Deliverable D4.1.2 Concerted Approach V2

Appendix 4: OGC documentation: Pan-European visualisation

Revision History			
Version	**Date**	**Modified by**	**Comments**
0.1	2011-11-14	Martin Scholl	Initial document template
0.2	2011-12-19	Mihai Bartha	Initial contents
0.3	2011-12-20	Peter Kutschera	Review
0.4	2012-01-13	Sascha Schlobinski	Revision

Table of Contents

1. Management Summary 5
2. Component usage 6
3. Pan-European service interfaces 7
3.1. Metadata 7
3.2. SOS interface 7
3.3. DataHandler interface 10
References 18

Table of Figures

Figure 1 - Components 6
Figure 2 - SOS client server interaction 7

Glossary

Information product	Raw data, such as the results of mathematical modelling, and the analysis thereof, will often need to be packaged in such a way as to be accessible to the various stakeholders of an analysis. The medium can be one of a wide variety, such as print, photo, video, slides, or web pages. The term *information product* refers to such an entity.
Model	A *model* is a simplified representation of a system, usually intended to facilitate analysis of the system through manipulation of the model. In the SUDPLAN context the term can be used to refer to mathematical models of processes or spatial models of geographical entities.
Profile	Within SUDPLAN a *profile* is a set of configuration parameters which are associated with an individual or group, and which are remembered in order to facilitate repeated use of the system.
Report	A *report* is a particular type of information product which is usually static and might integrate still images, static data representations, mathematical expressions, and narrative to communicate an analytical result to others.
Scenario	A *scenario* is a set of parameters, variables and other conditions which represent a hypothetical situation, and which can be analysed through the use of models in order to produce hypothetical outcomes.
Scenario Management System	*Scenario Management System* is synonymous with SUDPLAN platform
SUDPLAN application	A *SUDPLAN application* is a decision support system crafted by using the SUDPLAN platform and integrating models, data, sensors, and other services to meet the requirements of the particular application.
SUDPLAN platform	The *SUDPLAN platform* is an ensemble of software components which support the development of SUDPLAN applications.
SUDPLAN system	*SUDPLAN system* is synonymous with SUDPLAN application

User	The term *user* refers to people who have a more or less direct involvement with a system. Primary users are directly and frequently involved, while secondary users may interact with the system only occasionally or through an intermediary. Tertiary users may not interact with the system but have a direct interest in the performance of the system.
Web-based	Computer applications are said to be *web-based* if they rely on or take advantage of data and/or services which are accessible via the World Wide Web using the Internet.

1. Management Summary

This document contains the PanEuropean model specific instructions needed to access the data. General information can be found in [SUDPLAN specific OGC services - abstract spec], which is a required reading.

This is a living document; the actual version is available from Sudplan.EU.

This version reflects the implementation at the end of 2011 and concentrates to the usage of the services using Timeseries-Toolbox, as this is the method used in the project. The OGC-related part will be completed in 2012.

2. Component usage

The PanEuropean service provides access to sensor data via a SOS interface. This sensor data is timeseries in various temporal resolutions (some as high as 10 minutes) over rather long time periods (of more than 10 years).

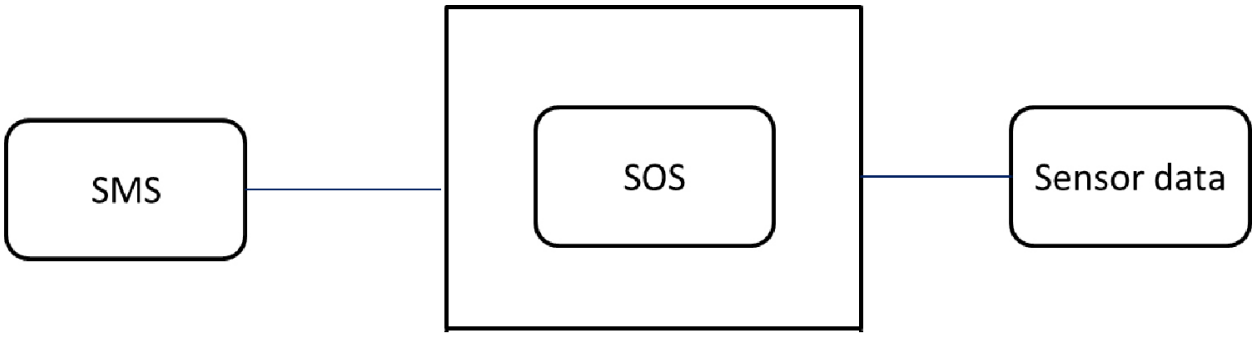

Figure 1 - Components

As depicted above the most tipical use-case involves the Scenarion Management System (SMS) accessing the PanEuropean SOS service. From the component POV the SMS accesses the SOS service through by using the SOSClientHandler component. The SOSClientHandler is marely a protocoll translator implementing the DataHandler interface (defined in the TimeSeries API) and acting a SOS client by implementing the SOS protocol. The block SensorData could be any data access component that implements the DataHandler interface. In our specific case is the SOSPanEuropeanHandler component that provides the functionality of accessing the PanEuropean data. It does so by invoking scripts. The SOS service implements the SOS protocoll (server) and acts as a client to the SOSPanEuropeanHandler DataHandler interface. Obvious advanteges are the possibility to have the client, server, handler and scripts in different locations of the network. Easily replace the handler, and use the client to connect to a different service.

3. Pan-European service interfaces

This section describes the OGC service interfaces to the PanEuropean SOS client. It is assumed that the reader knows about the OGC standards, especially SOS, SPS, O&M and SensorML.

3.1. Metadata

The service is located at http://sudplan.ait.ac.at:8080/

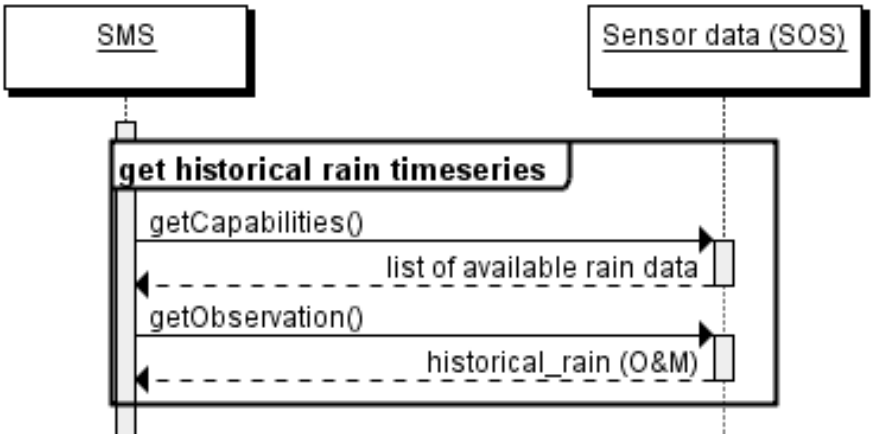

Figure 2 - SOS client server interaction

3.2. SOS interface

3.2.1 GetCapabilities

In Figure 2 the client (integral part of the SMS) upon instantiation invokes the GetCapabilities operation and retrieves from the server the list of the available timeseries including their meta-information. The SOS service follows the SUDPLAN convention of having exactly one offering by available timeseries. The list of available timeseries can be obtained from the ObservationOfferingList element of the Capabilities document.

3.2.2 GetObservation

The getObservation request does not differ from ref (OGC SOS implementation spec). It provides access to all model results. The SOS client generates and sends a GetObservation request to the server. Such a request contains filters on the requested data such as SOS offering name, procedure, featureOfInterest, observed property as well as temporal and spatial filters.

Example request containg temporal and spatial filters:

```
<sos:GetObservation xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:sos="http://www.opengis.net/sos/1.0" ... service="SOS" version="1.0.0">
 <sos:offering>EUROPE-O3-A1B3-coverage-10y</sos:offering>
 <sos:eventTime>
  <ogc:TM_During>
   <ogc:PropertyName>urn:ogc:data:time:iso8601</ogc:PropertyName>
   <gml:TimePeriod>
    <gml:beginPosition>1965-01-01T00:00:00+0100</gml:beginPosition>
```

```
    <gml:endPosition>2000-01-01T21:00:00+0100</gml:endPosition>
   </gml:TimePeriod>
  </ogc:TM_During>
 </sos:eventTime>
 <sos:procedure>urn:ogc:object:EUROPE:O3:A1B3:10y</sos:procedure>
 <sos:observedProperty>urn:ogc:def:property:OGC:O3</sos:observedProperty>
 <sos:featureOfInterest>
  <ogc:BBOX>
   <ogc:PropertyName>gml:location</ogc:PropertyName>
    <gml:Envelope>
     <gml:lowerCorner>14.18 48.24</gml:lowerCorner>
     <gml:upperCorner>14.18 48.24</gml:upperCorner>
    </gml:Envelope>
   </ogc:BBOX>
  </sos:featureOfInterest>
<sos:responseFormat>text/xml;subtype=&quot;om/1.0.0&quot;</sos:responseFormat
>
 <sos:resultModel>om:Observation</sos:resultModel>
 <sos:responseMode>inline</sos:responseMode>
</sos:GetObservation>
```

Example response:

```
<om:ObservationCollection xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:sos="http://www.opengis.net/sos/1.0"...>
 <om:member xlink:type="simple">
  <om:Observation>
   <gml:description>none</gml:description>
   <gml:boundedBy>
    <gml:Envelope srsName="EPSG:4326">
     <gml:lowerCorner>18.06 59.32</gml:lowerCorner>
     <gml:upperCorner>18.06 59.32</gml:upperCorner>
    </gml:Envelope>
   </gml:boundedBy>
   <om:procedure xlink:type="simple"
xlink:href="urn:ogc:object:STHLM:temp:A1B3"/>
   <om:observedProperty xlink:type="simple"
xlink:href="urn:ogc:def:property:OGC:temp"/>
   <om:featureOfInterest>
    <sa:SamplingPoint>
     <sa:sampledFeature xlink:href=""/>
     <sa:position>
      <gml:Point srsName="EPSG:4326">
       <gml:pos srsName="EPSG:4326">18.06 59.32</gml:pos>
      </gml:Point>
     </sa:position>
    </sa:SamplingPoint>
   </om:featureOfInterest>
   <om:result>
    <swe:DataArray>
     <swe:elementCount>
      <swe:Count>
       <swe:value>10</swe:value>
      </swe:Count>
     </swe:elementCount>
     <swe:elementType name="RootRecord">
      <swe:DataRecord>
       <swe:field name="Timestamp">
        <swe:Time definition="urn:ogc:data:time:iso8601"/>
       </swe:field>
```

```
       <swe:field name="value">
        <swe:Quantity definition="urn:ogc:def:property:OGC:temp">
         <swe:uom code="urn:ogc:def:uom:OGC::K"/>
        </swe:Quantity>
       </swe:field>
      </swe:DataRecord>
     </swe:elementType>
     <swe:encoding>
      <swe:TextBlock blockSeparator="@" decimalSeparator="." tokenSeparator="
"/>
     </swe:encoding>
     <swe:values>1961-01-01T00:00:00+0100 280.086 @1962-01-01T00:00:00+0100
279.513 @1963-01-01T00:00:00+0100 279.681 @1964-01-01T00:00:00+0100 280.5
@1965-01-01T00:00:00+0100 279.966 @1966-01-01T00:00:00+0100 279.726 @1967-01-
01T00:00:00+0100 278.475 @1968-01-01T00:00:00+0100 279.025 @1969-01-
01T00:00:00+0100 279.5 @1970-01-01T00:00:00+0100 278.138 @</swe:values>
    </swe:DataArray>
   </om:result>
  </om:Observation>
 </om:member>
</om:ObservationCollection>
```

3.2.3 GetGeatureOfInterest

The GetFeatureOfInterest request is sometimes generated by the client and sent to the server in order to retrieve the samplingFeature of the timeseries. For most situations the sampling feature is integral part of the Observation response. Our implementation uses the sa:samplingPoint feature defined in SA Sampling and defines one additional samplingFeature type, namely ait:samplingGrid. The samplingGrid element contains a gml:RectifiedGrid element that describes the origin, offset vectors, and dimensions of the grid for which values are available. At the same time it contains a reference to the sampled feature through the sa:sampledFeature element. The samplingGrid schema is listed as an example of the DescribeFeatureType request of the next chapter.

Sampling point example:

```
<sa:SamplingPoint>
    <sa:sampledFeature xlink:href=""/>
    <sa:position>
        <gml:Point srsName="EPSG:4326">
            <gml:pos srsName="EPSG:4326">18.06 59.32</gml:pos>
        </gml:Point>
    </sa:position>
</sa:SamplingPoint>
```

Sampling grid example

```
<ait:SamplingGrid gml:id="AITSGID0">
    <gml:description>grid</gml:description>
    <sa:sampledFeature xlink:href=""/>
    <gml:RectifiedGrid srsName="EPSG:4326">
        <gml:limits>
            <gml:GridEnvelope>
                <gml:low>0 0</gml:low>
                <gml:high>95 77</gml:high>
            </gml:GridEnvelope>
        </gml:limits>
        <gml:axisName>x</gml:axisName>
```

```
        <gml:axisName>y</gml:axisName>
        <gml:origin>
            <gml:Point srsName="EPSG:4326">
                <gml:pos>-12.25 32.75</gml:pos>
            </gml:Point>
        </gml:origin>
        <gml:offsetVector srsName="EPSG:4326">0.5 0.0</gml:offsetVector>
        <gml:offsetVector srsName="EPSG:4326">0.0 0.5</gml:offsetVector>
    </gml:RectifiedGrid>
</ait:SamplingGrid>
```

3.2.4 DescribeFeatureType

DescribeFeatureType returns the XML schema for the specified GML feature advertised in GetCapabilities or part of the Observation result. This may be used to obtain a description of the type of a feature.

The following contains the schema (featureType) of the ait:samplingGrid:

```
<schema xmlns=http://www.w3.org/2001/XMLSchema
xmlns:gml=http://www.opengis.net/gml
xmlns:sa=http://www.opengis.net/sampling/1.0
xmlns:ait="http://www.ait.ac.at/sampling"
targetNamespace=http://www.ait.ac.at/sampling elementFormDefault="qualified"
attributeFormDefault="unqualified" version="1.0.0">
 <import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
 <import namespace="http://www.opengis.net/sampling/1.0"
schemaLocation="http://schemas.opengis.net/sampling/1.0.0/sampling.xsd"/>
 <complexType name="SamplingGridType">
  <complexContent>
   <extension base="sa:SpatiallyExtensiveSamplingFeatureType">
    <sequence>
     <element ref="gml:RectifiedGrid"/>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
 <element name="SamplingGrid" type="ait:SamplingGridType"
substitutionGroup="gml:_Feature"/>
</schema>
```

3.3. DataHandler interface

The DataHandler implementation, SOSClientDataHandler (java class), need to be instantiated and the endpoint URL of the SOSServer has to be set. One test implementation is accessible at http://enviro3.ait.ac.at:8080

```
SOSClientDataHandler dh = new SOSClientDataHandler();
try {
dh.setEndpoint(new URL("http://enviro3.ait.ac.at:8080"));
} catch (MalformedURLException ex) {
     logger.error("misformed URL.", ex);
}
```

The SOSDataHandler instance has to be opened in order for it to actually connect to the SOS server. Upon open the client invokes the getCapabilities on the SOS service in order to retrieve

the Capabilities document containing meta-information describing the offered datasets (timeseries) and initialize the internal structures.

```
try {
dh.open();
} catch (IOException e) {

}
```

3.3.1 Retrieve filter names and values

Each DataPoint is uniquely identified by a set of properties called filter properties. These properties can be used to query datapoints from the DataHandler. The filter properties are key value pairs. Both key and value are of type java.lang.String. First the possible filter names and values have to be retrieved in order to have a basis for DataPoint queries. The following code lists the existing filter keys and their valid values.

```
System.out.println("Filter Names:");
Set filterNames = dh.getFilterNames(Access.READ);
for (String name : (Set<String>) filterNames) {
   System.out.println("\t" + name);
   for (Object value : dh.getFilterValues(name, null, Access.READ)) {
      System.out.println("\t" + value);
   }
}
```

3.3.2 Get Datapoint

In order to retrieve a set of Datapoints the filter (Properties java class) has to be first instantiated and initialized. One can provide here a number of filter key values as retrieved in 2.1.2. Retrieval of the Datapoint set is done by invocation of the DataHandler getDatapoints(...) method.

```
Properties filter = new Properties();
filter.put("ts:offering", "airquality_echam5a1b3_ozone_10Y");
Set<Datapoint> dpSet = dh.getDatapoints(filter, Access.READ);
```

Depending on the query parameters the getDatapoints method will retrieve zero or more Datapoints.

The list of datapoints is long. In the moment of writing this it is:

airquality_echam5a1b3rcp4.5_no2_10Y,
airquality_echam5a1b3rcp4.5_no2_1M,
airquality_echam5a1b3rcp4.5_no2_1Y,
airquality_echam5a1b3rcp4.5_no2_1d,
airquality_echam5a1b3rcp4.5_ozone_10Y,
airquality_echam5a1b3rcp4.5_ozone_1M,
airquality_echam5a1b3rcp4.5_ozone_1Y,
airquality_echam5a1b3rcp4.5_ozone_1d,
airquality_echam5a1b3rcp4.5_sia_10Y,

airquality_echam5a1b3rcp4.5_sia_1M,
airquality_echam5a1b3rcp4.5_sia_1Y,
airquality_echam5a1b3rcp4.5_sia_1d,
airquality_echam5a1b3rcp4.5_so2_10Y,
airquality_echam5a1b3rcp4.5_so2_1M,
airquality_echam5a1b3rcp4.5_so2_1Y,
airquality_echam5a1b3rcp4.5_so2_1d,
airquality_hadleya1brcp4.5_no2_10Y,
airquality_hadleya1brcp4.5_no2_1M,
airquality_hadleya1brcp4.5_no2_1Y,
airquality_hadleya1brcp4.5_no2_1d,
airquality_hadleya1brcp4.5_ozone_10Y,
airquality_hadleya1brcp4.5_ozone_1M,
airquality_hadleya1brcp4.5_ozone_1Y,
airquality_hadleya1brcp4.5_ozone_1d,
airquality_hadleya1brcp4.5_sia_10Y,
airquality_hadleya1brcp4.5_sia_1M,
airquality_hadleya1brcp4.5_sia_1Y,
airquality_hadleya1brcp4.5_sia_1d,
airquality_hadleya1brcp4.5_so2_10Y,
airquality_hadleya1brcp4.5_so2_1M,
airquality_hadleya1brcp4.5_so2_1Y,
airquality_hadleya1brcp4.5_so2_1d,
climate_echam5a1b3_prec_10Y,
climate_echam5a1b3_prec_1M,
climate_echam5a1b3_prec_1Y,
climate_echam5a1b3_prec_1d,
climate_echam5a1b3_prec_1h,
climate_echam5a1b3_prec_30m,
climate_echam5a1b3_temp_10Y,
climate_echam5a1b3_temp_1M,
climate_echam5a1b3_temp_1Y,
climate_echam5a1b3_temp_1d,
climate_hadleya1b_prec_10Y,
climate_hadleya1b_prec_1M,
climate_hadleya1b_prec_1Y,
climate_hadleya1b_prec_1d,
climate_hadleya1b_prec_1h,
climate_hadleya1b_prec_30m,
climate_hadleya1b_temp_10Y,
climate_hadleya1b_temp_1M,
climate_hadleya1b_temp_1Y,
climate_hadleya1b_temp_1d

The offering names are built by parts separated by “_” as follows:

Part	Example	Description
1	Climate	• Climate: Climate data, this is precipitation and temperature • Airquality: Gases in the atmosphere
2	echam5a1b3	Name of climate scenario
3	prec	Phenomenon, e. g. `prec (precipitation), temp (temperature)`
4	10Y	Time resolution, e.g. 10Y (one value every 10 years), 1M (Month), 1d (1 day), 1h (1 Hour).

3.3.3 Datapoint properties

Each Datapoint has a set of properties that describes the timeseries. These properties can be retrieved by an invocation of the Datapoint's getProperties method. Datapoint properties are key value pairs. Key is of type String and value of type Object. Following code lists the properties.

```
for (String propName : dp.getProperties().keySet()) {
   System.out.println(propName + ": " + dp.getProperties().get(propName));
}
```

The following properties are available on datapoints:

Name	**Type, Value (example)**	**Comment, Example**
TimeSeries.GEOMETRY ("ts:geometry")	Envelope Env[-12.25 : 32.75, 35.25 : 71.25]	Bounding box. Coordinates of point or area
TimeSeries. AVAILABLE_DATA_MIN ("ts:available_data_min")	TimeStamp	TimeStamp of first available value
TimeSeries. AVAILABLE_DATA_MAX ("ts:available_data_max")	TimeStamp	TimeStamp of last available value

3.3.4 Setup temporal filters

A temporal filter has to be provided, with most Datapoint implementations, when retrieving the timeseries from a datapoint. The temporal filter has to be of type TimeInterval as defined in the TS-API. The TimeInterval constructor allows for specification of begin and end timestamps as well as open/cloased boundaries of the interval.

```
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
TimeStamp t1 = null, t2 = null;
try {
   t1 = new TimeStamp(sdf.parse("1965-01-01 00:00:00").getTime());
   t2 = new TimeStamp(sdf.parse("2000-01-01 21:00:00").getTime());
} catch (ParseException e) {
    ...
```

```
}
TimeInterval temporalFilter = new TimeInterval( TimeInterval.Openness.OPEN,
t1, t2, TimeInterval.Openness.OPEN);
```

The above request translates into a getObservation request to the server that contains a temporal filter specified by an eventTime element as following:

```
<sos:eventTime>
   <ogc:TM_During>
      <ogc:PropertyName>urn:ogc:data:time:iso8601
      </ogc:PropertyName>
      <gml:TimePeriod>
         <gml:beginPosition>2008-03-01T17:44:15+00
         </gml:beginPosition>
         <gml:endPosition>2008-05-01T17:44:15+00
         </gml:endPosition>
      </gml:TimePeriod>
   </ogc:TM_During>
</sos:eventTime>
```

3.3.5 Setup spatial filters

A spatial filter can be provided when retrieving a Timeseries from a datapoint. The following spatial filter supported by the SOSClient implementation defines an EnvelopeQueryParameter that wraps a com.vividsolutions.jts.geom.Envelope element. Depending on the backend capabilities the provision of this filter will result in a subsetting of the timeseries. The EnvelopeQueryParameter can also be used to specify a single point by encoding an Envelope with a zero area.

```
Envelope envelope = new Envelope();
envelope.init(14.18, 14.18, 48.24, 48.24);
EnvelopeQueryParameter spatialFilter = new EnvelopeQueryParameter();
spatialFilter.setEnvelope(envelope);
```

The above will result in a getObservation request containing a spatial filter in the form of an sos:featureOfInterest element containing a gml:envelope (see below).

```
<sos:featureOfInterest>
  <ogc:BBOX>
  <ogc:PropertyName>gml:location</ogc:PropertyName>
    <gml:Envelope>
      <gml:lowerCorner>48.24 14.18</gml:lowerCorner>
      <gml:upperCorner>48.24 14.18</gml:upperCorner>
    </gml:Envelope>
  </ogc:BBOX>
</sos:featureOfInterest>
```

3.3.6 Retrieving the timeseries

The getTimeseries method will retrieve a TimeSeries based on the spatio-temporal filters provided. The TimeSeries might contain zero or more timestamp/slot values. A Slot, as defined in the TS-API is a containter that associates one or more values with a Timestamp. Upon invocation of the getTimeseries request the client generates and invokes a getObservation

request on the server and translates the retured ObservationCollection response into a TimeSeries specifica representation of the timeseries.

```
TimeSeries ts = dp.getTimeSeries(temporalFilter, spatialFilter);
```

3.3.7 Listing the values

Each value in the TimeSeries has a value key. The set of value keys can be retrieved by retrieving the TimeSeries property TimeSeries.VALUE_KEYS. The set of TimeStamp's in the TimeSeries for which associated values exist can be retrieved by an invocation of the getTimeStamps. A specific value in the TimeSeries can be retrieved by an invocation of the getValue method. The later method takes as parameter a Timestamp and a key value.

```
String[] valKeys = (String[]) ts.getTSProperty(TimeSeries.VALUE_KEYS);
for (String key : valKeys) {
   System.out.println(key);
   for (TimeStamp t : ts.getTimeStamps()) {
      System.out.println(t);
      Object val = ts.getValue(t, key);
      System.out.println(val);
   }
}
```

The timeseries is transported from the server to the client encoded according to the O&M standard as part of the <om:result> element. The result contains a result element that contains three significant parts:

- the result definition contains the structure of the timeseries. In our implementation is a swe:DataArray containing timestamp value pairs. The value's structure (swe:ElementType neme="RootRecord" can be rather complex usually containing a swe:DataRecord element containing one of more fields. Each field corresponds to exactly one key in the TSAPI TimeSeries.VALUE_KEYS set. Each field can contain a swe:DataArray, swe:DataRecord, swe:Quantity, etc. In the example below a timeseries of 3 grids is encoded in O&M.

- swe:encoding element defines the encoding of the values within the swe:values element by specifying decimal, token and block separators.

- The actual timestamp value pairs contained within the swe:values element.

```
<om:result>
 <swe:DataArray>
  <swe:elementCount>
   <swe:Count>
    <swe:value>3</swe:value>
   </swe:Count>
  </swe:elementCount>
  <swe:elementType name="RootRecord">
   <swe:DataRecord>
    <swe:field name="Timestamp">
     <swe:Time definition="urn:ogc:data:time:iso8601"/>
      </swe:field>
       <swe:field name="value">
        <swe:DataArray>
```

```
          <swe:elementCount>
           <swe:Count>
            <swe:value>77</swe:value>
           </swe:Count>
          </swe:elementCount>
          <swe:elementType name="RowType">
           <swe:DataArray>
            <swe:elementCount>
             <swe:Count>
              <swe:value>95</swe:value>
             </swe:Count>
            </swe:elementCount>
            <swe:elementType name="ValueType">
             <swe:Quantity definition= "urn:ogc:def:property:OGC:O3">
            <swe:uom code="urn:ogc:def:uom:OGC:ppb"/>
           </swe:Quantity>
          </swe:elementType>
         </swe:DataArray>
        </swe:elementType>
       </swe:DataArray>
      </swe:field>
     </swe:DataRecord>
    </swe:elementType>
   <swe:encoding>
    <swe:TextBlock blockSeparator="@" decimalSeparator="." tokenSeparator="
"/>
   </swe:encoding>
   <swe:values>1975-01-01T00:00:00+0100 0.0 0.0 0.0 0. .... swe:values>
  </swe:DataArray>
</om:result>
```

3.3.8 Timeseries properties

The following properties are available on retrieved timeseries in addition to those of the corresponding datapoint:

Name	Type, Value (example)	Comment
TimeSeries.VALUE_KEYS	String[] {TimeSeries.VALUE}	ValueKey of the timeseries values
TimeSeries.VALUE_UNITS	String[] {“mm”}	Units matching the valueKeys
TimeSeries. VALUE_OBSERVED_PROPERTY _URNS	String[] {“urn:ogc:def:property :OGC:1.0:precipitation”}	URN of observed property matching valueKeys
TimeSeries.VALUE_TYPES	String[]	Not set in the moment. See VALUE_CLASSES
TimeSeries. VALUE_JAVA_CLASS_NAMES	String[] {java.lang.Float} or {java.lang.Float[][]}	Java-Type of values. Float if data for a point is requested, Flost[][] for an

		area.
PropertyNames. SPATIAL_RESOLUTION	Float[] {1} of {95,77}	Actual value depends on request. {1} if data for a point wher requested, else size of grid
PropertyNames. TEMPORAL_RESOLUTION	Sting	Time between values (if known) Not available in the moment.
TimeSeries. DESCRIPTION_KEYS	String[]	PropertyNames of properties holding descriptions matching valueKeys. Not set.
PropertyNames. DESCRIPTION	String	Human readable description of the timeseries.
PropertyNames. COORDINATE_SYSTEM	String “EPSG:3423”	Coordinate system for GEOMETRY
Timeseries.GEOMETRY	Envelope, e.g. [14.17 : 48.18, 14.17 : 48.18]	Bounding box.
TimeSeries. AVAILABLE_DATA_MIN	TimeStamp	TimeStamp of first available value
TimeSeries. AVAILABLE_DATA_MAX	TimeStamp	TimeStamp of last available value

The data are organized as follows:

valueKey	**Type**	**Description**
Timeseries.VALUE_KEY	One float value of one 2D grid of floats	Available in the property values above.

References

[SPEC-A] SUDPLAN specific OGC services - abstract spec
Will be available from SUDPLAN.EU.