EMODnet Ingestion and safe-keeping of marine data n.2

EASME/EMFF/2018/1.3.1.8/01/SI2.810021

EMODnet Data Ingestion

D3.3 - SWE Service Installation User Guide

September 2020

Contents

Disclaimer

Document info

Title	EMODnet Data Ingestion – D3.3 - SWE Service Installation User Guide
Authors [affiliation]	Antonio Novellino, Luca Bonofiglio, Andrea Bocchimpani [ETT], Christian Autermann – 52North
Dissemination level	Public
Revision	1
Date	16/09/2020

1. Introduction

The European Marine Observation and Data Network, EMODnet, is a long-term marine initiative implementing mechanism of the European Commission's Marine Knowledge 2020 strategy1,2 to unlock the potential of Europe's wealth of marine data. Based on the principle of collecting data once and using it many times for many purposes, EMODnet is a network of organizations supported by the EU's Integrated Maritime Policy linked by a data management structure. These organizations work together to facilitate the discovery and access to marine data and data products representing the following seven main themes: bathymetry, biology, chemistry, geology, human activities, physics, and seabed habitats; six regional check points and a Data Ingestion facility. EMODnet provides a gateway to those marine data accompanied by their metadata and data products through a number of thematic portals and a central portal (www.emodnet.eu).

The EMODnet Data Ingestion portal seeks to identify and to reach out to other potential providers in order to make their data sets also part of the total offer. It aims at streamlining the data ingestion process so that data holders from public and private sectors that are not yet connected to the existing marine data management infrastructures can easily release their data for safekeeping and subsequent distribution through EMODnet. This will enrich the total offer for all types of users and conform to the EMODnet motto 'collect data once and use it many times'.

The EMODnet Real time Portal (http://www.emodnet-physics.eu/realtime) is a web application that is able to provide NRT data and metadata from marine data centres that offer a machine to machine interface based on the Sensor Observation Service (SOS) standard of the Open Geospatial Consortium (OGC). Its goal is to offer a simple point of access to distributed NRT data in a transparent way: users can add and/or remove available sensor systems to/from the portal and thus access their data.

This document is an introductory guide to users on how to exchange EMODnet Real Time Data using the Sensor Web Enablement Sensor Observation Service. In particular, this document presents how to set up a SWE SOS 52N server and how to connect it to the EMODnet RT data flow.

[1] European Commission (2010). Marine Knowledge 2020 Marine Data and Observation for Smart and Sustainable Growth. Commission Communication COM (2010) 461, Publications Office of the European Union.

[2] European Commission (2012). Marine Knowledge 2020 from Seabed Mapping to Ocean Forecasting. Green Paper, Publications Office of the European Union, Luxembourg.

2. Sensor Web Enablement / Sensor Observation Service

a. A brief intro of the Sensor Web Enablement (SWE)

The Sensor Web Enablement framework developed by the Open Geospatial Consortium (OGC) aims to develop and maintain standards for the interoperable integration of sensors and their observation data into Web-based (spatial) data infrastructures (Bröring et al., 2011). There exist several document types within the OGC, representing the maturity of a specification (e.g. discussion paper, best practice paper or standard).

A specification can be understood as a technical definition for a web service or data model (independent of the grade of maturity) while a standard is the document that has been officially adopted by the OGC.

The OGC Sensor Observation Service (SOS) interface allows pull-based access to observation data as well as sensor metadata. This means that the SOS acts as a mediator between clients and a measurement archive (e.g. database) or sensor system.

Through the SOS, it is possible for clients to query observation data of heterogeneous sources via a standardized interface.

On the one hand the SOS standard defines a set of operations and their parameters and on the other hand it relies of the data model/encoding standards of the SWE framework to provide standardised outputs.

The core operations of the SOS interface are:

- GetCapabilities: Retrieve metadata about a SOS server (e.g. supported operations and available data sets)
- DescribeSensor: Access metadata about the sensors or processes which have generated the observation data offered by the SOS server
- GetObservation: Retrieval of observation data/measurements

An important extension of the SOS interface is a group of transactional operations (InsertSensor and InsertObservation) for publishing new sensors and observations data on a SOS server.

Another important operation is the GetFeatureOfInterest operation which allows the retrieval of the geometric features to which observations are associated. It provides the required spatial context, by serving e.g. point or polygon features of the feature that is being observed.

Figure 1 illustrates the four interface methods and their corresponding response formats.

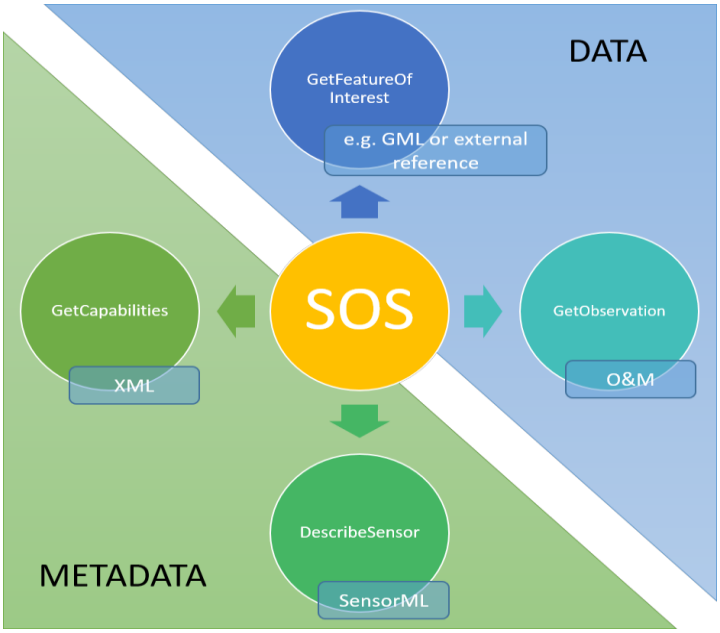

Figure 1. SOS Interfece Method Overview

Increasingly, marine data will be collected by smart sensors and platforms. Several developments are ongoing in this field of developing new sensors for an expanding range of parameters, and new platforms that can carry a payload of multiple sensors and operate efficiently for long durations. These developments need to be anticipated, also with respect to data management and data flow.

For that purpose, adoption of Sensor Web Enablement (SWE) standards holds great promise as it facilitates to streamline data from platforms in real-time to receivers, and to document many relevant aspects of the sensors, platforms, and observations using marine SWE profiles and vocabularies, thus enriching the available metadata from observations at their origin, which will contribute to improving the FAIRness of data sets and documenting the provenance of observed data.

The SeaDataNet consortium, has also made great progress in building upon the SWE standards to support the interoperable sharing of near-real time and real-time observation data streams. This methodology has already tested to streameline data flow into EMODnet Physics and EMODnet Data Intestion is going to uptake and exted futher the system.

This comprises especially a component, which have been developed led by 52North, the 52N SOS Service. The 52°North Sensor Observation Service 4.x3 implements the OGC SOS standard versions 1.0.0 and 2.0. The implementation comprises all extensions defined in the specification.

[3] www.52north.org/sos

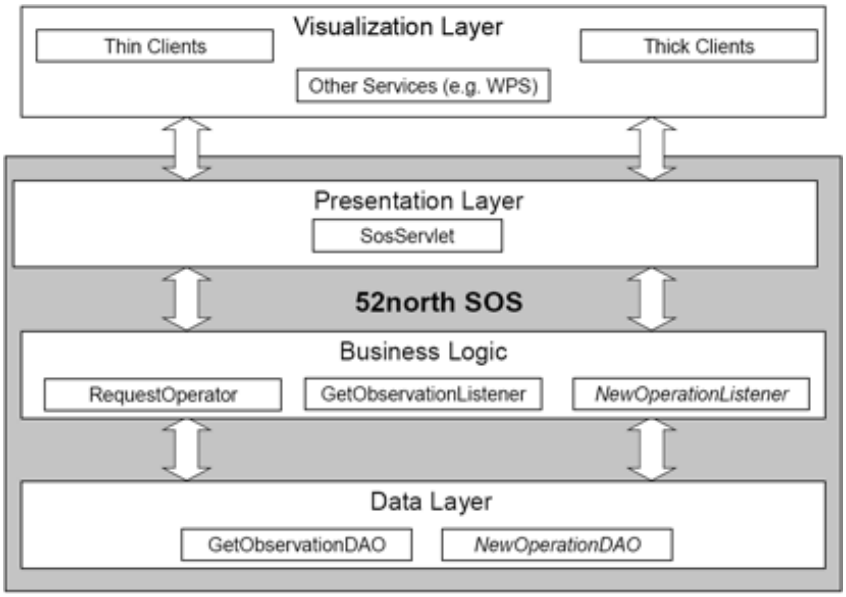

Figure 2. Layered architecture of the 52°North Sensor Observation Service

With its layered architecture, shown in Figure 2 the SOS server can be flexibly connected to different data sources ranging from file-based approaches to different database systems. By default, PostGIS is used as the Database Management System (DBMS). By customizing the Business Logic layer, new functionality may be added. Thereby, new encoders and decoders can be added in a plug and play fashion. For example, prototypical support for an EXI encoding for SOS messages, O&M as well as SensorML has already been implemented in the European research project NeXOS[4].

A new extension of the 52°North SOS implementation is the support of the OGC SensorThings API Sensing profile. Currently a first beta version of the implementation of the core profile of this specification is available. Further extensions of this module are currently work in progress. The aim of this component is to support sensor operators, researchers and data owners to ingest data and SWE metadata from operational observing platforms and sensors into a local storage system and to publish (selected) data streams from this database by means of SOS services to receiving servers. This facilitates operators to publish streams of near-real time and real-time observation data via SOS servers by first describing the structure of the observation network and data stream and then enabling an automated data ingestion, storage, and publication process. The 52N suite also comprises a SWE Viewing Services, based on the Helgoland Sensor Web Viewer, that is an application for exploring and visualising data streams from operational sensors and platforms. This tool is also available to partner that are joining the data sharing methodology as a complementary tool to self-check the correctness of system configuration and provide the users with a further data exploring and access tool.

These components are available as open source software via GitHub[5].

[4] http://www.nexosproject.eu/

[5] More information about these solutions, background, SWE profiles, how to apply, and GitHub locations, can be found at https://www.seadatanet.org/Software/Sensor-Web-Viewer/Documentation

3. SOS Server Service installation

a. General Requirements

This section summarizes the general necessary requirements to connect to 52N SOS server.

Follow these steps:

1) In order to access your services, it is necessary to connect with the Windows VPN to DLTM, using the following credentials:

   - User: xxxx

   - Password: yyyy

   - Shared key: dltmdell

2) Once you are connected to the VPN, create an SSH connection to your server.

To implement this operation, it is recommended to download a dedicated software, such as Putty.

You can install Putty following this link https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

3) Once Putty is installed, connect to your server:

- IP: 192.168.20.xxx

- User: xxxx

- Password: yyyy

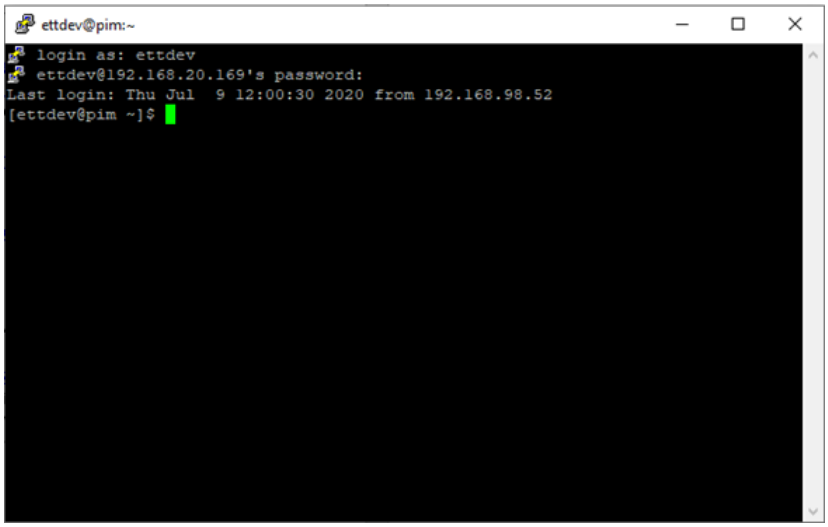

Figure 3. Server's connection

4) Check if the following software are present in your PC. Otherwise, install:
   - Java Runtime Enviroment 7.0: https://www.oracle.com/java/technologies/javase-downloads.html

   - Java Servlet-API, for example Tomcat: http://tomcat.apache.org/download-60.cgi

   - Database Management System, below are the steps to check the presence of PostgreSql and if it is not present how to install it from an open terminal on Putty:

      1) Install PgAdmin on your computer: https://www.pgadmin.org/download/

      2) 'which psql' (Check if PostgreSql is already installed)

      3) 'sudo yum - y install epel-release'

      4) 'sudo yum –y install https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/pgdg-redhat-repo-latest.noarch.rpm'

      5) 'sudo yum install postgis25_12'

5) After verifying the conditions described above (point 4), download the war file package directly from the open terminal on Putty through this command: 'wget https://github.com/52North/SOS/releases/download/v4.4.15/52n-sensorweb-sos-4.4.15.zip'

   It is raccomended to create a special folder with 'mkdir sos' and move the zipped file inside it with 'mv 52n-sensorweb-sos-4.4.15.zip./sos/'. Then, unzip the folder with 'unzip 52n-sensorweb-sos-4.4.15.zip'.

6) Copy the war file in the Tomcat folder with 'cp 52n-sensorweb-sos-4.4.15.war/usr/share/tomcat/webapps', so that it is visible and manageable by the Tomcat manager.

The download and deployment of the war file can also be done from the link: https://github.com/52North/SOS/releases.

After unziping, the war file will be found in UNZIPPED_PACKAGE / bin / target.

Connect to the Tomcat Manager: http: // localhost: 8080 / manager / html and scroll to the "WAR file to deploy" section, select the file and click on "deploy"

7) Create the database and the user with the relative password.

b. Pre-configuration

1. Download and configure SOS following this link: http://192.168.20.xxxxxxx/52n-sos-webapp/
2. Log in with your credentials (Username and password)
3. At the first access, setup the SOS server: insert User, Password, Database and all other required data

At every time, through the administration panel, it is possible to make changes of the data entered during the initial configuration and to set up new optional configurations.

c. Installing the package

Now you are ready to configure the 52N SOS server:

1) Open PgAdmin and connect to the newly created server.
2) Click on “Add New Server” and complete the requested data such as Name, Hostname/Address, Port, Database, User and Password.
3) If the data entered are correct, then you will find the new server in the menu on the left.
4) You can navigate it and find, in the “Database” submenu, the database created previously.

After the above procedure is completed, you should visualize a dashboard of the database, as shown in the image below:

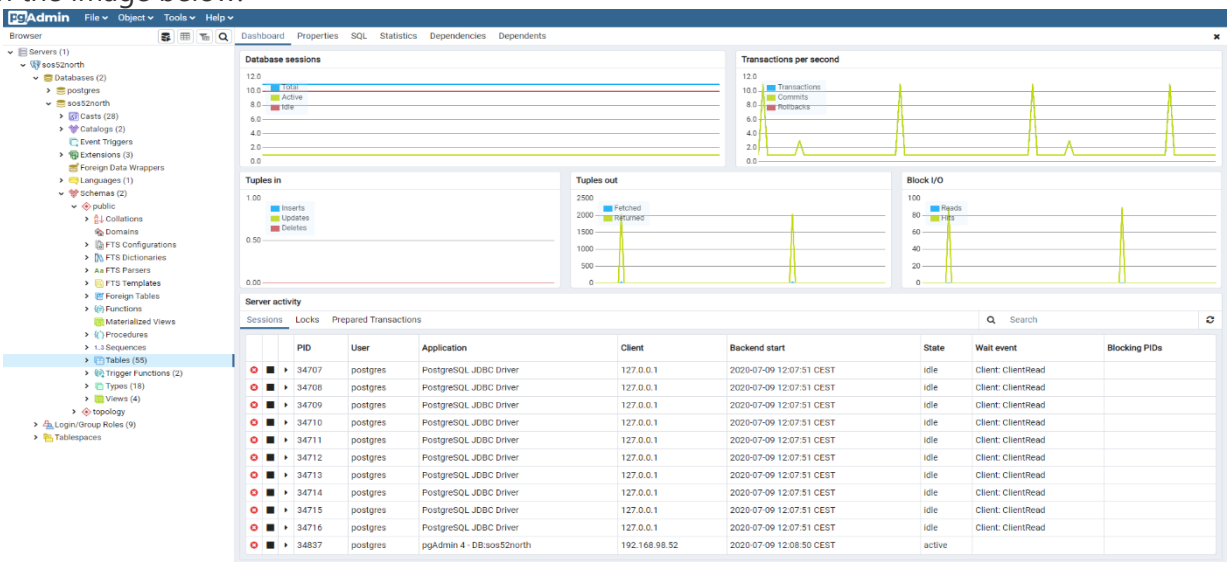

Figure 4. PgAdmin screen

For more information on installation requirements and configurations, you can directly consult the official website of the 52N SOS server at the following link: https://wiki.52north.org/SensorWeb/SensorObservationServiceIVDocumentation

4. SOS Services Configuration

This section describes how to configure Sensor and Observation services, using 52N SOS server.

a. Insert Sensor procedures

In order to insert the sensor in the server, please follow these steps:

1) Enter the measurements via the previously installed webapp and log in with the same username and password used before.
   Click on "Client" and choose the request "InsertSensor"

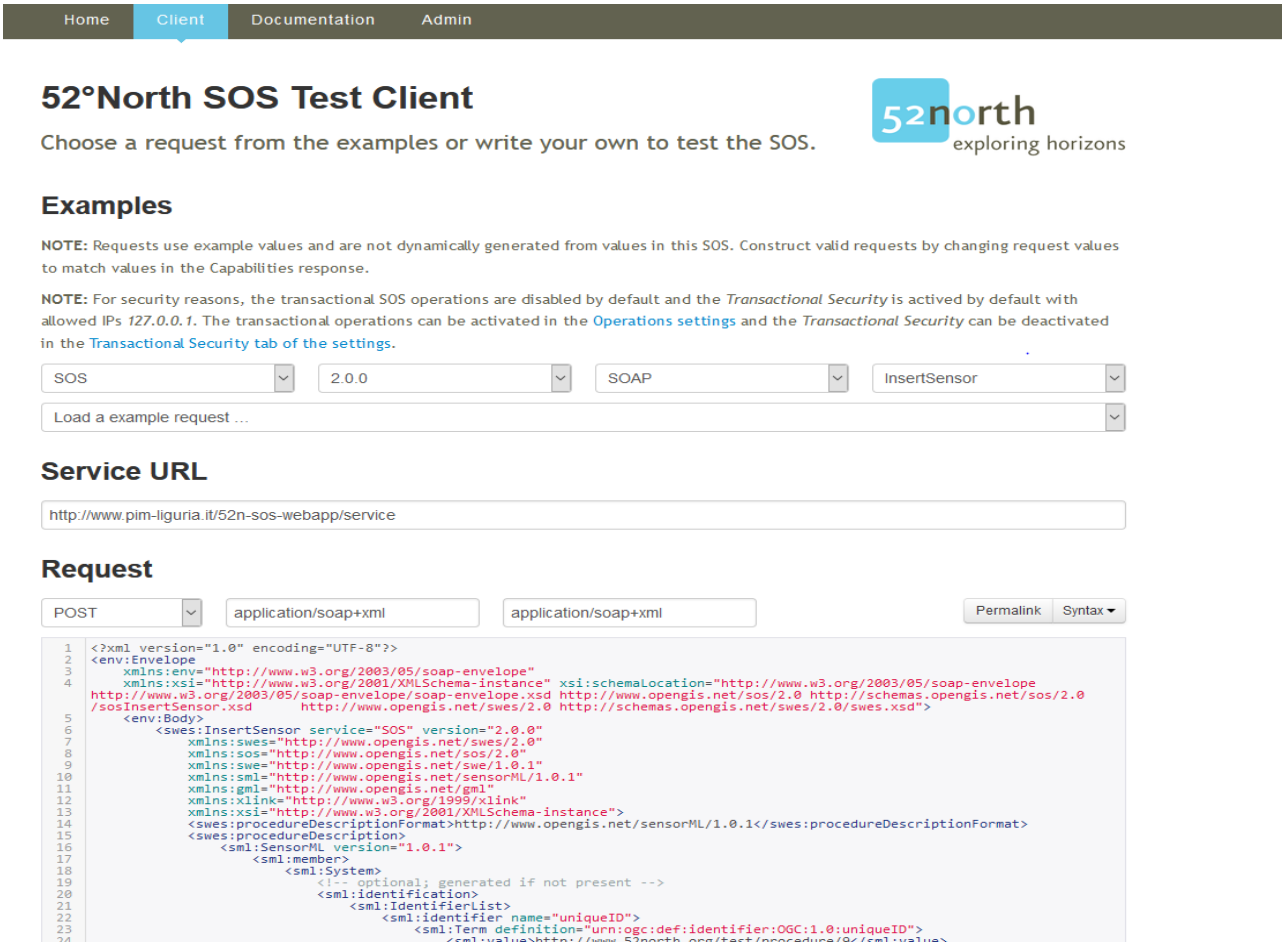

Figure 5. 52N SOS Test Client

2) Create the specific "Sensor" and insert the Service URL, as shown below:

   The data proposed in the figure above are just an example in order to show how to insert the sensor. Data should be modified accordingly when inserting new sensors.

3) Enter your XML code
   (i.e. http://www.pim-liguria.it/erddap/info/AMP_Portofino_MEDA2/index.html):

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
   xmlns:env="http://www.w3.org/2003/05/soap-envelope"
                                                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2003/05/soap-envelope http://www.w3.org/2003/05/soap-
envelope/soap-envelope.xsd                                  http://www.opengis.net/sos/2.0
http://schemas.opengis.net/sos/2.0/sosInsertSensor.xsd                 http://www.opengis.net/swes/2.0
http://schemas.opengis.net/swes/2.0/swes.xsd">
   <env:Body>
      <swes:InsertSensor
         xmlns:swes="http://www.opengis.net/swes/2.0"
         xmlns:sos="http://www.opengis.net/sos/2.0"
         xmlns:swe="http://www.opengis.net/swe/1.0.1"
         xmlns:sml="http://www.opengis.net/sensorML/1.0.1"
         xmlns:gml="http://www.opengis.net/gml"
         xmlns:xlink="http://www.w3.org/1999/xlink"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" service="SOS" version="2.0.0">
          <swes:procedureDescriptionFormat>http://www.opengis.net/sensorML/1.0.1</swes:procedu
reDescriptionFormat>
         <swes:procedureDescription>
            <sml:SensorML version="1.0.1">
               <sml:member>
                  <sml:System>
                     <!-- optional; generated if not present -->
                     <sml:identification>
                        <sml:IdentifierList>
                           <sml:identifier name="uniqueID">
                              <sml:Term definition="urn:ogc:def:identifier:OGC:1.0:uniqueID">
                                 <sml:value>Meda2</sml:value>
                              </sml:Term>
                           </sml:identifier>
```

```
                    <sml:identifier name="longName">
                        <sml:Term definition="urn:ogc:def:identifier:OGC:1.0:longName">
                            <sml:value>AMP Portofino MEDA2</sml:value>
                        </sml:Term>
                    </sml:identifier>
                    <sml:identifier name="shortName">
                        <sml:Term definition="urn:ogc:def:identifier:OGC:1.0:shortName">
                            <sml:value>AMP Portofino MEDA2</sml:value>
                        </sml:Term>
                    </sml:identifier>
                </sml:IdentifierList>
            </sml:identification>
            <sml:capabilities name="offerings">
                <!-- Special capabilities used to specify offerings. -->
                <!-- Parsed and removed during InsertSensor/UpdateSensorDescription,
added during DescribeSensor. -->
                <!-- Offering is generated if not specified. -->
                <swe:SimpleDataRecord>
                    <!-- Field name or gml:name is used for the offering's name -->
                    <swe:field name="Offering for Meda2">
                        <swe:Text definition="urn:ogc:def:identifier:OGC:offeringID">
                            <gml:name>Offering for Meda2</gml:name>
                            <swe:value>Meda2</swe:value>
                        </swe:Text>
                    </swe:field>
                </swe:SimpleDataRecord>
            </sml:capabilities>
            <sml:capabilities name="featuresOfInterest">
                <!-- Special capabilities used to specify features of interest. -->
                <!-- Parsed and removed during InsertSensor/UpdateSensorDescription,
added during DescribeSensor. -->
```

```
                    <swe:SimpleDataRecord>
                        <swe:field name="featureOfInterestID">
                            <swe:Text>
                                <swe:value>Meda2
</swe:value>
                            </swe:Text>
                        </swe:field>
                    </swe:SimpleDataRecord>
                </sml:capabilities>
                <sml:capabilities name="metadata">
                    <swe:SimpleDataRecord>
                        <!-- status indicates, whether sensor is insitu (true) or remote (false) -->
                        <swe:field name="insitu">
                            <swe:Boolean definition="insitu">
                                <swe:value>true</swe:value>
                            </swe:Boolean>
                        </swe:field>
                        <!-- status indicates, whether sensor is mobile (true) or fixed/stationary (false)
-->
                        <swe:field name="mobile">
                            <swe:Boolean definition="mobile">
                                <swe:value>false</swe:value>
                            </swe:Boolean>
                        </swe:field>
                    </swe:SimpleDataRecord>
                </sml:capabilities>
                <sml:position name="sensorPosition">
                    <swe:Position referenceFrame="urn:ogc:def:crs:EPSG::4326">
                        <swe:location>
                            <swe:Vector gml:id="STATION_LOCATION">
                                <swe:coordinate name="easting">
```

```
                <swe:Quantity axisID="x">
                    <swe:uom code="degree"/>
                    <swe:value>9.165838</swe:value>
                </swe:Quantity>
            </swe:coordinate>
            <swe:coordinate name="northing">
                <swe:Quantity axisID="y">
                    <swe:uom code="degree"/>
                    <swe:value>44.31372</swe:value>
                </swe:Quantity>
            </swe:coordinate>
            <swe:coordinate name="altitude">
                <swe:Quantity axisID="z">
                    <swe:uom code="m"/>
                    <swe:value>0</swe:value>
                </swe:Quantity>
            </swe:coordinate>
        </swe:Vector>
    </swe:location>
</swe:Position>
</sml:position>
<sml:inputs>
<sml:InputList>
    <sml:input name="observable_property_Meda2">
        <swe:ObservableProperty definition="observable_property_Meda2"/>
    </sml:input>
</sml:InputList>
</sml:inputs>
<sml:outputs>

    <sml:OutputList>
```

```
                <sml:output name="TEMP">
                  <swe:Quantity definition="TEMP">
                    <swe:uom code="degree_Celsius"/>
                  </swe:Quantity>
                </sml:output>
                <sml:output name="PSAL">
                  <swe:Quantity definition="PSAL">
                    <swe:uom code="psu"/>
                  </swe:Quantity>
                </sml:output>
                <sml:output name="CNDC">
                  <swe:Quantity definition="CNDC">
                    <swe:uom code="S/m"/>
                  </swe:Quantity>
                </sml:output>
                        </sml:OutputList>
                    </sml:outputs>
                  </sml:System>
                </sml:member>
              </sml:SensorML>
            </swes:procedureDescription>
<swes:observableProperty>TEMP</swes:observableProperty>
<swes:observableProperty>PSAL</swes:observableProperty>
<swes:observableProperty>CNDC</swes:observableProperty>

            <swes:metadata>
              <sos:SosInsertionMetadata>
                                <sos:observationType>http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement</sos:observationType>
                                <sos:observationType>http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_CategoryObservation</sos:observationType>
```

<sos:observationType>http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_CountObservation</sos:observationType>

<sos:observationType>http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_TextObservation</sos:observationType>

<sos:observationType>http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_TruthObservation</sos:observationType>

<sos:observationType>http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_GeometryObservation</sos:observationType>

<sos:observationType>http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_ComplexObservation</sos:observationType>

<sos:observationType>http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_SWEArrayObservation</sos:observationType>

<sos:observationType>http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_ReferenceObservation</sos:observationType>

<!-- multiple values possible -->

<sos:featureOfInterestType>http://www.opengis.net/def/samplingFeatureType/OGC-OM/2.0/SF_SamplingPoint</sos:featureOfInterestType>

</sos:SosInsertionMetadata>

</swes:metadata>

</swes:InsertSensor>

</env:Body>

</env:Envelop

NOTE: remember to provide, each time, specific data in the sections highlited in yellow

4) After requesting the server, you will find the response code. If this is equal to 200, the call is successful. Otherwise, it is failed.
5) If the procedure has been executed correctly, move to "Client" → "Sensor Web Thin Client (Helgoland)" and check that the sensor is available.
6) Choose the Timeseries. If you can select the sensor just inserted, then it means that the InsertSensor operation is correct and you can proceed to insert the relevant measurements.

b. Insert Observation procedures

To insert Observation's measurements, repeat the initial procedures described in paragraph 3.1 and follow specific cases:

CASE 1

1) Enter the measurements via the previously installed webapp and log in with the same username and password used before.
2) Click on "Client" and choose the request "InsertObservation"
3) Create the specific "Observation" and insert the observed properties from the server
4) Access the server, select the typology of measurements, choose the file type and click on Submit, accordingly to your Data Storage System. A page will be opened with a data table showing all the measurements made by the sensor.
5) Insert the measurements as showed below (i.e we used an example entering a measurement value for the TEMP field of the AMP Portofino MEDA sensor as described previously):


```
(
  "request": "InsertObservation",
  "service": "SOS",
  "version": "2.0.0",
  "offering": "Meda2",
  "observation": {
                          "type":          "http://www.opengis.net/def/observationType/OGC-
OM/2.0/OM_Measurement",
    "procedure": "Meda2",
    "observedProperty": "TEMP",
    "featureOfInterest": {
      "identifier": {
        "value": "Meda2",
        "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown"
      },
      "name": [
        {
          "value": "Meda2",
          "codespace": "http://www.opengis.net/def/nil/OGC/0/unknown"
        }
      ],
      "geometry": {
        "type": "Point",
        "coordinates": [
          44.313725,
          9.165838
        ],
        "crs": {
          "type": "name",
```

```
        "properties": {
          "name": "EPSG:4326"
        }
      }
    }
  },
  "phenomenonTime": "2018-07-20T19:16:16.000+00:00",
  "resultTime": "2018-07-20T19:16:16.000+00:00",
  "result": {
    "uom": "celsius",
    "value": 26.037
  }
 }
}
```

NOTE: remember to provide, each time, specific data in the sections highlited in yellow

CASE 2

Entering all the measurements, one at a time, may be not efficient. Therefore, a Python script has been created to be able to insert the data sequentially.

Apply the following procedures to use the Python script:

1) Create the file
2) Insert and modify the typology of fields and data necessary for the sensor, as shown below:

```
import urllib.request
import requests
import json

urlSOS = "http://www.pim-liguria.it/52n-sos-webapp/service"
headers = {
  'Authorization': 'Basic YWRtaW46M1RUMTIzIUA=',
  'Content-Type': 'application/json'
}
payload_template = "{\r\n  \"request\": \"InsertObservation\",\r\n  \"service\": \"SOS\",\r\n  \"version\": \"2.0.0\",\r\n  \"offering\": \"Meda2\",\r\n  \"observation\": {\r\n\"type\":
\"http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement\",\r\n \"procedure\": \"Meda2\",\r\n \"observedProperty\": \"[PARAM]\",\r\n \"featureOfInterest\": {\r\n \"identifier\": {\r\n \"value\":
\"Meda2\",\r\n \"codespace\": \"http://www.opengis.net/def/nil/OGC/0/unknown\"\r\n},\r\n\"name\": [\r\n{\r\n\"value\": \"Meda2\",\r\n\"codespace\":
\"http://www.opengis.net/def/nil/OGC/0/unknown\"\r\n}\r\n],\r\n\"geometry\": {\r\n\"type\": \"Point\",\r\n\"coordinates\": [\r\n44.313725,\r\n9.165838\r\n],\r\n\"crs\": {\r\n\"type\": \"name\",\r\n\"properties\":
{\r\n\"name\": \"EPSG:4326\"\r\n}\r\n}\r\n},\r\n\"phenomenonTime\": \"[TIME].000+00:00\",\r\n    \"resultTime\": \"[TIME].000+00:00\",\r\n    \"result\": {\r\n\"uom\": \"[UOM]\",\r\n      \"value\":
[VALUE]\r\n    }\r\n  }\r\n}"

url="http://www.pim-liguria.it/erddap/tabledap/AMP_Portofino_MEDA2.json?
FileTime%2Ctime%2CTIME_QC%2Cdepth%2CDEPTH_QC%2CPRES%2CPRES_QC%2Clatitude%2Clongitude%2CPOSITION_QC%2CTEMP%2CTEMP_QC%2CPSAL%2CCNDC&orderBy(%22time%22)"
savepath_file="meda2.json"
urllib.request.urlretrieve(url, savepath_file)
read_file= open(savepath_file, "r")
data = json.load(read_file)
read_file.close()
for row in data["table"]["rows"]:
  time=row[1].replace("Z","")
  temp=row[10]
  psal=row[12]
  cndc=row[13]
  payload=payload_template.replace("[PARAM]","TEMP").replace("[TIME]",time).replace("[VALUE]",str(temp)).replace("[UOM]","degree_Celsius")
  response = requests.request("POST", urlSOS, headers=headers, data = payload)
  payload=payload_template.replace("[PARAM]","PSAL").replace("[TIME]",time).replace("[VALUE]",str(psal)).replace("[UOM]","psu")
  response = requests.request("POST", urlSOS, headers=headers, data = payload)
  payload=payload_template.replace("[PARAM]","CNDC").replace("[TIME]",time).replace("[VALUE]",str(cndc)).replace("[UOM]","S/m")
  response = requests.request("POST", urlSOS, headers=headers, data = payload)
  print("DONE " + time)
```

Figure 6. Script Python to insert Observation

The procedure to request the server through the payload_template is defined in Figure 6.

The measurements entered in the script are those used in the example (TEMP, PSAL and CNDC fields of the sensor AMP Portofino MEDA).

NOTE: remember to provide, each time, specific data

3) Finally launch the script from the terminal with the command python filename.py

5. Services Validation

In this section, we describe the procedures to validate the services.

a. Timeseries

In order to verify if the Observation's measurement expalined in Case 1 is inserted correctly, go back to Sensor Web Thin Client (Helgoland).

Select the Timeseries, as shown in Figure 7:

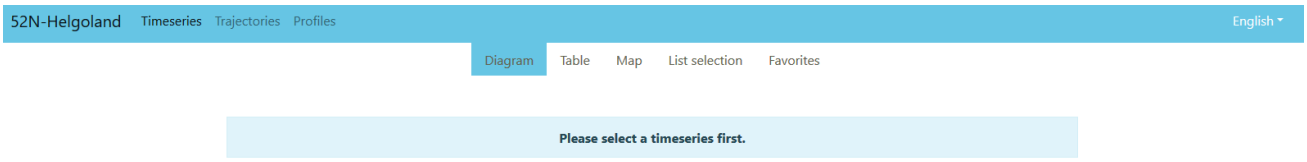

Figura 7. Timeseries selection

In the timeseries, you will find a map with the measurement inserted, as showed in Figure 8:

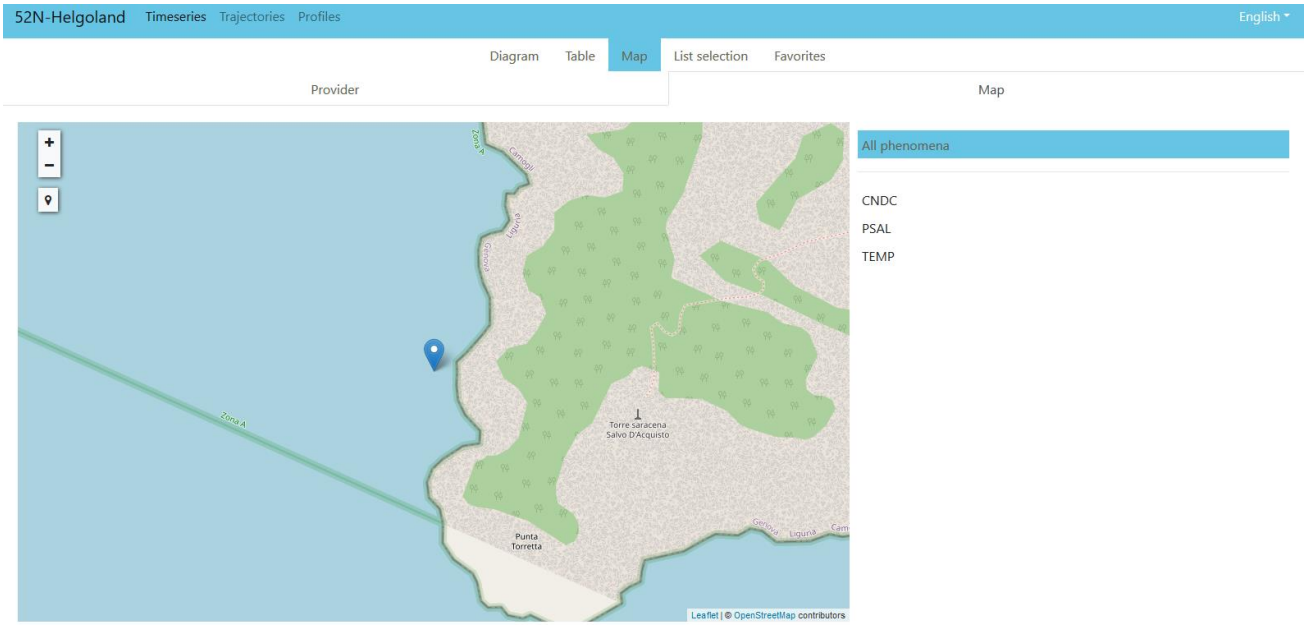

Figura 8. Timeseries Map

If the map is available and the measurements visible, then the procedures have been executed correctly.

Otherwise, start again.

b. Script validation procedures

In order to verify if the Observation's measurement, as explained in Case 2, is inserted correctly, go back to Sensor Web Thin Client (Helgoland).

You can see the insertion of the measurements both while the script is working (through the graph) and at the end of the process on a weekly base.

6. References

European Commission (2010), Marine Knowledge 2020 Marine Data and Observation for Smart and Sustainable Growth. Commission Communication COM (2010) 461, Publications Office of the European Union.

European Commission (2012), Marine Knowledge 2020 from Seabed Mapping to Ocean Forecasting. Green Paper, Publications Office of the European Union, Luxembourg.

Bröring, A., Echterhoff, J., Jirka, S., Simonis, I., Everding, T., Stasch, C., ... & Lemmens, R. (2011). New generation sensor web enablement. Sensors, 11(3), 2652-2699.