Oracle). Advantages of a database include at least the following:

- **generic data model** for arbitrary sensor data: In contrast to other data sources like file based sources, a database model, realized through multiple related tables, represents a generic data schema with which heterogeneous sensor data can be stored in a homogeneous way.
- **improved access**: a Data Base Management System (DBMS) provides a performant and quality assured access point for data modification/retrieval (management). In opposite to other sources, this enables efficient CRUD operations, which reduces response time of SOS-requests.
- support for different concrete DBMS using similar SQL statements (even better: using a high-level database management library like Hibernate allows support for several concrete DBMS)

In conclusion, a database as data source enables efficient and robust management of sensor data. In general, there are different setting scenarios on how to employ a SOS implementation on top of a database. For once, it has to be decided whether to use an already existing database with its own data schema or create a new database conforming to the specific requirements of the used SOS instance. Another aspect is whether the SOS instance shall directly access the database or use an intermediate proxy component to handle database access. The subsequent sub sections elaborate on each distinct setting scenario and provide helpful hints about advantages and disadvantages.

https://52north.github.io/sensor-web-tutorial/06_sos-setup.html

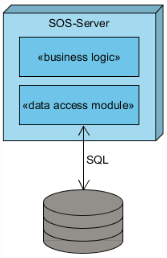
Aplicacions Hotmail Gmail Webmail Google diec Google Traductor Admin Connectingeo Twiki intranet del CREAF GeoFocus MiraMon FUM Twitter MEGA

Usage of existing Databases for direct Access

When setting up a SOS, existing databases can be used. However, considering the data schema (how the relevant information is structured within the database), each database might reveal differences compared to each other. Hence, it is a decisive prerequisite to adapt the SOS server to correctly identify and interpret the relevant data stored in a certain database. For this reason, multiple strategies exist, comprising an adaption of the SOS server (see following sections) or the creation of database views (see below as well) to prepare the database information in a SOS-friendly way.

Adaption of the Database Access Module of the SOS Implementation

This approach requires a programmatic adaption of the SOS data access module. The implementation logic is changed to reflect the data schema of the database. However, this is a very specialized approach only applicable for implemented schema and is hence not recommended.

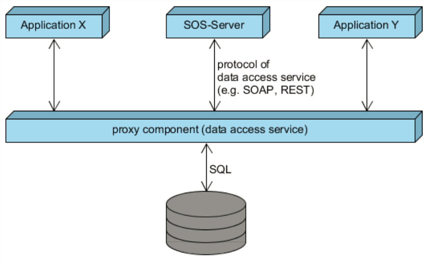


https://52north.github.io/sensor-web-tutorial/06_sos-setup.html

Aplicacions Hotmail Gmail Webmail Google diec Google Traductor Admin Connectingeo Twiki intranet del CREAF GeoFocus MiraMon FUM Twitter MEGA

Using an Intermediate Proxy

In the previous scenarios, the database was directly accessed by the SOS server. As an alternative, an intermediate proxy component can be employed to encapsulate the database access. Due to this architecture, the SOS server is made independent from the actual database model/schema, as the interface between SOS server and proxy remains stable. Should the data model/schema change, then only the proxy component has to be adapted accordingly. However, the additional proxy component creates additional communication, which might lead to performance issues. For instance, the proxy might offer several de- and encodings of the sensor data (e.g. SOAP or REST), which might require an additional de- or encoding step in the processing chain.



In order to reduce communication with the database, a SOS server may cache frequently queried data. For instance, the GetCapabilities operation (see corresponding section) provides a general overview of available sensor data. If the corresponding metadata was cached by the SOS server, it would not have to retrieve that data from the database for each request. Instead it would rely on the cached information to answer the request. Of course, the cached information should be updated regularly to reflect the most current state of stored information. Concluding, a caching approach allows better performance with regard to response time of SOS operations, but it has to be ensured that the cached information is up-to-date.

4.3. Sensor Web Client Software

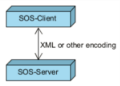
Another important aspect of SWE is to visualize and analyse observation data. In this context relevant data is often denoted as time series data to indicate the temporal variety of observed phenomena, which shall be perceived using applicable graphs and diagrams. In addition, certain data might be visualized on an interactive map enabling users to click on those objects and trigger certain tasks. Within this section, information about sensor web clients are provided, starting with an overview of different architectural concepts.

https://52north.github.io/sensor-web-tutorial/07_client-software.html

Architectural Concepts

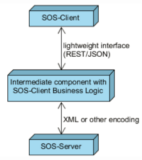
Direct Interaction with SOS server

As an intuitive setting, SOS client and server could communicate directly. Clients would use the dedicated operations of the SOS to retrieve relevant sensor data. This implies that any logic for interactions as well as de- and encoding of SOS-requests and corresponding responses have to be executed by clients themselves. Hence this setting is only recommended when the SOS client is running on high-performance hardware.



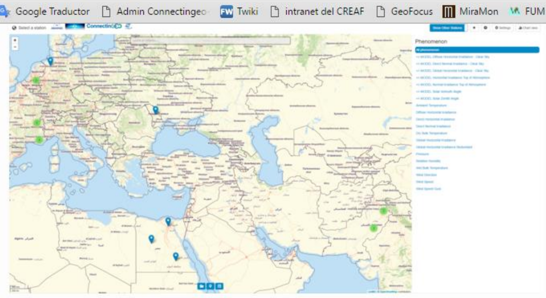
Intermediate Component Performing Business Logic

In contrast to direct communication, an additional intermediate component can be employed to perform arbitrary intermediate business logic tasks. This intermediate component runs on high-performance hardware and encapsulates the communication with the SOS server. In addition, it may offer other functionalities, such as storing cached sensor metadata, perform generalization tasks (see below) or offer a lightweight interface (e.g. REST/JSON) to clients. Following this architecture pattern includes the advantage that complex business tasks such as decoding large XML datasets or caching of metadata is shifted from clients to a server-side component. However, the client side is not allowed to interactively add another SOS server and visualize its data as this knowledge has to be added to the intermediate component itself.

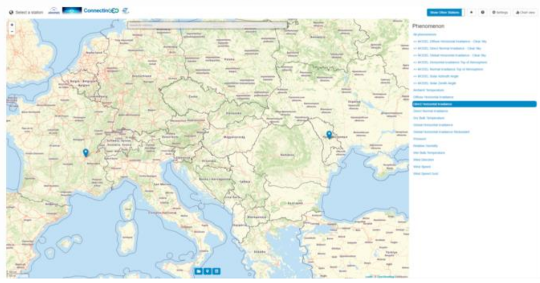


As request/response messages have to be frequently exchanged between a SOS client and a SOS, a REST interface can be facilitated for lightweight message exchange and reduction of message contents. With the help of a dedicated REST API, a SOS server (or intermediate proxy component) may provide an easy access to relevant observation data through a RESTful interface, where message exchange is realized by a JSON binding. Via dedicated REST API calls, only those elements, which are really necessary for client side visualization, can be retrieved. The benefit is that the content of exchanged messages can be minimized and, thus, message and network/bandwidth overload be reduced. This enables clients with an on-demand query interface for time series data.

https://52north.github.io/sensor-web-tutorial/07_client-software.html



On startup, the relevant data (stations as well as phenomena) is automatically retrieved from the configured SOS instance (http://insitu.webservice-energy.org/52n-sos-webapp). Via the phenomena list, stations can be filtered. E.g. by clicking on the phenomena *Direct Horizontal Irradiance*, the associated stations are determined and the map view is refreshed showing only the remaining stations.



5. Sensor Web and the Internet of Things

The keywords Internet of Things and Web of Things denote current approaches to bring smart devices into the Web. In the context of the Sensor Web, such smart devices can be sensors that measure or compute certain phenomena and bring the results into the Web. From a technological perspective, this means that the smart device (e.g. a sensor) is accessible via a unique URL identifier and its functionalities (e.g. observation results) addressable via standard HTTP operations (GET, POST, PUT, etc.), thus realizing a RESTful access to sensor data (Bröring et al, 2011 a). This section intends to give an overview of how the Web of Things approach can be linked to the SWE framework/infrastructure. According to Bröring et al. (2011 b), a smart device may respond to data requests with formats standardized by SWE. To be precise, metadata about the sensor itself should be encoded via SensorML and observation data as O&M. With this format restrictions, any other application/component may retrieve the sensor data of the smart device in an interoperable standardized manner.

However, considering the access to sensor data, SWE's goal is to standardize the query interface as well as other services (like tasking, subscribing) through dedicated Web Services (SOS, SPS, SES). As a smart device, each individual sensor may offer RESTful access to its sensor data, but this service interface may not conform to the service interface specified/standardized by SWE. For this reason, the OGC specified the SensorThings API to access data from smart sensors in an interoperable way (Liang et al, 2016). The SensorThings API is part of SWE and bases on the data model of O&M. Basically, it can be considered as a lightweight SWE profile designed specifically for resource-restricted smart devices. The keyword lightweight indicates that in contrast to other SWE services like SOS or SPS, the SensorThings API may simplify the connections between devices-to-devices and devices-to-applications through usage of efficient REST binding including a reduced JSON encoding and the MQTT protocol for sensor access. Liang et al. stress that the classical SWE services are too heavyweight for combination with smart sensors. But they also highlight, that the complexity of SOS, SPS and SES are well justified for more complex use case scenarios, where the Web of Things approach is not applicable. Overall, the following table compares the SensorThings API to the SOS.

https://52north.github.io/sensor-web-tutorial/08_iot.html

Aplicacions Hotmail Gmail Webmail Google diec Google Traductor Admin Connectingeo Twiki intranet del CREAF GeoFocus MiraMon FUM Twitter

combination with smart sensors. But they also highlight, that the complexity of SOS, SPS and SES are well justified for more complex use case scenarios, where the Web of Things approach is not applicable. Overall, the following table compares the SensorThings API to the SOS.

	SensorThings API	Sensor Observation Service
Encoding	JSON	XML
Architecture Style	Resource Oriented Architecture	Service Oriented Architecture
Binding	REST	SOAP
Inserting new sensors or observations	HTTP POST (e.g., CRUD)	using SOS specific interfaces, e.g. InsertSensor(), InsertObservation()
Deleting existing sensors	HTTP DELETE	using SOS specific interfaces, i.e. DeleteSensor()
Pagination	$top, $skip, $nextLink	Not Supported
Pub/Sub Support	MQTT and SensorThings MQTT Extension	Not Supported
Updating properties of existing sensors or observations	HTTP PATCH and JSON PATCH	Not Supported
Deleting observations	HTTP DELETE	Not Supported
Linked data support	JSON-LD	Not Supported
Return properties subset	$select	Not Supported
Request multiple O&M entities (e.g., FeatureOfInterest and Observation) in one request/response	$expand	Not Supported

Overall, the SensorThings API may be facilitated to connect to heterogeneous single smart sensors using a homogeneous standardized RESTful interface. However, this Web of Things approach is only applicable for simple and concrete use cases. Considering more complex scenarios like disaster management, where sensor data from various sensors is required for a composed analyzation, the Web of Things approach may not be applicable (Bröring et al, 2011 b).

In conclusion, SWE and the Web of Things show potential to be combined. Smart sensors can encode their data using standardized SWE formats (Bröring et al., 2011 b) and their data can be offered through the standardized SWE SensorThings API (Liang et al., 2016). Especially for simple use cases, where the full functionality on data filtering, sensor discovery, tasking and event handling, as provided by classical SWE services (SOS, SPS, SES), are not required, the Web of Things approach can be facilitated to provide access to the sensor data of the smart sensor. However, with regard to complex scenarios, the Web of things approach might not be applicable, as highlighted by Bröring et al. (2011 a) and Liang et al. (2016).

6. References

- 52°North (2016): Sensor Discovery. http://52north.org/communities/sensorweb/discovery/index.html.

- 52°North (2016 b): Sensor Event Service (SES). http://52north.org/communities/sensorweb/ses/0.0.1/index.html.

- 52°North (2016 c): 52°North Sensor Observation Service 4.x Database Model. https://wiki.52north.org/SensorWeb/SensorObservationServiceDatabaseModel.

- 52°North (2016 d): SOS 4.x Documentation. https://wiki.52north.org/SensorWeb/SensorObservationServiceIVDocumentation.

- Botts, Mike; Robin, Alexandre (2014): OGC® SensorML: Model and XML Encoding Standard. https://portal.opengeospatial.org/files/?artifact_id=55939.

- Bröring, Arne; Echterhoff, Johannes; Jirka, Simon; Simonis Ingo; Everding, Thomas; Stasch, Christoph; Liang, Steve; Lemmens, Rob (2011 a): New Generation Sensor Web Enablement. http://www.mdpi.com/1424-8220/11/3/2652.

- Bröring, Arne; Remke, Albert; Lasnia, Damian (2011 b): SenseBox – A Generic Sensor Platform for the Web of Things. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Volume 104, pp. 186-196.

- Cox, Simon (2013): OGC Abstract Specification - Geographic information — Observations and measurements. http://portal.opengeospatial.org/files/?artifact_id=41579.

- enviroCar (2015): enviroCar – off we go!. https://envirocar.org/.

- GEO Group on Earth Observations (2016): GEOSS – ACCESS CONNECTING USERS. http://www.earthobservations.org/geoss.php.

- Global Spatial Data Infrastructure Association (2012): Spatial Data Infrastructure Cookbook. http://gsdiassociation.org/images/publications/cookbooks/SDI_Cookbook_from_Wiki_2012_update.pdf.

- Liang, Steve; Huang, Chih-Yuan; Khalafbeigi, Tania (2016): “OGC SensorThings API Part I:Sensing” OGC® Implementation Standard. http://docs.opengeospatial.org/is/15-078r6/15-078r6.html.

- Ménard Lionel (n. y.): ConnectinGEO Energy – In-situ Measurements – Unleash Surface Solar Irradiance (SSI) Observation – A challenge. http://www.connectingeo.net/Docs/ConnectinGEO_Energy_Mexico_Booth-V2.pdf.

- Ménard, Lionel; Nüst, Daniel; Jirka, Simon; Masó, Joan; Ranchin, Thierry; Wald, Lucien (2015): Open Surface Solar Irradiance Observations - A Challenge.

Geophysical Research Abstracts Vol. 17, EGU2015-6607, 2015: http://meetingorganizer.copernicus.org/EGU2015/EGU2015-6607.pdf. European Geosciences Union General Assembly 2015 - EGU 2015, Vienna (Austria)

- Nebert, Douglas; Whiteside, Arliss; Vretanos, Panagiotis (2007): OGC® Implementation Specification 07-006r1: OpenGIS Catalogue Services Specification. http://portal.opengeospatial.org/files/?artifact_id=20555.
- NeXOS Project (2016): http://www.nexosproject.eu/.
- OGC (2011): Introduction to SensorML. http://www.ogcnetwork.net/SensorML_Intro.
- OGC (2012): OGC® Sensor Observation Service Interface Standard. https://portal.opengeospatial.org/files/?artifact_id=47599.
- OGC (n. y.): Sensor Web Enablement (SWE). http://www.opengeospatial.org/ogc/markets-technologies/swe.
- Redhat (2016): Hibernate. Everything data. http://hibernate.org/.
- Simonis, Ingo; Echterhoff, Johannes (2011): OGC® Sensor Planning Service Implementation Standard. http://portal.opengeospatial.org/files/?artifact_id=38478.