Open Geospatial Consortium Inc.

Date: 2010-06-30

Reference number of this document: OGC 10-134

Category: OpenGIS® Discussion Paper

Editors: Arne Broering, Stefan Below

OpenGIS® Sensor Interface Descriptors

Warning

This document is not an OGC Standard. This document is an OGC Discussion Paper and is therefore not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, an OGC Discussion Paper should not be referenced as required or mandatory technology in procurements.

Document type: OpenGIS® Discussion Paper
Document subtype: N.A.
Document stage: Approved for public release
Document language: English

Contents

Page

Figures

Page

Tables

Page

i. Preface

This document presents the Sensor Interface Descriptor (SID) schema that enables the declarative description of sensor interfaces, including the definition of the communication protocol, sensor commands, processing steps and metadata association. This schema is designed as a profile and extension of SensorML. Based on this schema, SID interpreters can be implemented, independently of particular sensor technology, which are able to translate between sensor protocol and SWE protocols. They establish the connection to a sensor and are able to communicate with it by using the sensor protocol definition of the SID. SID instances for particular sensor types can be reused in different scenarios and can be shared among user communities. The ability of an SID interpreter to connect sensors and Sensor Web services in an ad hoc manner based on the sensor's SID instance is a next step towards realizing sensor plug & play within the Sensor Web.

ii. Document terms and definitions

This document uses the standard terms defined in Subclause 5.3 of [OGC 05-008], which is based on the ISO/IEC Directives, Part 2. Rules for the structure and drafting of International Standards. In particular, the word "shall" (not "must") is the verb form used to indicate a requirement to be strictly followed to conform to this specification.

iii. Submitting organizations

The following organizations submitted this document to the Open Geospatial Consortium Inc.

52° North Initiative for Geospatial Open Source Software GmbH

Institute for Geoinformatics (IfGI) of the University of Muenster

iv. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Name	Organization
Stefan Below	IfGI
Arne Broering	52° North
Felix Bache	52° North
Thomas Everding	IfGI

v. Revision history

Date	Release	Editor	Primary clauses modified	Description
23.03.2010	0.0.1	Broering	Throughout	Initial setup
02.06.2010	0.0.1	Broering	Throughout	Addition of examples and Annexes
05.06.2010	0.0.1	Broering	Throughout	Major polishing
08.06.2010	0.0.l	Everding	Throughout	Review and comments
09.06.2010	0.0.1	Bache	Outlook	Description of graphical SID creator added
09.06.2010	0.0.1	Broering	Throughout	Incorporation of comments
09.06.2010	0.0.1	Below	Throughout	Review and comments

vi. Changes to the OGC Abstract Specification

The OpenGIS® Abstract Specification does not require changes to accommodate the technical contents of this document.

vii. Future work

The developed approach has been tested under "lab conditions" with different sensor types. Next, it will be applied in real-world scenarios to demonstrate its benefits in sensor asset management. It is anticipated that the design will evolve as new issues and opportunities arise. The gained experimental feedback will be included in the next version of the SID schema.

Foreword

This document presents the concept of Sensor Interface Descriptors (SID). SID is a schema that enables the declarative description of sensor interfaces, including the definition of the communication protocol, sensor commands, processing steps and metadata association. The schema is designed as a profile and extension of the OGC SensorML (SML) schema.

This document includes 3 annexes; Annexes A and B are normative, and Annex C is informative.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium Inc. shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Introduction

The Sensor Web Enablement (SWE) initiative of the Open Geospatial Consortium (OGC) defines standards for Web Service interfaces and data encodings that encapsulate sensors for web-based discovery, tasking and access. SWE has been applied in a multitude of projects, demonstrating its suitability in real world scenarios. However, there is still a fundamental challenge to be tackled. While SWE enables interoperability with the upper application layer, the connection between SWE and the underlying sensor layer and its heterogeneous protocols is not yet sufficiently described.

This work addresses this challenge and presents a declarative schema for *Sensor Interface Descriptors* (SID) based on OGC's SensorML standard. A SID for a particular sensor enables a SID interpreter to translate between the communication protocol of the sensor and the Sensor Web.

The SID concept is the basis for achieving the long-term vision of true sensor plug & play within the Sensor Web and will finally make sensors on-the-fly available on the Sensor Web.

OpenGIS® Sensor Interface Descriptors

1 Scope

This work presents the concept of Sensor Interface Descriptors (SID), a schema which enables the declarative description of sensor interfaces, including the definition of the communication protocol, sensor commands, processing steps and metadata association. This schema is designed as a profile and extension of SensorML. Based on this schema, SID interpreters can be built which are able to translate between sensor protocol and SWE protocols. Such interpreters can be built independently of particular sensor technology. They establish the connection to a sensor and are able to communicate with it by using the explicit sensor protocol definition of the SID. SID instances for particular sensor types can be reused in different scenarios and can be shared among user communities. The SID concept is a next step towards the long term vision of realizing sensor plug & play within the Sensor Web.

2 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

[OGC 07-000] *OGC™ Sensor Model Language (SensorML) Standard*, Version 1.0

[OGC 07-122r2] *OGC™ SensorML Encoding Standard v 1.0 Schema Corregendum 1 (1.01)*, Version 1.0.1

[OGC 07-022r1]: *Observations and Measurements - Part 1 - Observation schema*, Version 1.0.

[OGC 07-165] *OGC® Sensor Web Enablement: Overview And High Level Architecture*. Version 3.

[OGC 06-009r6] *OpenGIS Sensor Observation Service (SOS)*, Version 1.0.

[OGC 06-028r3] *OGC Sensor Alert Service Candidate Implementation Specification*, Version 1.0.

[OGC 07-014r3] *OpenGIS Sensor Planning Service*, Version 1.0.

[OGC 08-133] *OpenGIS® Sensor Event Service Implementation Specification*, Version 0.0.1.

In addition to this document, this specification includes several normative XML Schema Document files as specified in Annexes A and B.

3 Terms and definitions

For the purposes of this standard, the definitions specified in Clause 4 of the OWS Common Implementation Specification [OGC 05-008] and the SensorML Implementation Specification [OGC 07-000] shall apply.

4 Conventions

4.1 Abbreviated terms

Some more frequently used abbreviated terms:

API	Application Program Interface
GML	Geography Markup Language
HTTP	HyperText Transfer Protocol
ISO	International Organization for Standardization
OGC	Open Geospatial Consortium
OWS	OGC Open Web Services
O&M	Observations and Measurements
SAS	Sensor Alert Service
SensorML	Sensor Model Language
SES	Sensor Event Service
SID	Sensor Interface Descriptor
SOAP	Simple Object Access Protocol
SOS	Sensor Observation Service
SPS	Sensor Planning Service
SWE	Sensor Web Enablement
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name

XML eXtensible Markup Language

4.2 UML notation

Some of the diagrams in this document are presented using the Unified Modeling Language (UML) static structure diagram. The UML notations used in this document are described in Figure 1, below.

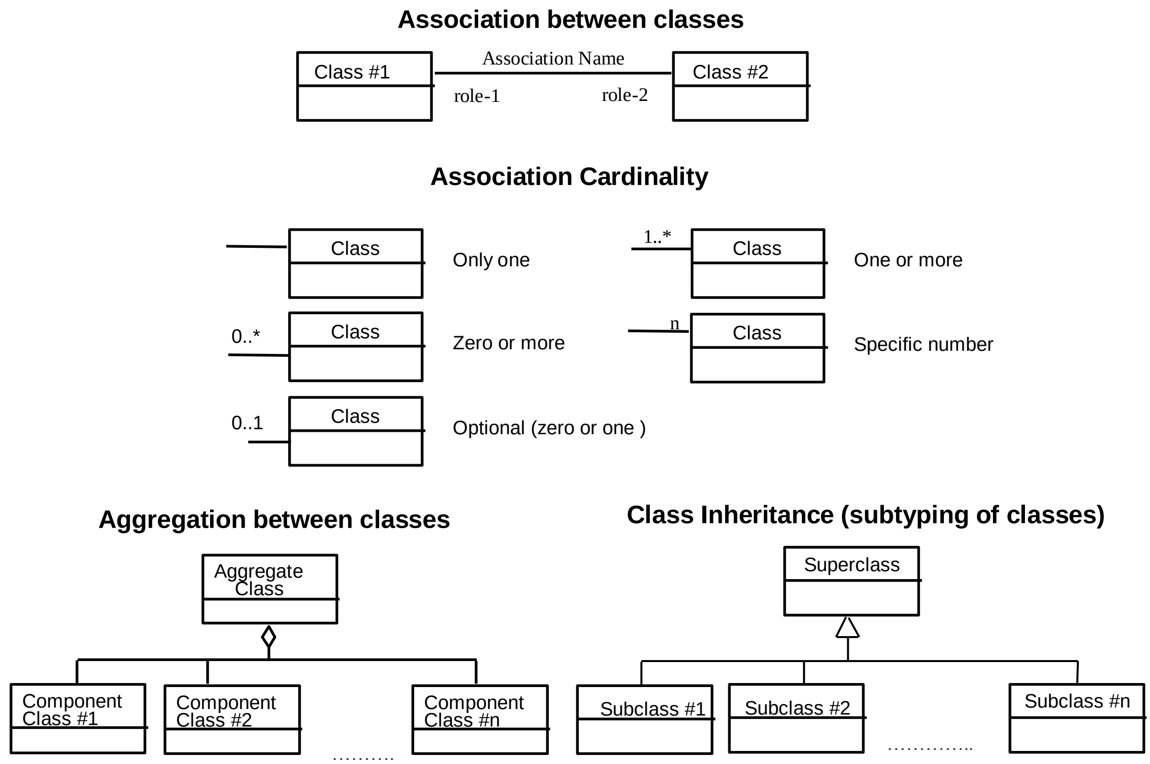

Figure 1 - UML notations

In these UML class diagrams, the class boxes with a light background are the primary classes being shown in this diagram, often the classes from one UML package. The class boxes with a gray background are other classes used by these primary classes, usually classes from other packages.

5 Introduction to Sensor Web Enablement

The Sensor Web Enablement (SWE) [1] initiative of the Open Geospatial Consortium (OGC) defines standards which can be used as building blocks of the Sensor Web. The SWE framework decouples sensor data from the way they are collected and makes them available over the web through standardized formats and interfaces. The main Web Services of the SWE framework are the *Sensor Observation Service* (SOS), the *Sensor Planning Service* (SPS), the *Sensor Alert Service* (SAS), and the *Sensor Event Service* (SES). The SOS [2] is designed for accessing real time as well as historic sensor data, and sensor metadata descriptions. While the SOS follows the pull-based communication paradigm, SAS [3] and SES [4] are capable of pushing sensor data to subscribers. To control and task sensors the SPS [5] can be used. A common application of SPS is to define simple sensor parameters such as the sampling rate but also more complex tasks such as mission planning of satellite systems.

Apart from these Web Service specifications, SWE incorporates an information model for observed sensor data, the *Observations& Measurements* (O&M) [6] standard, as well as an information model for the description of sensors, the *Sensor Model Language* (SensorML) [7].

SensorML specifies a generic model and XML encoding for sensor related processes such as measurement procedures or post processing procedures. Physical as well as logical sensors are modeled as processes. The functional model of a process can be described in detail, including its identification, classification, inputs, outputs, parameters, as well as characteristics such as temporal availability or spatial description. Processes can be composed by process chains.

O&M defines a model and encoding for *observations*. An observation has a result (e.g. 0,7 mSv/a) which is an estimated value of an *observed property* (e.g. radiation), a particular characteristic of a *feature of interest* (e.g. the city center of Muenster). The result value is generated by a *procedure*, in general a sensor (e.g. a radiation detector) described in SensorML. These four central components are linked within SWE concepts.

5.1 The Gap between Sensor Web and Sensor Layer

In recent years, the SWE standards have been applied in various projects (e.g. [8, 9, 10, 11]) showing their practicability and suitability in real world scenarios. However, there is still an essential challenge to be tackled. As pointed out by [12], there is a gap of interoperability between the Sensor Web layer, the architectural level of the SWE services, and the sensor layer (see Figure 2). SWE defines service interfaces from an application oriented perspective. The connection of a SWE service to sensors is not sufficiently specified. Although, SAS and SOS allow the upload of sensor data, the utilization of the according operations still requires reformatting of the sensor protocol to the SWE protocol. A sensor itself is usually not able to upload its data directly to a SWE service, since its bandwidth and processing power are typically limited and the SWE protocol is rather complex and verbose. Most obvious is the interoperability gap at the SPS. It is not defined by the specification how an SPS transforms a retrieved sensor task to a command of the sensor protocol.

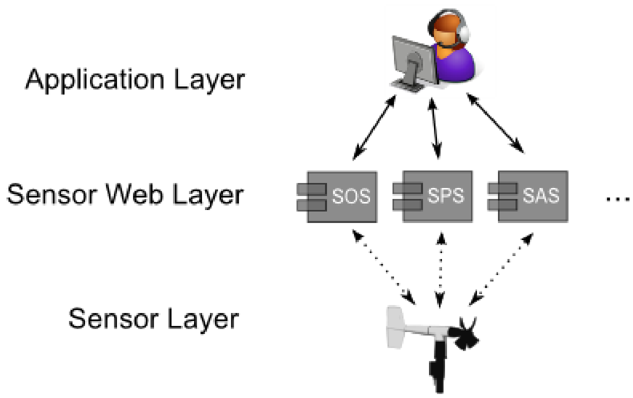

Figure 2 - Sensor Infrastructure Stack

Today, the connection between sensors and SWE services is usually established by manually adapting the internals of the SWE service implementation to the specific sensor type. Such adaptations have to be built for each pair of service implementation and sensor type which leads to extensive efforts in developing large-scale sensor network systems [13].

5.2 Approach

This work addresses the identified interoperability gap with the concept of Sensor Interface Descriptors (SID). Chapter 6 presents a schema for SIDs which enables the declarative description of sensor interfaces, including the definition of the communication protocol, sensor commands, processing steps and metadata association. The schema is designed as a profile and extension of the SensorML standard. The SID schema is illustrated by examples which are excerpts of an SID instance for a radiation detector of the German Federal Office for Radiation Protection. The complete SensorML description of this radiation detector is presented in Annex C.

Based on this schema, SID interpreters can be built which are able to translate between sensor protocol and Sensor Web protocols and hence close the described interoperability gap. Such interpreters for SID instances can be built independently of particular sensor technology. They establish the connection to a sensor and are able to communicate with it by using the explicit sensor protocol definition of the SID. Chapter 7 outlines the implementation of a generic SID interpreter which has been developed in context of this work. It transfers data, retrieved from a sensor, to a Sensor Observation Service and transforms tasks, submitted to a Sensor Planning Service, to commands which are then forwarded to a sensor.

SID instances for particular sensor types can be reused in different scenarios and can be shared among user communities. The ability of an SID interpreter to connect sensors and Sensor Web services in an ad hoc manner based on the sensor's SID instance is a next step towards realizing ***sensor plug & play*** within the Sensor Web.

6 Sensor Interface Descriptors

This Chapter presents the SID concept and schema. First, an overview on the usage of SIDs in a SWE deployment is given (Section 6.1). Next, the requirement for encapsulation of the SID within the SensorML document is emphasized (Section 6.2). The data flow between sensor and SWE service through the SID is described in Section 6.3. Then, the different aspects of the SID schema are described. Section 6.4 outlines how the basic addressing parameters of a physical connection to the sensor are specified. After establishing the physical connection, a definition of the raw sensor protocol is needed, which is illustrated in Section 6.5. With its definition, the sensor protocol can be interpreted and further processed, as described in Section 6.6. Before retrieved, interpreted and processed sensor data can be forwarded to SWE services, certain observation metadata has to be added, which is outlined in Section 6.7. To enable the tasking of sensors, the commands accepted by the sensor interface are described in Section 6.8.

6.1 Overview on SID Utilization

To give an overview of how to utilize an SID Interpreter and SID instances in real world scenarios, Figure 3 shows the deployment of a SWE infrastructure including the usage of SIDs. A sensor communicates with a data acquisition system in its specific sensor protocol over a transmission technology such as ISDN or GSM. This sensor can also act as a sensor gateway (or “network sink”) so that other nodes of a (possibly mobile) sensor network communicate with it. The SID interpreter runs on the data acquisition system and uses SID instances for the different sensors of the sensor network to translate between the sensor specific protocol and the SWE protocols. The interpreter is responsible to register a sensor at a SWE service and to upload sensor data to an SOS, SAS, or SES. Also, it is responsible for the opposite communication direction and forwards tasks received by an SPS to a sensor.

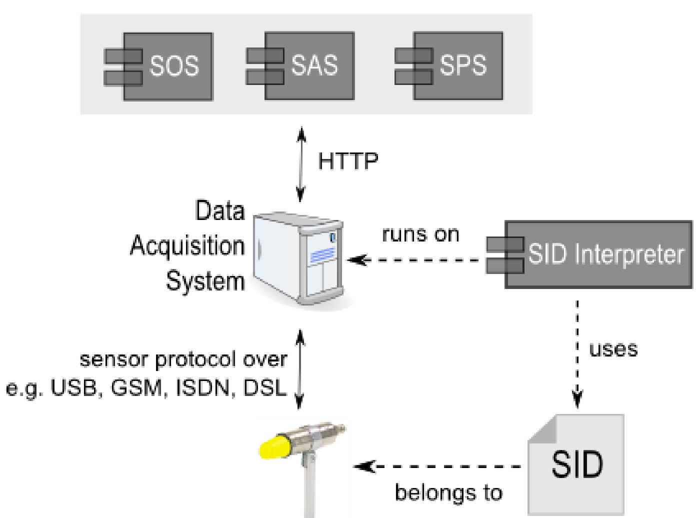

Figure 3 - Usage of SID in SWE deployment

6.2 Encapsulation of the SID

The SID is strictly encapsulated within the SensorML structure. This way, it can be easily replaced or removed. This characteristic is important since the SensorML document has to be usable in situations where the detailed interface description is not of interest or shall not be publicly accessible. Further, an encapsulation of the SID makes it possible to reuse it in different use cases for sensors with same interfaces. The ability of reusing SIDs also enables the installation of repositories to share SIDs for particular sensor types among user communities.

For these reasons, the approach developed here, encapsulates all SID specific information within the *sml:interface* element of an *sml:System*. The *interface* element contains a stack of layers (Figure 4), aligned with the Open System Interconnection (OSI) reference model [14]. In contrast to the OSI model, SensorML does not further define how to use these layers. The SID schema makes use of this layer stack and concretes its usage to describe the sensor interface.

If a sensor has more than one interface (e.g. an Ethernet interface for the data output and an RS-232 interface for sensor programming) the SensorML document can list more than one interfaces containing separate SIDs.

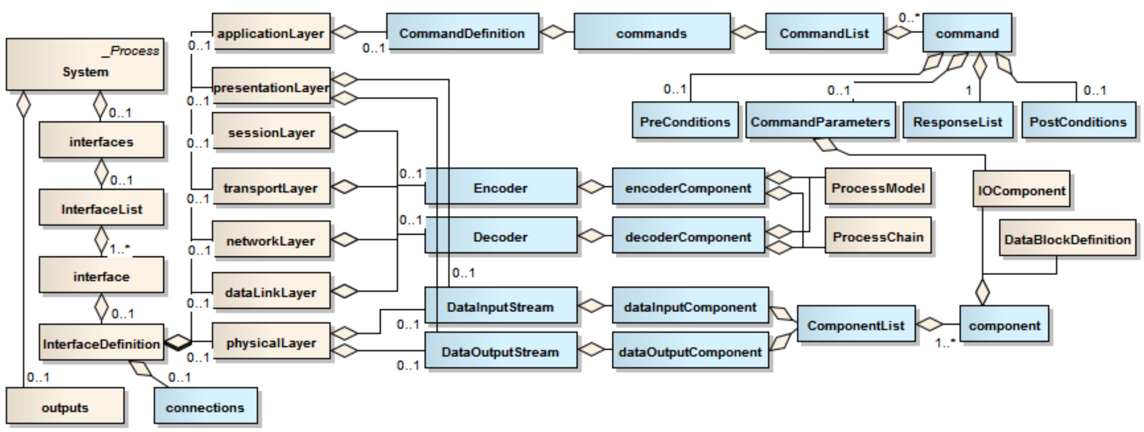

Figure 4 - Overview of SID schema included in SensorML. Elements added by the SID schema are colored in blue.

6.3 Definition of Data Flow

The description of the data flow between the components of the different layers is essential for the SID schema. Figure 5 illustrates the data flow between sensor and SWE through the SensorML document and the SID. To reflect this data flow within the SID, the *sml:connections* element, which is also associated with the *sml:System*, is reused. The *sml:connections* element is associated with the *InterfaceDefinition* (Figure 4). This allows the description of the data flow within the SID internally, and ensures the required encapsulation of the SID.

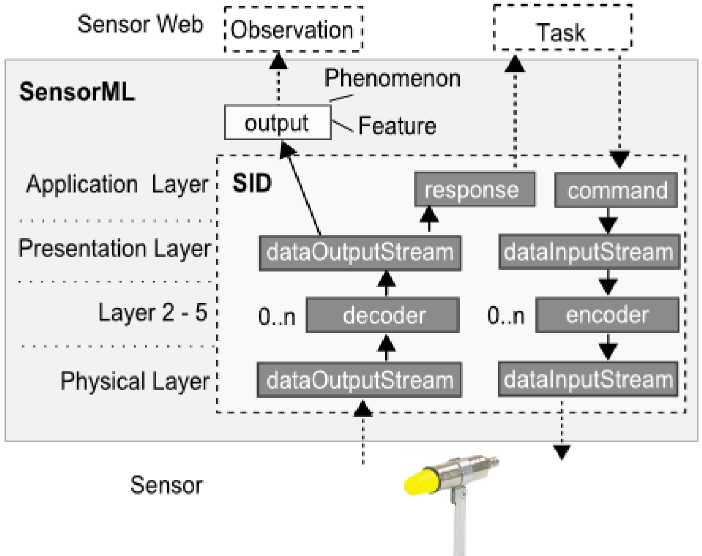

Figure 5 - Data flow between sensor and SWE through the SID

The arrows in Figure 5 symbolize the data flow through the SID. They can be represented by *sml:Link* elements in the *sml:connections* element associated with the *InterfaceDefinition*. Listing 1 shows two examples of such SID internal links. This first link defines a data output component of the *physicalLayer* (the time tag of the incoming sensor data) as the input of a decoder component (a date conversion process) located on the *networkLayer*. The second link points the output of the decoder process to a data output component on the *presentationLayer*. In the SID external *sml:connections* element of the *sml:System*, this output component of the *presentationLayer* is then linked to an output of the sensor system.

Listing 1 - Example of SID internal connection

```
<sml:connections>
  <sml:ConnectionList>

    <sml:connection>
      <sml:Link type="urn:ogc:def:link:OGC:sid">
        <sml:source ref="physicalLayer/dataOutputComponents/M01/data/timeTag"/>
        <sml:destination ref="networkLayer/decoderComponent/inputs/date01"/>
      </sml:Link>
    </sml:connection>

    <sml:connection>
      <sml:Link type="urn:ogc:def:link:OGC:sid">
        <sml:source ref="networkLayer/decoderComponent/outputs/date01"/>
        <sml:destination
ref="presentationLayer/dataOutputComponents/odl_basis_01/samplingTime"/>
      </sml:Link>
    </sml:connection>

    :
```

```
  </sml:ConnectionList>
<sml:connections>
```

6.4 Definition of Addressing Parameters

The addressing parameters (e.g. port and baud rate of a serial connection) are the basis for establishing a physical connection to the sensor. This physical connection is established through the operating system which runs the SID interpreter. The addressing parameters are stored externally in a document accompanying the SID, since the SID can be published publicly (e.g. via a SWE service) and the addressing parameters are security relevant.

Listing 2 - Definition of addressing parameters

```
<sml:interface name="mws3" xlink:role="urn:ogc:def:connection:OGC:serial">
  <sml:InterfaceDefinition>
  :
```

The *xlink:role* attribute of the *sml:interface* is used to specify the type of the connection. Details of the connection for a particular interface, which is identified by the *name* attribute, are stated in an external file whose structure is SID interpreter dependent. The type of connection is specified by using a Unified Resource Name (URN) pointing to globally defined semantics. The currently allowed URNs and their definitions are listed in Table 1.

Table 1 — List of URNs for connection specification

Names	Definition
urn:ogc:def:connection:OGC:serial	Serial connection
urn:ogc:def:connection:OGC:tcpIP	TCP/IP connection
urn:ogc:def:connection:OGC:http	HTTP connection
urn:ogc:def:connection:OGC:udpIP	UDP connection
urn:ogc:def:connection:OGC:file	File system

The usage of the *xlink:role* element to define the connection is formally specified as Schematron rules (see Annex B.5). Those rules which are part of the SID schema can be considered as a profile of the SensorML schema.

6.5 Definition of Sensor Protocol

For the declarative definition of a sensor protocol, the exact definition of the raw data streams exchanged between sensor and data acquisition system is essential. The structure of this raw data is described within the *physicalLayer* element. As shown in Figure 4 and Figure 6 new elements for the data input and data output stream are attached to the element. The two elements are necessary to support duplex communication with sensors.

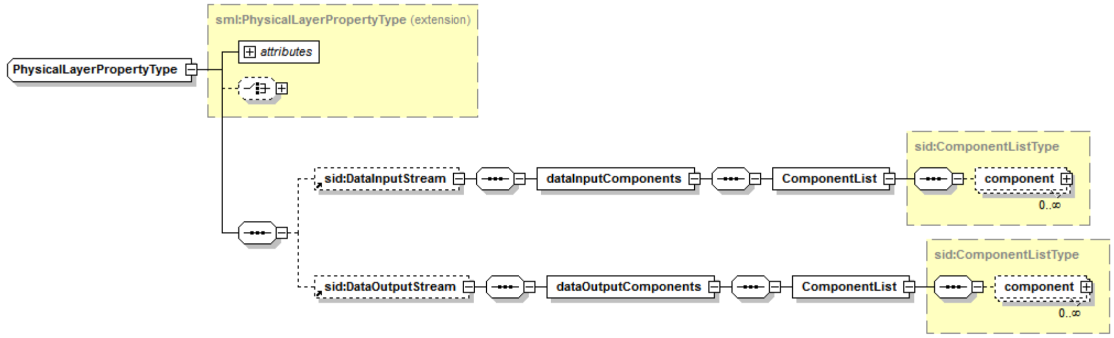

Figure 6 - SID extension of physical layer

Figure 7 shows the structure of the *sid:ComponentListType* which is used to model the incoming or outgoing data stream. The data can be described by using the *sml:IOComponentType* or the *swe:DataBlockDefinition*. The latter one also allows the specification of an encoding.

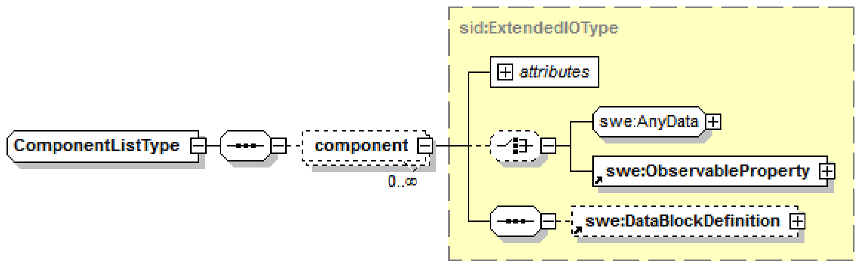

Figure 7 - SID ComponentListType

Listing 3 shows an example of a data stream output of an MWS3 sensor system[1]. Such sensor systems are used in the sensor network of the German Federal Office for Radiation Protection. Besides a radiation detector, an MWS3 carries sensors to measure atmospheric phenomena such as temperature and humidity. The gateway of the MWS3 communicates either via ISDN, GSM, or DSL with a base station, the data acquisition system.

Listing 3 - Example data output stream coming from an MWS3 sensor

```
Station|1275482685|33UUU 932 592|10530Q|#
Status|1275482686|2|43|72|0|#
M01|1275482698|147.0|150.0|23.0|16.3|#
:
```

Listing 4 describes the input and output of an MWS3. The *DataRecord* element of the *component* named '*M01*' defines the structure and meaning of each token of a single data block within the incoming data stream of measurement data. Data blocks are separated by the '#' sign as defined in the *encoding* element. The field of the data record which is annotated with the URN *urn:ogc:def:encoding:OGC:assertedValue* identifies the data

[1] http://de.wikipedia.org/wiki/MWS3-Messwertsender

block of the data stream to which the structure definition refers. In the example of Listing 4, the data record of the *component* named '*M01*' defines the structure for those data blocks where the first token has the value 'M01'.

The *component* of the *DataInputStream* named '*samplingrateCommand*' represents the data structure which is sent to an MWS3 sensor as a command to set its sampling rate. The detailed structure of commands is intentionally not defined on the *physicalLayer* but instead on the *applicationLayer* (Section 6.8). Only on the *applicationLayer*, a command consists of several subcomponents. On the layers below the *applicationLayer*, the command is considered as one aggregated data set. The reason for this is that the processes (e.g. character escaping or checksum validation processes), which are defined on the layers below the *applicationLayer*, need to be executed on the command as one data set.

The *component* of Listing 4 named '*ackResponse*' defines the data structure returned by an MWS3 sensor as an acknowledgement response to the command which sets the sensor's sampling rate.

Listing 4 - Description of incoming and outgoing MWS3 data stream

```
<sml:physicalLayer>

  <!-- incoming data stream: -->
  <sid:DataInputStream>
    <dataInputComponents>
      <ComponentList>
        <component name="samplingrateCommand">
          <swe:DataRecord>
            <swe:field name="command"/>
          </swe:DataRecord>
        </component>
      </ComponentList>
    </dataInputComponents>
  </sid:DataInputStream>

  <!-- outgoing data stream: -->
  <sid:DataOutputStream>
    <dataOutputComponents>
      <ComponentList>

        <!-- responses to sensor commands: -->
        <component name="ackResponse">
          <swe:DataBlockDefinition>
            <swe:components name="data">
              <swe:DataRecord>
                <swe:field name=”ack”
xlink:role="urn:ogc:def:encoding:OGC:assertedValue">
                  <swe:Text>
                    <swe:value>0x06</swe:value>
                  </swe:Text>
                </swe:field>
              </swe:DataRecord>
            </swe:components>
            <swe:encoding>
              <swe:BinaryBlock byteEncoding="hex" byteOrder="bigEndian">
                <swe:member>
                  <swe:Block ref="data/ack" byteLength="1"/>
                </swe:member>
```

```
              </swe:BinaryBlock>
            </swe:encoding>
          </swe:DataBlockDefinition>
        </component>

        <!-- measured data coming from the sensor: -->
        <component name="M01">
          <swe:DataBlockDefinition>
            <swe:components name="data">
              <swe:DataRecord>
                <swe:field name="id"
xlink:role="urn:ogc:def:encoding:OGC:assertedValue">
                  <swe:Text><swe:value>M01</swe:value></swe:Text>
                </swe:field>
                <swe:field name="timeTag"><swe:Count /></swe:field>
                <swe:field name="imp_hd_resistance"><swe:Count /></swe:field>
                <swe:field name="temp"><swe:Count /></swe:field>
                <swe:field name="humidity"><swe:Count /></swe:field>
              </swe:DataRecord>
            </swe:components>
            <swe:encoding>
              <swe:TextBlock
                decimalSeparator="."
                tokenSeparator="|"
                blockSeparator="#" />
            </swe:encoding>
          </swe:DataBlockDefinition>
        </component>

        :

      </ComponentList>
    </dataOutputComponents>
  </sid:DataOutputStream>
</sml:physicalLayer>
```

6.6 Definition of Protocol Processing

For enabling the definition of processing steps which are necessary to translate between the sensor protocol and the SWE protocol, the *dataLinkLayer*, *networkLayer*, *transportLayer*, and *sessionLayer* are utilized. To allow data processing in both directions, from sensor domain to SWE domain and the other way round, elements for data decoding and encoding are added to each of these layers. Figure 8 shows this extension. Instances of the *Encoder* and *Decoder* elements contain either an *sml:ProcessModel* or *sml:ProcessChain* to define the process. A process model can be used to describe a single non-physical process with its inputs, outputs, parameters and its computational method. A process chain can be used to represent a chain of multiple processes and to encapsulate them as one process.

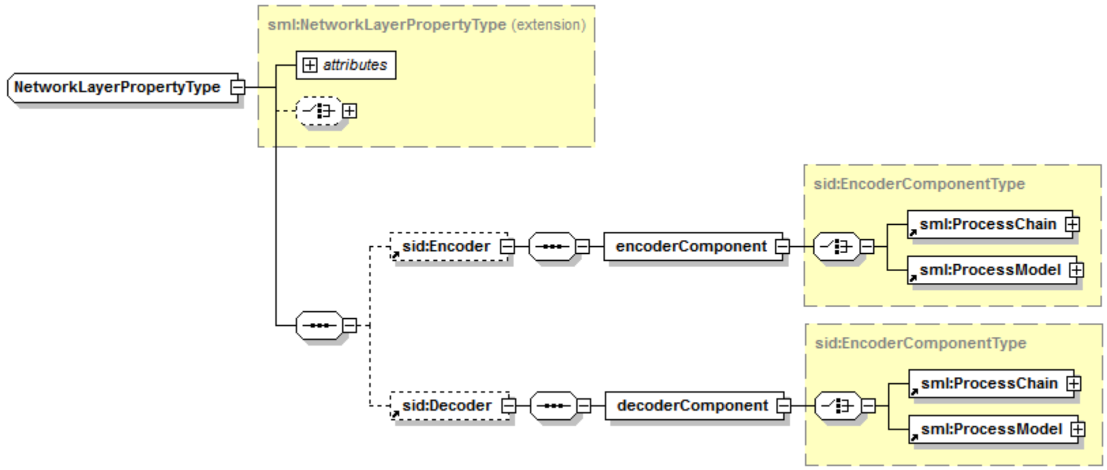

Figure 8 - SID Encoder and Decoder of a *LayerPropertyType[2]

Each of the four layers is optional in an SID. Which processes are described on which layer, depends on the design of a particular SID instance. An interpreter executes the processes defined in these layers sequentially.

Due to their significance for sensor communication, an SID interpreter shall natively support the following four process types: (i) character escaping, (ii) checksum computation and validation, (iii) value interpolation, and (iv) date conversion.

For each of those process types a URN is defined, which can be specified in the *method* property of the process model. Besides those four natively supported processes, other process methods can be incorporated by describing them inline[3] using Content MathML[4].

An example for a typical usage of these four processes to encode a sensor data stream to higher level measurements and the association of the processes to layers of the SID can look like this:

1. *dataLinkLayer* specifies a process for character escaping.
2. *networkLayer* computes a checksum validation.
3. *transportLayer* transforms raw data by applying an interpolation.
4. *sessionLayer* computes a date conversion.

SensorML profiles for each of the four natively supported process types are described in the next subsections.

---

[2] Note: This figure shows the NetworkLayerPropertyType. DataLinkLayerPropertyType, TransportLayerPropertyType and SessionLayerPropertyType are structured in the same way.

[3] Note: The possibility of referencing binary or source code representations of processes is intentionally not supported by the SID schema, since such kinds of process definitions would depend on implementation of a particular SID interpreter.

[4] http://www.w3.org/TR/MathML2

6.6.1 Character Escaping Process Type

In sensor communication, escape characters are used to induce an alternative interpretation of a transmitted character. As seen in the example of Listing 3, the end token of a data set within a data stream is indicated by a particular control character, the '#' sign. In the raw sensor data, this control character is masked by an escape character (e.g. '\'). Which characters are used for escaping is defined by the sensor protocol.

Hence, every SID interpreter shall support a process type for removing and adding escape characters. The process type can be referenced by defining the URN *urn:ogc:def:process:OGC:escCharacter* in the method property. The mandatory properties of the character escaping process type are listed in Table 2. A formal definition of the character escaping process type as Schematron rules can be found in Annex B.1. Listing 5 shows an example usage of the character escaping process.

Table 2 — Mandatory properties for character escaping process type

Element	Definition
sml:inputs	Input data on which character escaping shall be performed.
sml:outputs	Processed output data.
sml:parameters	Details on escape characters as 2-dimensional *swe:DataArray*.

Listing 5 - Example of the character escaping process

```
<sml:ProcessModel>
  <sml:inputs>
    <sml:InputList>
      <sml:input name="data"/>
    </sml:InputList>
  </sml:inputs>
  <sml:outputs>
    <sml:OutputList>
      <sml:output name="data"/>
    </sml:OutputList>
  </sml:outputs>
  <sml:parameters>
    <sml:ParameterList>
      <sml:parameter name="escapeDefinition">
        <swe:DataArray>
          <swe:elementCount/>
          <swe:elementType name="escapeCharacters">
            <swe:DataRecord>
              <swe:field name="charFrom" xlink:role="urn:ogc:def:hex"/>
              <swe:field name="charTo"/>
            </swe:DataRecord>
          </swe:elementType>
          <swe:encoding>
            <swe:BinaryBlock byteEncoding="hex" byteOrder="littleEndian">
              <swe:member>
                <swe:Block ref="escapeCharacters/charFrom" byteLength="1"/>
              </swe:member>
              <swe:member>
                <swe:Block ref="escapeCharacters/charTo" byteLength="2"/>
```

```
              </swe:member>
            </swe:BinaryBlock>
          </swe:encoding>
          <swe:values>0x02 0x05 0x02 0x03 0x05 0x03</swe:values>
        </swe:DataArray>
      </sml:parameter>
    </sml:ParameterList>
  </sml:parameters>
  <sml:method xlink:role="urn:ogc:def:process:OGC:escCharacter"/>
</sml:ProcessModel>
```

6.6.2 Checksum Process Type

For a reliable sensor communication, the ability to **compute** and **validate** checksums is essential. Each SID interpreter shall offer a process type for that purpose. It can be referenced using the URN *urn:ogc:def:process:OGC:1.0:checkSum*.

The most widely used checksum method is the Cyclic Redundancy Check (CRC). However, there is no standard describing how to compute a CRC. Hence, the SID schema supports the parameterized model for the definition of CRC algorithms, the Rocksoft Model [15]. The parameters of this model (e.g. polynomial, and name of the algorithm to be used) are passed along in the parameters element of this process type. The necessary parameters and their meaning are described in Table 3.

The mandatory properties of the process type to **compute** a checksum are listed in Table 4. Listing 6 shows an example usage of the checksum computation process.

The mandatory properties of the process type to **validate** a checksum are listed in Table 5. For validating a checksum, the asserted value of a checksum shall be defined in the *sml:inputs* element of the process.

A formal definition of the checksum process type using Schematron rules can be found in Annex B.2.

Table 3 — Parameters of checksum process type

Names	SWE Type	Definition[5]
name	Text	Name of algorithm to be used.
width	Count	Length of polynomial.
poly	Text	Specification of the polynomial in hex. The first bit is omitted.
init	Count	Initial value of the register at the beginning of the calculation.
refin	Boolean	Condition which has to be fulfilled before the execution can be started.
refout	Boolean	Indicates whether the register is forwarded directly to the XOROUT operation or whether a binary swap is invoked first.
xorout	Text	Hexadecimal value associated with the result of the XOR operation.
check	Text	Validation value of the algorithm for the String “123456789”

Table 4 — Mandatory properties for checksum *computation* process type

Element	Definition
sml:inputs	Data for which the checksum shall be computed.
sml:outputs	Computed checksum.
sml:parameters	See Table 3.

Table 5 — Mandatory properties for checksum *validation* process type

Element	Definition
sml:inputs	1. `data` - Data for which the checksum shall be computed. 2. `checksum` - asserted checksum for given data.
sml:outputs	Returns the input data if the validation is successful.
sml:parameters	See Table 3.

Listing 6 - Example of the checksum *computation* process

```
<sml:ProcessModel>
    <sml:inputs>
        <sml:InputList>
            <sml:input name="data"/>
        </sml:InputList>
    </sml:inputs>
    <sml:outputs>
        <sml:OutputList>
            <sml:output name="checkSum">
                <swe:Text/>
            </sml:output>
        </sml:OutputList>
    </sml:outputs>
    <sml:parameters>
        <sml:ParameterList>
```

[5] Note: A detailed description of these parameters can be found in [15].


```
            <sml:parameter name="type">
                <swe:DataRecord>
                    <swe:field name="crc">
                        <swe:DataRecord>
                            <swe:field name="name">
                                <swe:Text>
                                    <swe:value>CRC-5/USB</swe:value>
                                </swe:Text>
                            </swe:field>
                            <swe:field name="width">
                                <swe:Count>
                                    <swe:value>5</swe:value>
                                </swe:Count>
                            </swe:field>
                            <swe:field name="poly">
                                <swe:Text>
                                    <swe:value>05</swe:value>
                                </swe:Text>
                            </swe:field>
                            <swe:field name="init">
                                <swe:Text>
                                    <swe:value>1F</swe:value>
                                </swe:Text>
                            </swe:field>
                            <swe:field name="refin">
                                <swe:Boolean>
                                    <swe:value>true</swe:value>
                                </swe:Boolean>
                            </swe:field>
                            <swe:field name="refout">
                                <swe:Boolean>
                                    <swe:value>true</swe:value>
                                </swe:Boolean>
                            </swe:field>
                            <swe:field name="xorout">
                                <swe:Text>
                                    <swe:value>1F</swe:value>
                                </swe:Text>
                            </swe:field>
                            <swe:field name="check">
                                <swe:Text>
                                    <swe:value>19</swe:value>
                                </swe:Text>
                            </swe:field>
                        </swe:DataRecord>
                    </swe:field>
                </swe:DataRecord>
            </sml:parameter>
        </sml:ParameterList>
    </sml:parameters>
    <sml:method xlink:role="urn:ogc:def:process:OGC:1.0:checkSum"/>
</sml:ProcessModel>
```

6.6.3 Interpolation Process Type

Interpolations need to be computed to transform raw sensor data to observations, to compute calibrations, or to correct measurements. For example, an electric current returned by a detector needs to be transformed to an actual measurement value of a particular phenomenon (e.g. radiation or temperature).

This process type can be referenced by *urn:ogc:def:process:OGC:interpolation:cubic* or *urn:ogc:def:process:OGC:interpolation:linear*, depending on the interpolation method which shall be applied. The mandatory properties of this process type are listed in Table 6. A formal definition of this process type using Schematron rules can be found in Annex B.3. Listing 7 shows an example usage of the interpolation process.

Parameters of this process type are the two axes x and y between which the interpolation shall be performed. The actual values of the coordinate tuples defining the curve are usually defined outside of the SID. This is done, since the definition of the interpolation curve is depending on the sensor deployment, not the sensor interface. For example, a weather station might use different curve definitions to transform raw data to observations depending on the set up how and where it is installed. Listing 8 shows an example of an *sml:Component* which can be listed in the *sml:components* element of an *sml:System* and is hence outside of the SID. This *sml:Component* contains the parameterization of a curve for the two axes 'resistance' and 'radiation'. The *fields* representing these two axes can be linked to the two axes of the interpolation process by specifying according links in the *sml:connections* element of the *sml:System* (see Annex C). This way, the interpolation process is externally parameterized.

Table 6 — Mandatory properties for interpolation process type

Element	Definition
sml:inputs	Value which shall be interpolated.
sml:outputs	Interpolated value.
sml:parameters	The two axes between which the interpolation shall be performed. The actual coordinate tuples defining the curve are usually defined outside of the SID.

Listing 7 - Example of a date interpolation process

```
<sml:ProcessModel>
    <sml:inputs>
        <sml:InputList>
            <sml:input name="value">
                <swe:Quantity/>
            </sml:input>
        </sml:InputList>
    </sml:inputs>
    <sml:outputs>
        <sml:OutputList>
            <sml:output name="result">
                <swe:Quantity/>
            </sml:output>
        </sml:OutputList>
    </sml:outputs>
    <sml:parameters>
        <sml:ParameterList>
            <sml:parameter name="settings">
                <swe:DataArray>
                    <swe:elementCount/>
                    <swe:elementType name="values">
                        <swe:DataRecord>
                            <swe:field name="x">
```



```
                                <swe:Quantity/>
                            </swe:field>
                            <swe:field name="y">
                                <swe:Quantity/>
                            </swe:field>
                        </swe:DataRecord>
                    </swe:elementType>
                </swe:DataArray>
            </sml:parameter>
        </sml:ParameterList>
    </sml:parameters>>
    <sml:method xlink:role="urn:ogc:def:process:OGC:1.0:interpolation:cubic"/>
</sml:ProcessModel>
```

Listing 8 - Example parameterization of a date interpolation process

```
<sml:component name="radiometer">
  <sml:Component>
    <sml:parameters>
      <sml:ParameterList>
        <sml:parameter name="steadyStateCurve">
          <swe:Curve>

            <swe:elementCount>
              <swe:Count><swe:value>2</swe:value></swe:Count>
            </swe:elementCount>

            <swe:elementType>
              <swe:SimpleDataRecord>
                <swe:field name="resistance">
                  <swe:Quantity
definition="urn:ogc:def:property:OGC:resistance"/>
                </swe:field>
                <swe:field name="radiation">
                  <swe:Quantity
definition="urn:ogc:def:property:OGC:radiation"/>
                </swe:field>
              </swe:SimpleDataRecord>
            </swe:elementType>

            <swe:encoding>
              <swe:TextBlock
                tokenSeparator   = ","
                decimalSeparator = "."
                blockSeparator   = "; " />
            </swe:encoding>

            <swe:values>-3,-5; 6,7; 8.2,7.5</swe:values>

          </swe:Curve>
        </sml:parameter>
      </sml:ParameterList>
    </sml:parameters>
  </sml:Component>
</sml:component>
```

6.6.4 Date Conversion Process Type

If sensors tag their data with a timestamp, usually a conversion of the sensor time (e.g. milliseconds since Unix Epoch) to another time representation (e.g. ISO 8601) is necessary. This date conversion shall be natively supported by every SID interpreter. In the parameters element of this process type, literals are used to define the input and output time formats (e.g. a 'T' for Unix time in seconds, and 'YYYY' for the years with 4 digits). By specifying the URN *urn:ogc:def:process:OGC:dateConversion* it is indicated to the SID interpreter that this process method shall be applied. The mandatory properties of this process type are listed in Table 7 and the literals usable in date formatting are listed in Table 8. A formal definition of this process using Schematron rules can be found in Annex B.4. Listing 9 shows an example usage of the date conversion process.

Table 7 — Mandatory properties for the date conversion process type

Element	Definition
sml:inputs	Date string which shall be converted.
sml:outputs	Converted date string.
sml:parameters	Definition of date format of input and output using the literals defined in Table 8.

Table 8 — Literals for date conversion[6]

[6] Note: This list of literals for the definition of time strings is influenced by the Java SimpleDateFormat (http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html)

Literal	Definition
G	Epoch (e.g. ‘AD’)
Y	Year
M	Month (e.g. ‘01’ = January)
w	Week of year
W	Week of month
D	Day of year
d	Day of month
F	Day of week
E	Day textual (e.g. ‘Wednesday’)
a	am/pm
H	Hour (0-23)
k	Hour (1-24)
K	Hour am/pm (0-11)
h	Hour am/pm (1-12)
m	Minutes of hour
s	Seconds of minute
S	Milliseconds of second
z	Time zone textual (e.g. ‘GMT’)
Z	Time zone as defined in RFC 822 (e.g. ‘-0800’)
T	Unix timestamp in seconds (number of seconds elapsed since midnight UTC of January 1, 1970)
t	Unix timestamp in milliseconds

Listing 9 - Example of a date conversion process

```
<sml:ProcessModel>
    <sml:inputs>
        <sml:InputList>
            <sml:input name="date">
                <swe:Text />
            </sml:input>
        </sml:InputList>
    </sml:inputs>
    <sml:outputs>
        <sml:OutputList>
            <sml:output name="date">
                <swe:Text />
            </sml:output>
        </sml:OutputList>
    </sml:outputs>
    <sml:parameters>
        <sml:ParameterList>
            <sml:parameter name="dateSettings">
                <swe:DataRecord>
                    <swe:field name="inputFormat">
                        <swe:Text>
                            <swe:value>T</swe:value>
                      </swe:Text>
                    </swe:field>
```



```
                    <swe:field name="outputFormat">
                        <swe:Text>
                            <swe:value>YYYY-MM-dd HH:mm:ss</swe:value>
                        </swe:Text>
                    </swe:field>
                </swe:DataRecord>
            </sml:parameter>
        </sml:ParameterList>
    </sml:parameters>
    <sml:method xlink:role="urn:ogc:def:process:OGC:dateConversion"/>
</sml:ProcessModel>
```

6.7 Definition of Observation Metadata

The data, resulting from the preceding processing steps (Section 6.6), has to be associated with certain metadata, which is part of the O&M model, before it can be forwarded to an SOS. The measured data needs to be associated with units of measure so that an interpretation is possible. Further, the data needs to be linked to the elementary SWE components (Section 5), the observed property and the feature of interest, so that observations of the O&M model can be created and inserted into an SOS.

The association of the data with a unit of measure is done on the *presentationLayer* as shown in Listing 10. The *DataOutputStream* element is used to describe the data coming from the sensor on this layer. For example, the *component* named '*odl_basis_01*' contains the field descriptions of the measured sensor data. The *field* named '*highDoseRadiation*' represents the radiation data measured in millisievert per year ('mSv/a').

Listing 10 - Definition of sensor data on presentation layer

```
<sml:presentationLayer>

  <!-- incoming data stream: -->
  <sid:DataInputStream>
    <dataInputComponents>
      <ComponentList>
        <component name="samplingRateCommand">
          <swe:DataRecord>
            <swe:field name="command"/>
          </swe:DataRecord>
        </component>
      </ComponentList>
    </dataInputComponents>
  </sid:DataInputStream>

  <!-- outgoing data stream: -->
  <sid:DataOutputStream>
    <dataOutputComponents>
      <ComponentList>

        <component name="ackResponse">
          <swe:DataRecord>
            <swe:field name="ack"></swe:field>
          </swe:DataRecord>
        </component>

        <component name="odl_basis_01">
```



```
          <swe:DataRecord>
            <swe:field name="samplingTime">
              <swe:Time>
                <swe:uom code="ISO8601"/>
              </swe:Time>
            </swe:field>
            <swe:field name="highDoseRadiation">
              <swe:Quantity gml:id="highDoseRadiationData">
                <swe:uom code="mSv/a" />
              </swe:Quantity>
            </swe:field>

            :

          </swe:DataRecord>
        </component>

      </ComponentList>
    </dataOutputComponents>
  </sid:DataOutputStream>
</sml:presentationLayer>
```

In the *sml:outputs* element of the *sml:System*, the sensor data is linked to the observed property, and to the feature of interest. Also, the data is linked to an observation offering, an element of the SOS which groups thematically similar observations. The linking to an observation offering is necessary for being able to execute the *InsertObservation* operation of the SOS. As shown in Listing 11, the *gml:metaDataProperty* is used for this linking. The feature of interest and the observed property are referenced by the means of the *xlink:href* attribute. For the observation offering, its service side identifier is given. The *field* element named '*dataSet*' includes the description of the data by specifying the reference to the element with the *gml:id* '*highDoseRadiationData*' (see Listing 10) in the *xlink:href* attribute.

The usage of those elements within the *sml:output* element to define the observation metadata is specified as Schematron rules (see Annex B.6). Those rules which are part of the SID schema can be considered as a profile of the SensorML schema.

The *sml:outputs* element is not part of the SID, since it is not a sub-element of the *InterfaceDefinition* (Figure 4). The information contained in the *sml:output* elements is intentionally kept out of the SID, due to the fact that the linkage of a sensor to a feature of interest, an observed property, and an observation offering is dependent on the particular use case, not the sensor interface. Not including this information into the SID, enables a reusing of the SID in different SWE deployments.

Listing 11 - Definition of observation metadata in output element

```
<sml:outputs>
  <sml:OutputList>

    <sml:output name="highDoseRadiation_output">
      <swe:DataRecord>

        <gml:metaDataProperty>
          <offering>RADIATION_OFFERING</offering>
        </gml:metaDataProperty>
```

```
        <gml:metaDataProperty>
          <featureOfInterest xlink:href="http://myWFS.org/features/Muenster"/>
        </gml:metaDataProperty>

        <gml:metaDataProperty>
          <observedProperty xlink:href="urn:ogc:def:property:OGC:radiation"/>
        </gml:metaDataProperty>

        <swe:field name="dataSet" xlink:href="#highDoseRadiationData" />

      </swe:DataRecord>
    </sml:output>

    :
  <sml:OutputList>
</sml:outputs>
```

6.8 Definition of Sensor Commands

The application layer of the OSI model describes interfaces to access the OSI stack. Compliant to this view, the *applicationLayer* is used here to define the commands accepted by the sensor. These command definitions can be used by an SPS so that it can provide information to clients how to task a sensor. Figure 9 shows the structure of the SID CommandType. The *command* element contains sub-elements to describe the responses of the sensor, the pre- and postconditions for executing the command, as well as the command parameters.

Each command has a mandatory attribute to define its unique *name*. The *timeout* attribute is of type *xs:integer* representing the time in milliseconds until a response of the sensor has to be received. The attribute *repeat* of type *xs:boolean* can be used to define whether a command call has to be repeated before a response is received. The attribute *auto* of type *xs:boolean* defines whether the command call is automatically executed every n seconds. The number of seconds is defined by the *interval* attribute.

The element *sid:CommandParameters* contains the parameters of the command. The *parameter* element is of type *sml:IoComponentPropertyType* and uses a *swe:DataRecord* to list the parameters in *swe:field* elements. The data type of the parameter can be specified using the basic SWE Common types (e.g. *swe:Text* or *swe:Count*). Additionally, the allowed value domain can be restricted using the *swe:constraint* element. The parameters can be further characterized using the URNs defined in Table 9 as value of the *xlink:role*. The order of the command parameters is the same as in the sensor protocol to which the command is mapped by the processes defined in the lower layers (Section 6.6).

Table 9 — List of URNs to characterize command parameters

Names	Definition
urn:ogc:def:command:OGC:name	Command name
urn:ogc:def:command:OGC:value	Constant value in the command definition (e.g. a separator sign)
urn:ogc:def:parameter:OGC:optional	Optional parameter
urn:ogc:def:parameter:OGC:required	Required parameter

Under consideration that a command might have different responses with different meanings, the element *sid:ResponseList* contains the possible responses of the command. The *response* element is of type *sid:IoComponentType* which extends the *sml:IoComponentPropertyType* by adding the *valid* attribute of type *xs:boolean*. This attribute marks whether the response represents a successful or unsuccessful execution of the command.

The element *PostCondition* shall be used to reference commands which are executed after execution of this command. The *PreCondition* element references commands which shall be executed before this command. The sub-elements *assertTrue* and *assertFalse* of *PreCondition* and *PostCondition* shall be used to demand a successful or unsuccessful execution of those referenced commands.

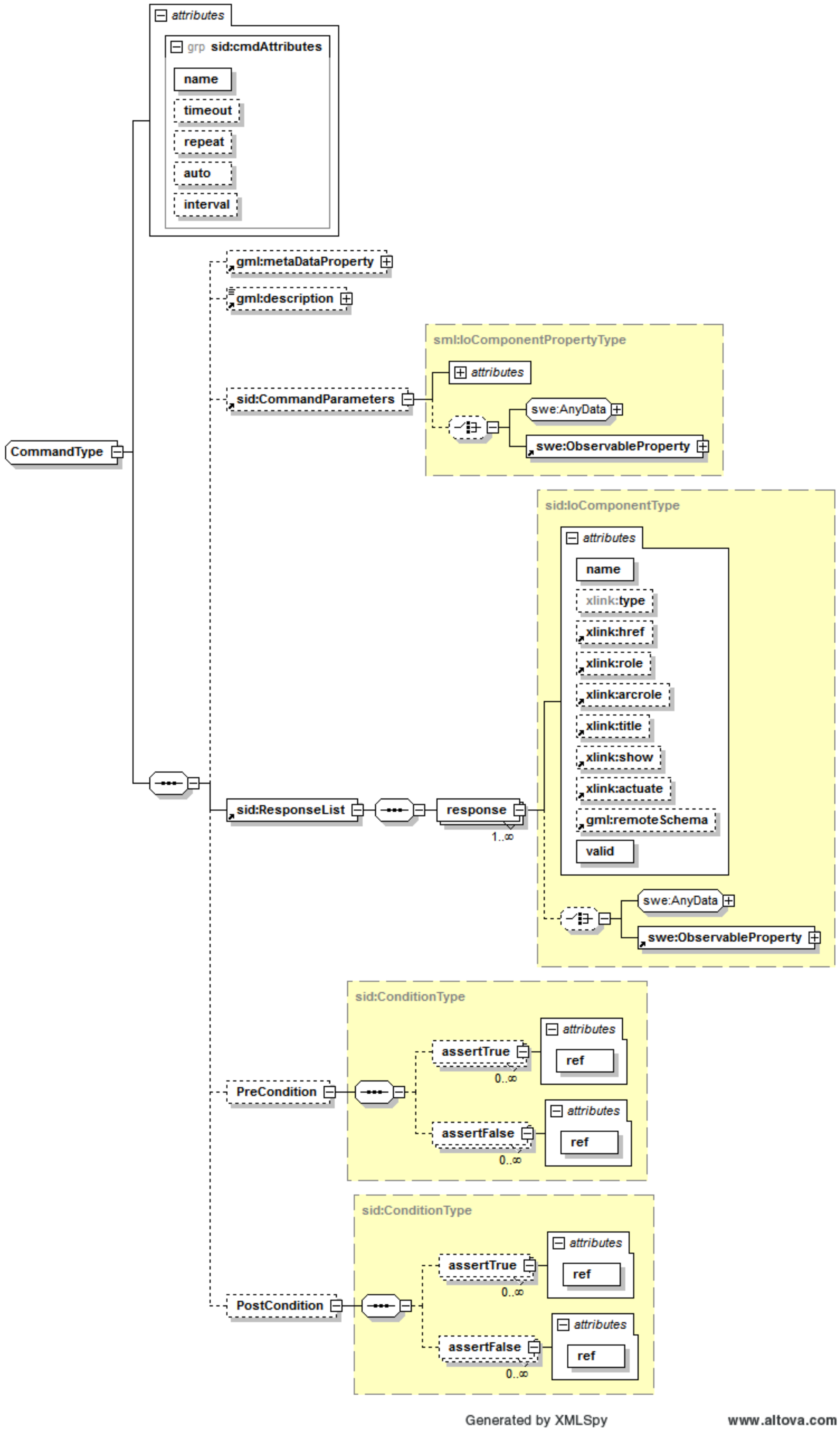

Figure 9 - SID CommandType

Listing 12 shows an example of a *CommandDefinition* of a command which sets the sampling rate of a sensor. The command has four parameters: the first one is the command name and fixed to the value 'SR', the second one is a text representing the sensor ID, the third one is a constant value (‘|’) acting as a separator sign, and the fourth one is the measuring interval, with a minimum value of 5 seconds.

Listing 12 - Example of a command definition to set the sampling rate of a sensor

```
<sml:applicationLayer>
  <sid:CommandDefinition>
    <sid:commands>
      <command name="setSamplingRate">
        <sid:CommandParameters name="setSamplingRateParameters">
          <swe:DataRecord>
            <swe:field name="cmd"
              xlink:role="urn:ogc:def:command:OGC:name">
              <swe:Text>
                <swe:value>SR</swe:value>
              </swe:Text>
            </swe:field>
            <swe:field name="sensorID"
              xlink:role="urn:ogc:def:command:OGC:required">
              <swe:Text/>
            </swe:field>
            <swe:field name="separator"
              xlink:role="urn:ogc:def:command:OGC:value">
              <swe:Text>|<swe:Text>
            </swe:field>
            <swe:field name="interval"
              xlink:role="urn:ogc:def:command:OGC:optional">
              <swe:Quantity>
                <swe:uom code="sec" />
                <swe:constraint>
                  <swe:AllowedValues>
                    <swe:min>5</swe:min>
                  </swe:AllowedValues>
                </swe:constraint>
              </swe:Quantity>
            </swe:field>
            :
          </swe:DataRecord>
        </sid:CommandParameters>
        <sid:ResponseList>
          <response valid="true" name="successfulCommand">
            <swe:DataRecord>
              <swe:field name="ack" />
            </swe:DataRecord>
          </response>
          <response valid="false" name="unsuccessfulCommand">
            <swe:DataRecord>
              <swe:field name="nack" />
            </swe:DataRecord>
          </response>
         </sid:ResponseList>
         <PreCondition>
           <assertTrue ref="initCommand"/>
         </PreCondition>
       </command>
     </sid:CommandList>
   </sid:commands>
  </sid:CommandDefinition>
</sml:applicationLayer>
```


7 Interpreter Implementation

This Chapter outlines a possible design of an SID interpreter[7] as it has been developed by 52° North. The implementation of the SID interpreter is based on the OSGi framework[8] which is extendible by pluggable and loosely coupled *Bundle* components (Figure 10).

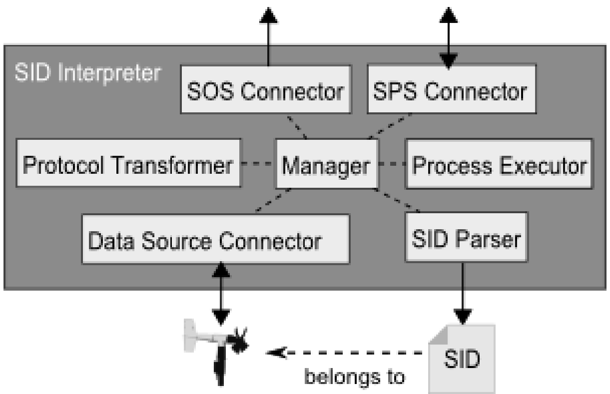

Figure 10 - Design of the SID Interpreter

The implemented components of the 52° North SID interpreter are the following:

- A central *Manager* component controls the workflow.
- The *SID Parser* is used to read in the SID document of the sensor.
- Depending on the specified addressing parameters (Section 6.4), a particular *Data Source Connector* implementation (e.g. for serial connections) is chosen to connect to the sensor.
- Based on the protocol definition of the SID (Section 6.5), the *Protocol Transformer* communicates with the sensor in a bi-directional way.
- The *Process Executor* is able to execute the four native process types (Section 6.6). Also, user-defined MathML processes can be executed by the means of the MathML Solver library[9].
- The *SOS Connector* triggers the SOS operation *RegisterSensor* to add the new sensor to the Sensor Web and executes the *InsertObservation* operation to upload sensor data as observations (Section 6.7) to an SOS.
- The *SPS Connector* forwards the SensorML document and the contained SID to an SPS which uses the sensor command descriptions (Section 6.8) to provide detailed

[7] http://52north.org/sid

[8] http://www.osgi.org

[9] http://sourceforge.net/projects/mathmlsolver

information how to task the sensor. Sensor tasks, submitted to the SPS, are forwarded by the SPS to the SID interpreter and received by the *SPS Connector* component. The tasks are transformed to the sensor protocol, and passed through the *Data Source Connector* to the sensor.

8 Conclusions & Outlook

The concept of SIDs can minimize the efforts of integrating new sensors with the Sensor Web. Once created for a particular sensor type, an SID can be exchanged and reused in different contexts, projects, or user communities. In future, repositories can be set up which facilitate the sharing and the discovery of SID instances. By using an SID instance, the generic SID interpreter then allows an on-the-fly integration of sensors with the Sensor Web by minimizing the administration and deployment efforts. This is significant step towards the vision of sensor plug & play within the Sensor Web.

A further step towards vision will be the incorporation of the SID interpreter into the Sensor Bus[10] architecture [16]. This publish/subscribe architecture (Figure 11) establishes an event-driven communication between SWE services and sensors. Thereby, a Sensor Adapter transforms the sensor protocol to the simple bus message protocol. It is planned to realize one generic Sensor Adapter by making use of the SID interpreter.

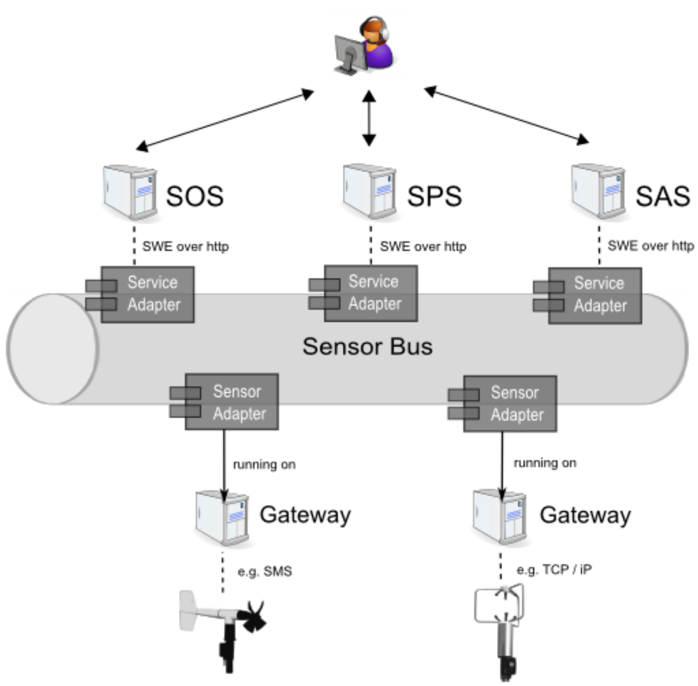

Figure 11 - Sensor Bus concept

---

10 http://52north.org/sensorBus

To support the creation of SIDs, the 52°North Sensor Web Community is currently developing tools and graphical user interfaces. This will lower the entry threshold for sensor network administrators to make their sensors available on the Sensor Web.

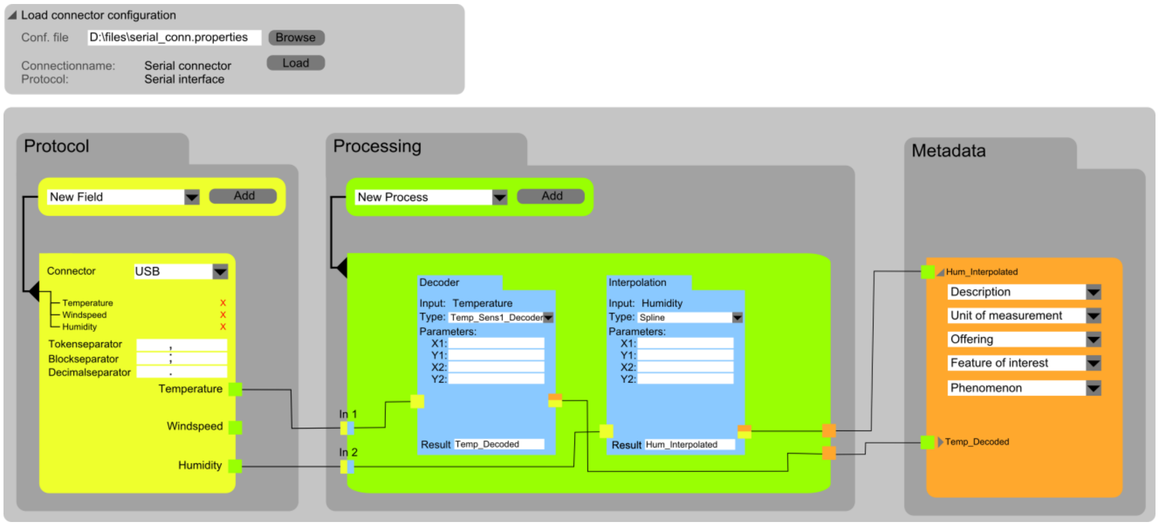

Figure 12 - Draft of SID creator GUI

A first draft of a graphical user interface (GUI) which facilitates the creation of SDIs is shown in Figure 12. This GUI enables a user to create an SID by presenting separate sections for data input, processing and metadata association. Another version of this GUI contains a section for sensor tasking instead of the metadata association. This enables a user to model the data flow from an SPS to the sensor protocol. The data flow can be modeled intuitively by creating connections between the different components in a drag & drop manner.

The protocol section defines the structure of data coming from the sensor. New fields can be added and the format is defined. The user can drag the data fields to the processing section where new processes can be created with their according parameters. The processing outputs are then passed to the last section, where the metadata is added to the processed data.

9 Acknowledgment

This work is financially supported by the project "Flexible and Efficient Integration of Sensors and Sensor Web Services" funded by the European Regional Development Fund (ERDF) for NRW (grant number N 114/2008) of the European Union, as well as the 52° North Sensor Web community (http://52north.org/SensorWeb) which envisions a real time integration of heterogeneous sensors into a coherent information infrastructure.

Annex A
(normative)

XML Schema Documents

In addition to this document, this specification includes several normative XML Schema Documents. These XML Schema Documents are bundled in a zip file with the present document.

The XML Schema Documents developed for SIDs are based on the SensorML XML Schema Documents specified in [OGC 07-000]. However, for being able to extend SensorML in the way described in this specification, it needed to be slightly modified:

1. Each layer associated with the *sml:InterfaceDefinition* has gotten its own type instead of using *sml:LayerPropertyType* for all layers.
2. The *sml:connections* element has been associated with the *sml:InterfaceDefinition*.

The updated schema files are bundled with this document.

The abilities specified in this document use 4 specified XML Schema Documents included in the zip file with this document. These XML Schema Documents are named:

basicElements.xsd

command.xsd

InterfaceDefinition.xsd

sid.xsd

Annex B
(normative)

Schematron Rules

This Annex contains the Schematron definitions which act as a profile of the SensorML schema. First, the Schematron rules of the four processes natively supported by every SID interpreter are listed. Further, Schematron rules for the definition of addressing parameters and for the definition of observation metadata are listed.

B.1 Schematron rules of Character Escaping Process

```
<!DOCTYPE schema [
<!ENTITY process
"//sml:ProcessModel[sml:method/@xlink:role='urn:ogc:def:process:OGC:1.0:escChar
acter']">
<!ENTITY input "sml:inputs/sml:InputList/sml:input">
<!ENTITY output "sml:outputs/sml:OutputList/sml:output">
<!ENTITY param "sml:parameters/sml:ParameterList/sml:parameter"> ]>
<sch:schema xmlns:sch="http://www.ascc.net/xml/schematron">
    <sch:title>Escape Character  Process</sch:title>
<!-- Namespaces definitions -->
<sch:ns prefix="sml" uri="http://www.opengis.net/sensorML/1.0.1"/>
<sch:ns prefix="swe" uri="http://www.opengis.net/swe/1.0.1"/>
<sch:ns prefix="xlink" uri="http://www.w3.org/1999/xlink"/>

<sch:pattern id="IO_CHECK" name="Check I/O Characteristics">
    <sch:rule context="&process;">
        <!-- Escape Character input -->
        <sch:assert test="&input;[@name='data']">Input named 'data' must be
present</sch:assert>
        <!-- Escape Character output -->
        <sch:assert test="&output;[@name='data']">Output named 'data' must be
present</sch:assert>
        <!-- Escape Character parameter -->
        <sch:assert test="&param;[@name='escapeDefinition']">Prameter named
'escapeDefinition' must be present</sch:assert>
    </sch:rule>
</sch:pattern>
    <sch:pattern name="CRC_Rocksoft">
        <sch:rule context="&param;[@name='escapeDefinition']">
            <sch:assert
test="swe:DataArray/swe:elementType[@name='escapeCharacters']/swe:DataRecord">D
ataRecord named 'escapeCharacters' must be present in Parameter block
'escapeDefinition'</sch:assert>
            <sch:assert
test="swe:DataArray/swe:elementType[@name='escapeCharacters']/swe:DataRecord/sw
e:field[@name='charFrom']">Parameter  named 'charFrom' must be present in
escapeCharacters Block </sch:assert>
            <sch:assert
test="swe:DataArray/swe:elementType[@name='escapeCharacters']/swe:DataRecord/sw
e:field[@name='charTo']">Parameter  named 'charTo' must be present in
escapeCharacters Block </sch:assert>
            <sch:assert test="swe:DataArray/swe:encoding">encoding must be
present in escapeDefinition </sch:assert>
            </sch:rule>
```

```
    </sch:pattern>
</sch:schema>
```

B.2 Schematron rules of Checksum Validation Process

```
<!DOCTYPE schema [
<!ENTITY process
"//sml:ProcessModel[sml:method/@xlink:role='urn:ogc:def:process:OGC:1.0:checkSu
m']">
<!ENTITY input "sml:inputs/sml:InputList/sml:input">
<!ENTITY output "sml:outputs/sml:OutputList/sml:output">
<!ENTITY param "sml:parameters/sml:ParameterList/sml:parameter"> ]>
<sch:schema xmlns:sch="http://www.ascc.net/xml/schematron">
<sch:title>CheckSum Process</sch:title>
<!-- Namespaces definitions -->
<sch:ns prefix="sml" uri="http://www.opengis.net/sensorML/1.1.0"/>
<sch:ns prefix="swe" uri="http://www.opengis.net/swe/1.1.0"/>
<sch:ns prefix="xlink" uri="http://www.w3.org/1999/xlink"/>

<sch:pattern id="IO_CHECK" name="Check I/O Characteristics">
    <sch:rule context="&process;">
        <!-- CheckSum  input -->
        <sch:assert test="&input;[@name='data'] or &input;[@name='data'] and
&input;[@name='checkSum']">Input named 'data' must be present</sch:assert>
        <!-- CheckSum  output -->
        <sch:assert test="&output;[@name='checkSum']/swe:Text or
&output;[@name='data']">Output Text named 'checkSum' must be
present</sch:assert>
        <!-- CheckSum  parameter -->
        <sch:assert test="&param;[@name='type']">Prameter named 'type' must be
present</sch:assert>
    </sch:rule>
</sch:pattern>
    <sch:pattern name="CRC_Rocksoft">
        <sch:rule
context="&param;[@name='type']/swe:DataRecord/swe:field[@name='crc']">
            <sch:assert test="swe:DataRecord">Parameter DataRecord named 'crc'
must be present</sch:assert>
            <sch:assert
test="swe:DataRecord/swe:field[@name='name']/swe:Text">Parameter Text named
'name' must be present in crc Block </sch:assert>
            <sch:assert
test="swe:DataRecord/swe:field[@name='width']/swe:Count">Parameter Count named
'widt' must be present in crc Block </sch:assert>
            <sch:assert
test="swe:DataRecord/swe:field[@name='poly']/swe:Text">Parameter Text named
'poly' must be present in crc Block </sch:assert>
            <sch:assert
test="swe:DataRecord/swe:field[@name='init']/swe:Text">Parameter Text named
'init' must be present in crc Block </sch:assert>
            <sch:assert
test="swe:DataRecord/swe:field[@name='refin']/swe:Boolean">Parameter Boolean
named 'refin' must be present in crc Block </sch:assert>
            <sch:assert
test="swe:DataRecord/swe:field[@name='refout']/swe:Boolean">Parameter Boolean
named 'refout' must be present in crc Block </sch:assert>
            <sch:assert
test="swe:DataRecord/swe:field[@name='xorout']/swe:Text">Parameter Text named
'xorout' must be present in crc Block </sch:assert>
            <sch:assert
test="swe:DataRecord/swe:field[@name='check']/swe:Text">Parameter Text named
'check' must be present in crc Block </sch:assert>
```

```
            </sch:rule>
    </sch:pattern>
</sch:schema>
```

B.3 Schematron rules of Interpolation Process

```
<!DOCTYPE schema [
<!ENTITY process
"//sml:ProcessModel[sml:method/@xlink:role='urn:ogc:def:process:OGC:1.0:interpo
lation:linear'] |
//sml:ProcessModel[sml:method/@xlink:role='urn:ogc:def:process:OGC:1.0:interpol
ation:bicubic']">
<!ENTITY input "sml:inputs/sml:InputList/sml:input">
<!ENTITY output "sml:outputs/sml:OutputList/sml:output">
<!ENTITY param "sml:parameters/sml:ParameterList/sml:parameter"> ]>
<sch:schema xmlns:sch="http://www.ascc.net/xml/schematron">
    <sch:title>Interpolation Process</sch:title>
    <!-- Namespaces definitions -->
    <sch:ns prefix="sml" uri="http://www.opengis.net/sensorML/1.1.0"/>
    <sch:ns prefix="swe" uri="http://www.opengis.net/swe/1.1.0"/>
    <sch:ns prefix="xlink" uri="http://www.w3.org/1999/xlink"/>

    <sch:pattern id="IO_CHECK" name="Check I/O Characteristics">
        <sch:rule context="&process;">
            <!-- Interpolation input -->
            <sch:assert test="&input;[@name='value']/swe:Quantity">Input
Quantity named 'value' must be present </sch:assert>
            <!-- Interpolation output -->
            <sch:assert test="&output;[@name='result']/swe:Quantity">Output
Quantity named 'result' must be present</sch:assert>
            <!-- Interpolation parameter -->
            <sch:assert test="&param;[@name='settings']/swe:DataArray">Prameter
DataRecord named 'settings' must be present</sch:assert>
            <sch:assert
test="&param;[@name='settings']/swe:DataArray/swe:elementType[@name='values']/s
we:DataRecord">The elementType named 'values' of parameter 'settings' must be
present and
                contain a DataRecord element</sch:assert>
            <sch:assert
test="&param;[@name='settings']/swe:DataArray/swe:elementType[@name='values']/s
we:DataRecord/swe:field[@name='x']/swe:Quantity">Prameter Quantity named
'settings/x' must be
                present</sch:assert>
            <sch:assert
test="&param;[@name='settings']/swe:DataArray/swe:elementType[@name='values']/s
we:DataRecord/swe:field[@name='y']/swe:Quantity">Prameter Quantity named
'settings/y' must be
                present</sch:assert>
        </sch:rule>
    </sch:pattern>
</sch:schema>
```

B.4 Schematron rules of Date Conversion Process

```
<!DOCTYPE schema [
<!ENTITY process
"//sml:ProcessModel[sml:method/@xlink:role='urn:ogc:def:process:OGC:1.0:dateCon
version']">
<!ENTITY input "sml:inputs/sml:InputList/sml:input">
<!ENTITY output "sml:outputs/sml:OutputList/sml:output">
```

```
<!ENTITY param "sml:parameters/sml:ParameterList/sml:parameter"> ]>
<sch:schema xmlns:sch="http://www.ascc.net/xml/schematron">
<sch:title>Date Conversion Process</sch:title>
<!-- Namespaces definitions -->
<sch:ns prefix="sml" uri="http://www.opengis.net/sensorML/1.1.0"/>
<sch:ns prefix="swe" uri="http://www.opengis.net/swe/1.1.0"/>
<sch:ns prefix="xlink" uri="http://www.w3.org/1999/xlink"/>

<sch:pattern id="IO_CHECK" name="Check I/O Characteristics">
    <sch:rule context="&process;">
        <!-- Date Conversion input -->
        <sch:assert test="&input;[@name='date']/swe:Text">Input Text named
'date' must be present</sch:assert>
        <!-- Date Conversion output -->
        <sch:assert test="&output;[@name='date']/swe:Text">Output Text named
'date' must be present</sch:assert>
        <!-- Date Conversion parameter -->
        <sch:assert
test="&param;[@name='dateSettings']/swe:DataRecord">Prameter DataRecord named
'dateSettings' must be present</sch:assert>
        <sch:assert
test="&param;[@name='dateSettings']/swe:DataRecord/swe:field[@name='inputFormat
']/swe:Text">Prameter Text named 'dateSettings/inputFormat' must be
present</sch:assert>
        <sch:assert
test="&param;[@name='dateSettings']/swe:DataRecord/swe:field[@name='inputFormat
']/swe:Text/swe:value/text()">Value of 'dateSettings/inputFormat' must be
present and not null</sch:assert>
        <sch:assert
test="&param;[@name='dateSettings']/swe:DataRecord/swe:field[@name='outputForma
t']/swe:Text">Prameter 'dateSettings/outputFormat' must be present</sch:assert>
        <sch:assert
test="&param;[@name='dateSettings']/swe:DataRecord/swe:field[@name='outputForma
t']/swe:Text/swe:value/text()">Value of 'dateSettings/outputFormat' must be
present and not null</sch:assert>
</sch:rule>
</sch:pattern>
</sch:schema>
```

B.5 Schematron rules for Definition of Addressing

```
<!DOCTYPE schema [
<!ENTITY addressing "//sml:interface">
]>

<sch:schema xmlns:sch="http://www.ascc.net/xml/schematron">
<sch:title>Interface Definition addressing parameter</sch:title>
<!-- Namespaces definitions -->
<sch:ns prefix="sml" uri="http://www.opengis.net/sensorML/1.1.0"/>
<sch:ns prefix="xlink" uri="http://www.w3.org/1999/xlink"/>

<sch:pattern id="ADDR_CHECK" name="Check addressing parameter">
    <sch:rule context="&addressing;">
        <!-- CheckSum  input -->
         <sch:assert test="[@xlink:role='urn:ogc:def:connection:OGC:serial'] or
                         [@xlink:role='urn:ogc:def:connection:OGC:tcpIP'] or
                         [@xlink:role='urn:ogc:def:connection:OGC:http'] or
                         [@xlink:role='urn:ogc:def:connection:OGC:udpIP'] or
                         [@xlink:role='urn:ogc:def:connection:OGC:file']">
xlink:role must be present</sch:assert>
    </sch:rule>
</sch:pattern>
</sch:schema>
```


B.6 Schematron rules for Definition of Observation Metadata

```
<!DOCTYPE schema [
<!ENTITY output "//sml:outputs/sml:OutputList/sml:output">
<!ENTITY offering "swe:DataRecord/gml:metaDataProperty/offering">
<!ENTITY foi
"swe:DataRecord/gml:metaDataProperty/featureOfInterest/@xlink:href">
<!ENTITY observedProperty
"swe:DataRecord/gml:metaDataProperty/observedProperty/@xlink:href">
]>
<sch:schema xmlns:sch="http://www.ascc.net/xml/schematron">
<sch:title>CheckSum Process</sch:title>
<!-- Namespaces definitions -->
<sch:ns prefix="sml" uri="http://www.opengis.net/sensorML/1.1.0"/>
<sch:ns prefix="swe" uri="http://www.opengis.net/swe/1.1.0"/>
<sch:ns prefix="xlink" uri="http://www.w3.org/1999/xlink"/>
<sch:ns prefix="gml" uri="http://www.opengis.net/gml"/>

<sch:pattern id="METADATA_CHECK" name="Check Observation Metadata">
    <sch:rule context="&output;">
        <!-- CheckSum  input -->
        <sch:assert test="&offering;">offering must be present</sch:assert>
        <!-- CheckSum  output -->
        <sch:assert test="&foi;">feature of interest must be
present</sch:assert>
        <!-- CheckSum  parameter -->
        <sch:assert test="&observedProperty;">observed property must be
present</sch:assert>
    </sch:rule>
</sch:pattern>
</sch:schema>
```

Annex C
(informative)

Example XML document

This annex provides the complete example of the SensorML document describing an MWS3 sensor system including its SID.


```
<?xml version="1.0" encoding="UTF-8"?>
<sml:System
    xmlns:sml="http://www.opengis.net/sensorML/1.1.0"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:swe="http://www.opengis.net/swe/1.1.0"
    xmlns:sid="http://www.orangenkiste.de/SID/0.5.0"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/sensorML/1.1.0 ../xsd/SID/sid.xsd">

    <sml:identification>
        <sml:IdentifierList>
            <sml:identifier name="uniqueID">
                <sml:Term definition="urn:ogc:def:identifier:OGC:uniqueID">
                    <sml:value>urn:ogc:object:feature:MWS3</sml:value>
                </sml:Term>
            </sml:identifier>
            <sml:identifier name="longName">
                <sml:Term definition="urn:ogc:def:identifier:OGC:1.0:longName">
                    <sml:value>Messwertsensor 3</sml:value>
                </sml:Term>
            </sml:identifier>
        </sml:IdentifierList>
    </sml:identification>
    <sml:capabilities>
        <swe:DataRecord definition="urn:ogc:def:property:OGC::status">
            <!--station is collecting data-->
            <swe:field name="status">
                <swe:Boolean>
                    <swe:value>true</swe:value>
                </swe:Boolean>
            </swe:field>
            <swe:field name="mobile">
                <swe:Boolean>
                    <swe:value>false</swe:value>
                </swe:Boolean>
            </swe:field>
            <swe:field name="measuringInterval">
                <swe:Quantity
definition="urn:ogc:def:property:OGC:1.0:measuringInterval">
                    <gml:description>The measuring interval of the MWS3
</gml:description>
                    <swe:uom code="ms"/>
                    <swe:value>1000</swe:value>
                </swe:Quantity>
            </swe:field>
        </swe:DataRecord>
    </sml:capabilities>
    <sml:position name="Position">
        <swe:Position definition="urn:ogc:def:property:OGC:1.0:stationPosition"
referenceFrame="urn:ogc:def:crs:EPSG:4326">
            <swe:location>
                <swe:Vector>
                    <swe:coordinate name="latitude">
                        <swe:Quantity axisID="y">
                            <swe:uom code="deg"/>
```



```
                            <swe:constraint>
                                <swe:AllowedValues>
                                    <swe:interval>-180 180</swe:interval>
                                </swe:AllowedValues>
                            </swe:constraint>
                            <swe:value>7.2</swe:value>
                        </swe:Quantity>
                    </swe:coordinate>
                    <!-- name="Geodetic longitude" -->
                    <swe:coordinate name="longitude">
                        <swe:Quantity axisID="x">
                            <swe:uom code="deg"/>
                            <swe:constraint>
                                <swe:AllowedValues>
                                    <swe:interval>-180 180</swe:interval>
                                </swe:AllowedValues>
                            </swe:constraint>
                            <swe:value>52</swe:value>
                        </swe:Quantity>
                    </swe:coordinate>
                </swe:Vector>
            </swe:location>
        </swe:Position>
    </sml:position>
    <sml:interfaces>
        <sml:InterfaceList>
            <sml:interface name="mws" xlink:role="urn:ogc:def:connection:OGC:file">
                <sml:InterfaceDefinition>
                 <sml:applicationLayer>
                        <sid:CommandDefinition>
                                 <sid:commands>
                                     <sid:CommandList>
                                         <command name="setSamplingRateCommand">
                                             <sid:CommandParameters
name="setSamplingRateCommandParameters">
                                                 <swe:DataRecord>
                                                     <swe:field name="commandName"
xlink:role="urn:ogc:def:command:OGC:name">
                                                         <swe:Text>
                                                             <swe:value>SR</swe:value>
                                                         </swe:Text>
                                                     </swe:field>
                                                     <swe:field name="space"
xlink:role="urn:ogc:def:command:OGC:value">
                                                         <swe:Text>
                                                         <swe:value> </swe:value>
                                                         </swe:Text>
                                                     </swe:field>
                                                     <swe:field name="measuringInterval"
xlink:role="urn:ogc:def:parameter:OGC:required">
                                                         <swe:Count/>

                                                     </swe:field>
                                                 </swe:DataRecord>
                                             </sid:CommandParameters>
                                             <sid:ResponseList>
                                                 <response valid="true"
name="ackResponse">
                                                     <swe:DataRecord>
                                                         <swe:field
name="ack"></swe:field>
                                                     </swe:DataRecord>
                                                 </response>
                                                 <response valid="false"
name="nackResponse">
                                                     <swe:DataRecord>
                                                         <swe:field
name="nack"></swe:field>
                                                     </swe:DataRecord>
                                                 </response>
```



```
                                             </sid:ResponseList>

                                         </command>
                             </sid:CommandList>
                         </sid:commands>
                     </sid:CommandDefinition>
                     </sml:applicationLayer>
                    <sml:presentationLayer>

                           <!-- incoming data stream: -->

                     <sid:DataInputStream>
                             <dataInputComponents>
                                 <ComponentList>
                                     <component name="samplingRateCommand">
                                         <swe:DataRecord>
                                             <swe:field name="command"/>
                                         </swe:DataRecord>
                                     </component>

                                 </ComponentList>
                             </dataInputComponents>
                         </sid:DataInputStream>

                        <!-- outgoing data stream: -->

                        <sid:DataOutputStream>
                            <dataOutputComponents>
                                <ComponentList>

                                 <component name="ackResponse">
                                       <swe:DataRecord>
                                           <swe:field name="ack"></swe:field>
                                       </swe:DataRecord>
                                   </component>

                                   <component name="nackResponse">
                                       <swe:DataRecord>
                                           <swe:field name="nack"></swe:field>
                                       </swe:DataRecord>
                                   </component>

                                   <component name="odl_basis_01">
                                     <swe:DataRecord>

                                             <!-- ~~~~~~~~~~~~~~~~~~ sampling time
~~~~~~~~~~~~~~~~~~~~-->

                                            <swe:field name="samplingTime">
                                                <swe:Time>
                                                      <gml:description>time when a data set
was measured</gml:description>

                                                    <swe:uom code="ISO8601"/>
                                                </swe:Time>
                                            </swe:field>

                                            <!-- ~~~~~~~~~~~~~~~~~~ high dose radiation
~~~~~~~~~~~~~~~~~~~~-->

                                            <swe:field name="highDoseRadiation">
                                                  <swe:Quantity
gml:id="highDoseRadiationData">

                                                      <gml:description>high dose radiation
impulses</gml:description>

                                                      <swe:uom code="mSv/a" />
                                                  </swe:Quantity>
                                            </swe:field>

                                            <!-- ~~~~~~~~~~~~~~~~~~ temperature
~~~~~~~~~~~~~~~~~~~~-->

                                              <swe:field name="temp">
```

```
                                                <swe:Quantity gml:id="temperatureData">
<gml:description>temperature</gml:description>
                                                    <swe:uom code="CEL" />
                                                </swe:Quantity>
                                            </swe:field>

                                            <!-- ~~~~~~~~~~~~~~~~~~ humidity
~~~~~~~~~~~~~~~~~~~-->

                                            <swe:field name="humidity">
                                                <swe:Quantity gml:id="humidityData">
<gml:description>humidity</gml:description>
                                                    <swe:uom code="g/m3" />
                                                </swe:Quantity>
                                            </swe:field>

                                      </swe:DataRecord>
                                    </component>
                                </ComponentList>
                            </dataOutputComponents>
                        </sid:DataOutputStream>
                    </sml:presentationLayer>
                    <sml:transportLayer>
                     <sid:Decoder>
                         <decoderComponent>
                            <sml:ProcessChain>
                                    <sml:inputs>
                                        <sml:InputList>
                                            <sml:input name="resistance_hd">
                                               <swe:Quantity/>
                                            </sml:input>
                                        </sml:InputList>
                                    </sml:inputs>
                                    <sml:outputs>
                                        <sml:OutputList>
                                            <sml:output name="radiation_hd">
                                              <swe:Quantity/>
                                            </sml:output>
                                        </sml:OutputList>
                                    </sml:outputs>
                                    <sml:parameters>
                                        <sml:ParameterList>
                                            <sml:parameter name="interpolationParameter">
                                                <swe:DataRecord>
                                                    <swe:field name="resistance">
                                                        <swe:Quantity/>
                                                    </swe:field>
                                                    <swe:field name="radiation">
                                                       <swe:Quantity/>
                                                    </swe:field>
                                                </swe:DataRecord>
                                            </sml:parameter>
                                        </sml:ParameterList>
                                    </sml:parameters>
                                    <sml:components>
                                        <sml:ComponentList>
                                            <sml:component
name="interporlationProcess_hd">
                                              <sml:ProcessModel
gml:id="interpolationProcess">
                                                      <sml:inputs>
                                                          <sml:InputList>
                                                              <sml:input name="value">
                                                                  <swe:Quantity />
                                                              </sml:input>
                                                          </sml:InputList>
                                                      </sml:inputs>
                                                      <sml:outputs>
                                                          <sml:OutputList>
```

```
                                                            <sml:output name="result">
                                                                <swe:Quantity />
                                                            </sml:output>
                                                        </sml:OutputList>
                                                    </sml:outputs>
                                                    <sml:parameters>
                                                        <sml:ParameterList>
                                                            <sml:parameter
name="settings">
                                                                <swe:DataArray>
                                                                    <swe:elementCount
/>
                                                                    <swe:elementType
name="values">
    <swe:DataRecord>
                                                                            <swe:field
name="x">
    <swe:Quantity />
    </swe:field>
                                                                            <swe:field
name="y">
    <swe:Quantity />
    </swe:field>
    </swe:DataRecord>
                                                                    </swe:elementType>
                                                                </swe:DataArray>
                                                            </sml:parameter>
                                                    </sml:ParameterList>
                                                    </sml:parameters>
                                                <sml:method
xlink:href="interpolationMethod.xml"
xlink:role="urn:ogc:def:process:OGC:1.0:interpolation:linear"/>
                                                </sml:ProcessModel>
                                        </sml:component>
                                    </sml:ComponentList>
                                </sml:components>
                                 <sml:connections>
                                        <sml:ConnectionList>

                                            <sml:connection>
                                                <sml:Link
type="urn:ogc:def:link:OGC:sid">
                                                    <sml:source
ref="this/inputs/resistance_hd"/>
                                                    <sml:destination
ref="interpolationProcess_hd/inputs/value"/>
                                                </sml:Link>
                                            </sml:connection>
                                            <sml:connection>
                                                <sml:Link
type="urn:ogc:def:link:OGC:sid">
                                                    <sml:source
ref="interpolationProcess_hd/outputs/result"/>
                                                    <sml:destination
ref="this/outputs/radiation_hd"/>
                                                </sml:Link>
                                            </sml:connection>

                                            <sml:connection>
                                                <sml:Link
type="urn:ogc:def:link:OGC:sid">
                                                    <sml:source
ref="this/parameters/interpolationParameter/resistance"/>
                                                    <sml:destination
ref="interpolationProcess_hd/parameter/settings/values/x"/>
```



```
                                             </sml:Link>
                                       </sml:connection>
                                       <sml:connection>
                                             <sml:Link
type="urn:ogc:def:link:OGC:sid">
                                                 <sml:source
ref="this/parameters/interpolationParameter/radiation"/>
                                                 <sml:destination
ref="interpolationProcess_hd/parameter/settings/values/y"/>
                                             </sml:Link>
                                       </sml:connection>

                                   </sml:ConnectionList>
                             </sml:connections>
                         </sml:ProcessChain>
                     </decoderComponent>
                 </sid:Decoder>
                </sml:transportLayer>
                <sml:networkLayer>
                    <sid:Decoder>
                        <decoderComponent>
                            <sml:ProcessChain>
                                <sml:inputs>
                                    <sml:InputList>
                                        <sml:input name="date01">
                                            <swe:Text
definition="urn:ogc:def:TimeString"/>
                                        </sml:input>
                                        <sml:input name="date10">
                                            <swe:Text
definition="urn:ogc:def:TimeString"/>
                                        </sml:input>
                                    </sml:InputList>
                                </sml:inputs>
                                <sml:outputs>
                                    <sml:OutputList>
                                        <sml:output name="date01">
                                            <swe:Text
definition="urn:ogc:def:TimeString"/>
                                        </sml:output>
                                        <sml:output name="date10">
                                            <swe:Text
definition="urn:ogc:def:TimeString"/>
                                        </sml:output>
                                    </sml:OutputList>
                                </sml:outputs>
                                <sml:parameters>
                                    <sml:ParameterList>
                                        <sml:parameter name="dateSettings">
                                            <swe:DataRecord>
                                                <swe:field name="inputFormat">

<swe:Text><swe:value>T</swe:value></swe:Text>
                                                </swe:field>
                                                <swe:field name="outputFormat">
                                                    <swe:Text><swe:value>yyyy-MM-
dd'T'HH:mm:ssz</swe:value></swe:Text>
                                                </swe:field>
                                            </swe:DataRecord>
                                        </sml:parameter>
                                    </sml:ParameterList>
                                </sml:parameters>
                                <sml:components>
                                    <sml:ComponentList>
                                        <sml:component name="dateConversion_01">
                                            <sml:ProcessModel
gml:id="dateConversion">
                                                <sml:inputs>
                                                    <sml:InputList>
                                                        <sml:input name="date">
```

```
                                                        <swe:Text
definition="urn:ogc:def:TimeString"/>
                                                    </sml:input>
                                                </sml:InputList>
                                            </sml:inputs>
                                            <sml:outputs>
                                                <sml:OutputList>
                                                    <sml:output name="date">
                                                        <swe:Text
definition="urn:ogc:def:TimeString"/>
                                                    </sml:output>
                                                </sml:OutputList>
                                            </sml:outputs>
                                            <sml:parameters>
                                                <sml:ParameterList>
                                                    <sml:parameter
name="dateSettings">
                                                        <swe:DataRecord>
                                                            <swe:field
name="inputFormat"/>
                                                            <swe:field
name="outputFormat"/>
                                                        </swe:DataRecord>
                                                    </sml:parameter>
                                                </sml:ParameterList>
                                            </sml:parameters>
                                            <sml:method
xlink:href="dateConversion.xml#123"
xlink:role="urn:ogc:def:process:OGC:1.0:dateConversion"/>
                                        </sml:ProcessModel>
                                   </sml:component>
                                    <sml:component name="dateConversion_10"
xlink:href="#dateConversion"/>
                                </sml:ComponentList>
                            </sml:components>
                             <sml:connections>
                                 <sml:ConnectionList>
                                     <sml:connection>
                                         <sml:Link
type="urn:ogc:def:link:OGC:sid">
                                             <sml:source
ref="this/inputs/date01"/>
                                             <sml:destination
ref="dateConversion_01/inputs/date"/>
                                         </sml:Link>
                                     </sml:connection>
                                     <sml:connection>
                                         <sml:Link
type="urn:ogc:def:link:OGC:sid">
                                             <sml:source
ref="this/inputs/date10"/>
                                             <sml:destination
ref="dateConversion_10/inputs/date"/>
                                         </sml:Link>
                                     </sml:connection>
                                     <sml:connection>
                                         <sml:Link
type="urn:ogc:def:link:OGC:sid">
                                             <sml:source
ref="dateConversion_01/outputs/date"/>
                                             <sml:destination
ref="this/outputs/date01"/>
                                         </sml:Link>
                                     </sml:connection>
                                     <sml:connection>
                                         <sml:Link
type="urn:ogc:def:link:OGC:sid">
                                             <sml:source
ref="dateConversion_10/outputs/date"/>
                                             <sml:destination
ref="this/outputs/date10"/>
```

```
                                            </sml:Link>
                                        </sml:connection>
                                        <sml:connection>
                                            <sml:Link
type="urn:ogc:def:link:OGC:sid">
                                                <sml:source
ref="this/parameters/dateSettings/inputFormat"/>
                                                <sml:destination
ref="dateConversion_01/parameters/dateSettings/inputFormat"/>
                                            </sml:Link>
                                        </sml:connection>
                                        <sml:connection>
                                            <sml:Link
type="urn:ogc:def:link:OGC:sid">
                                                <sml:source
ref="this/parameters/dateSettings/outputFormat"/>
                                                <sml:destination
ref="dateConversion_01/parameters/dateSettings/outputFormat"/>
                                            </sml:Link>
                                        </sml:connection>
                                        <sml:connection>
                                            <sml:Link
type="urn:ogc:def:link:OGC:sid">
                                                <sml:source
ref="this/parameters/dateSettings/inputFormat"/>
                                                <sml:destination
ref="dateConversion_10/parameters/dateSettings/inputFormat"/>
                                            </sml:Link>
                                        </sml:connection>
                                        <sml:connection>
                                            <sml:Link
type="urn:ogc:def:link:OGC:sid">
                                                <sml:source
ref="this/parameters/dateSettings/outputFormat"/>
                                                <sml:destination
ref="dateConversion_10/parameters/dateSettings/outputFormat"/>
                                            </sml:Link>
                                        </sml:connection>
                                    </sml:ConnectionList>

                                </sml:connections>
                           </sml:ProcessChain>
                       </decoderComponent>
                   </sid:Decoder>
               </sml:networkLayer>
               <sml:physicalLayer>

                    <!-- incoming data stream: -->

                <sid:DataInputStream>
                        <dataInputComponents>
                            <ComponentList>
                                <component name="samplingrateCommand">
                                    <swe:DataRecord>
                                        <swe:field name="command"/>
                                    </swe:DataRecord>
                                </component>
                            </ComponentList>
                        </dataInputComponents>
                    </sid:DataInputStream>

                      <!-- outgoing data stream: -->

                   <sid:DataOutputStream>
                       <dataOutputComponents>
                           <ComponentList>

                            <component name="ackResponse">
                                  <swe:DataBlockDefinition>
                                      <swe:components name="data">
                                          <swe:DataRecord>
```

```
                                                    <swe:field name="ack"
xlink:role="urn:ogc:def:encoding:assertedValue">
                                                        <swe:Text>
                                                            <swe:value>0x06</swe:value>
                                                        </swe:Text>
                                                    </swe:field>
                                                </swe:DataRecord>
                                            </swe:components>
                                            <swe:encoding>
                                               <swe:BinaryBlock byteEncoding="hex"
byteOrder="bigEndian">
                                                   <swe:member>
                                                       <swe:Block ref="data/ack"
byteLength="1"/>
                                                   </swe:member>
                                               </swe:BinaryBlock>
                                            </swe:encoding>
                                        </swe:DataBlockDefinition>
                                    </component>

                                    <component name="nackResponse">
                                        <swe:DataBlockDefinition>
                                            <swe:components name="data">
                                                <swe:DataRecord>
                                                    <swe:field name="nack"
xlink:role="urn:ogc:def:encoding:assertedValue">
                                                        <swe:Text>
                                                            <swe:value>0x15</swe:value>
                                                        </swe:Text>
                                                    </swe:field>
                                                </swe:DataRecord>
                                            </swe:components>
                                            <swe:encoding>
                                                <swe:BinaryBlock byteEncoding="hex"
byteOrder="bigEndian">
                                                    <swe:member>
                                                        <swe:Block ref="data/nack"
byteLength="1"/>
                                                    </swe:member>
                                                </swe:BinaryBlock>
                                            </swe:encoding>
                                        </swe:DataBlockDefinition>
                                    </component>

                                    <component name="M01">
                                        <swe:DataBlockDefinition>
                                            <swe:components name="data">
                                                <swe:DataRecord>

                                                      <!-- ~~~~~~~~~~~~ identifier of the
data block ~~~~~~~~~~~-->

                                                    <swe:field name="id"
xlink:role="urn:ogc:def:encoding:assertedValue">
                                                        <swe:Text>
                                                            <gml:description>data block
identifier</gml:description>
                                                            <swe:value>M01</swe:value>
                                                        </swe:Text>
                                                    </swe:field>

                                                    <!-- ~~~~~~~~~~~~~~~~~~ time
tag~~~~~~~~~~~~~~~~~~-->

                                                    <swe:field name="timeTag">
                                                        <swe:Count>
                                                            <gml:description>time tag of
the measurement in ms</gml:description>
                                                        </swe:Count>
                                                    </swe:field>
```

```
                                                    <!-- ~~~~~~~~~~~~~~~high dose
resistance~~~~~~~~~~~~~~-->

                                                    <swe:field name="imp_hd_resistance">
                                                        <swe:Quantity>
                                                            <gml:description>resistance
of high dose radiation detector</gml:description>

                                                        </swe:Quantity>
                                                    </swe:field>

                                                    <!-- ~~~~~~~~~~~~~~~ temperature
~~~~~~~~~~~~~~~~~~-->

                                                    <swe:field name="temp">
                                                        <swe:Quantity>

<gml:description>temperature</gml:description>

                                                        </swe:Quantity>
                                                    </swe:field>

                                                    <!-- ~~~~~~~~~~~~~~~~~~ humidity
~~~~~~~~~~~~~~~~~~~-->

                                                    <swe:field name="humidity">
                                                        <swe:Quantity>

<gml:description>humidity</gml:description>

                                                        </swe:Quantity>
                                                    </swe:field>

                                                </swe:DataRecord>
                                            </swe:components>
                                            <swe:encoding>
                                                <swe:TextBlock tokenSeparator="|"
blockSeparator="&#x000A;" decimalSeparator="."/>
                                            </swe:encoding>
                                        </swe:DataBlockDefinition>
                                    </component>

                                </ComponentList>
                            </dataOutputComponents>
                        </sid:DataOutputStream>
                    </sml:physicalLayer>
                    <sml:connections>
                        <sml:ConnectionList>

                             <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~ -->
                             <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Connections for
ack DataSet ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
                             <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~ -->

                             <!-- Linking of incoming setSamplingRateCommand: -->

                            <sml:connection>
                                <sml:Link type="urn:ogc:def:link:OGC:sid">
                                    <sml:source
ref="applicationLayer/commands/setSamplingRateCommand"/>
                                    <sml:destination
ref="presentationLayer/dataInputComponents/samplingrateCommand"/>
                                </sml:Link>
                            </sml:connection>
                            <sml:connection>
                                <sml:Link type="urn:ogc:def:link:OGC:sid">
                                    <sml:source
ref="presentationLayer/dataInputComponents/samplingrateCommand/setSamplingRateCommandPara
meters"/>
```

```
                                     <sml:destination
ref="physicalLayer/dataInputComponents/samplingrateCommand"/>
                                 </sml:Link>
                             </sml:connection>

                              <!-- Linking of outgoing 'ack' response (response to
setSamplingRateCommand): -->

                              <sml:connection>
                                 <sml:Link type="urn:ogc:def:link:OGC:sid">
                                     <sml:source
ref="physicalLayer/dataOutputComponents/ackResponse/data/ack"/>
                                     <sml:destination
ref="presentationLayer/dataOutputComponents/ackResponse/ack"/>
                                 </sml:Link>
                             </sml:connection>
                             <sml:connection>
                                 <sml:Link type="urn:ogc:def:link:OGC:sid">
                                     <sml:source
ref="presentationLayer/dataOutputComponents/ackResponse/ack"/>
                                     <sml:destination
ref="applicationLayer/commands/setSamplingRateCommand/ackResponse/ack"/>
                                 </sml:Link>
                             </sml:connection>

                              <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~ -->
                              <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Connections for
M01 DataSet ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
                              <!--
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~ -->

                              <!-- Linking time tag through date conversion process
(networkLayer) to presentationLayer: -->

                             <sml:connection>
                                 <sml:Link type="urn:ogc:def:link:OGC:sid">
                                     <sml:source
ref="physicalLayer/dataOutputComponents/M01/data/timeTag"/>
                                     <sml:destination
ref="networkLayer/decoderComponent/inputs/date01"/>
                                 </sml:Link>
                             </sml:connection>
                             <sml:connection>
                                 <sml:Link type="urn:ogc:def:link:OGC:sid">
                                     <sml:source
ref="networkLayer/decoderComponent/outputs/date01"/>
                                     <sml:destination
ref="presentationLayer/dataOutputComponents/odl_basis_01/samplingTime"/>
                                 </sml:Link>
                             </sml:connection>

                              <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ high dose
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->

                             <!-- Linking high dose radiation data through interpolation
process (transportLayer) to presentationLayer: -->

                             <sml:connection>
                                 <sml:Link type="urn:ogc:def:link:OGC:sid">
                                     <sml:source
ref="physicalLayer/dataOutputComponents/M01/data/imp_hd_resistance"/>
                                     <sml:destination
ref="transportLayer/decoderComponents/inputs/resistance_hd"/>
                                 </sml:Link>
                             </sml:connection>
                             <sml:connection>
                                 <sml:Link type="urn:ogc:def:link:OGC:sid">
```

```
                                    <sml:source
ref="transportLayer/decoderComponents/outputs/radiation_hd"/>
                                    <sml:destination
ref="presentationLayer/dataOutputComponents/odl_basis_01/highDoseRadiation"/>
                                </sml:Link>
                            </sml:connection>

                            <!-- Direct linking of physicalLayer::dataOutputComponents to
presentationLayer::dataOutputComponents -->

                            <sml:connection>
                                <sml:Link type="urn:ogc:def:link:OGC:sid">
                                    <sml:source
ref="physicalLayer/dataOutputComponents/M01/data/temp"/>
                                    <sml:destination
ref="presentationLayer/dataOutputComponents/odl_basis_01/temp"/>
                                </sml:Link>
                            </sml:connection>
                            <sml:connection>
                                <sml:Link type="urn:ogc:def:link:OGC:sid">
                                    <sml:source
ref="physicalLayer/dataOutputComponents/M01/data/humidity"/>
                                    <sml:destination
ref="presentationLayer/dataOutputComponents/odl_basis_01/humidity"/>
                                </sml:Link>
                            </sml:connection>

                        </sml:ConnectionList>
                    </sml:connections>
                </sml:InterfaceDefinition>

            </sml:interface>
        </sml:InterfaceList>
    </sml:interfaces>

    <sml:inputs></sml:inputs>
    <sml:outputs>
        <sml:OutputList>

            <sml:output name="highDoseRadiation_output">
                <swe:DataRecord>
                    <gml:metaDataProperty>
                        <offering>RADIATION_OFFERING</offering>
                    </gml:metaDataProperty>
                    <gml:metaDataProperty>
                         <featureOfInterest
xlink:href="http://myWFS.org/features/Muenster_city"/>
                    </gml:metaDataProperty>
                    <gml:metaDataProperty>
                         <observedProperty
xlink:href="urn:ogc:def:property:OGC:radiation"/>
                    </gml:metaDataProperty>
                    <swe:field name="dataSet" xlink:href="#highDoseRadiationData" /> <!--
this xlink:href includes the element with that gml:id as a sub-element here -->
                </swe:DataRecord>
            </sml:output>

            <sml:output name="temperature_output">
                 <swe:DataRecord>
                    <gml:metaDataProperty>
                        <offering>TEMPERATURE_OFFERING</offering>
                    </gml:metaDataProperty>
                    <gml:metaDataProperty>
                         <featureOfInterest
xlink:href="http://myWFS.org/features/Muenster_city"/>
                    </gml:metaDataProperty>
                    <gml:metaDataProperty>
                         <observedProperty
xlink:href="urn:ogc:def:property:OGC:temperature"/>
                    </gml:metaDataProperty>
```

```
                    <swe:field name="dataSet" xlink:href="#temperatureData" /> <!-- this
xlink:href includes the element with that gml:id as a sub-element here -->
                </swe:DataRecord>

            </sml:output>

            <sml:output name="humidity_output">
                <swe:DataRecord>
                    <gml:metaDataProperty>
                        <offering>HUMIDITY_OFFERING_01</offering>
                    </gml:metaDataProperty>
                    <gml:metaDataProperty>
                         <featureOfInterest
xlink:href="http://myWFS.org/features/Muenster_city"/>
                    </gml:metaDataProperty>
                    <gml:metaDataProperty>
                         <observedProperty
xlink:href="urn:ogc:def:property:OGC:humidity"/>
                    </gml:metaDataProperty>
                    <swe:field name="dataSet" xlink:href="#humidityData" />
                </swe:DataRecord>
            </sml:output>

        </sml:OutputList>
    </sml:outputs>

    <sml:components>
    <sml:ComponentList>
        <sml:component name="radiometer">
                <sml:Component>
                    <sml:parameters>
                        <sml:ParameterList>
                            <sml:parameter name="steadyStateCurve">
                                <swe:Curve>
                                    <swe:elementCount>
                                        <swe:Count>
                                            <swe:value>2</swe:value>
                                        </swe:Count>
                                    </swe:elementCount>
                                    <swe:elementType>
                                        <swe:SimpleDataRecord>
                                            <swe:field name="resistance">
                                                <swe:Quantity
definition="urn:ogc:def:property:OGC:resistance"/>
                                            </swe:field>
                                            <swe:field name="radiation">
                                                <swe:Quantity
definition="urn:ogc:def:property:OGC:radiation"/>
                                            </swe:field>
                                        </swe:SimpleDataRecord>
                                    </swe:elementType>
                                    <swe:encoding>
                                        <swe:TextBlock
                                            tokenSeparator=","
                                            decimalSeparator="."
                                            blockSeparator="; " />
                                    </swe:encoding>
                                    <swe:values>-3,-5; 6,7; 8.2,7.5</swe:values>
                                </swe:Curve>
                            </sml:parameter>
                        </sml:ParameterList>
                    </sml:parameters>
                </sml:Component>
        </sml:component>
    </sml:ComponentList>
    </sml:components>

    <sml:connections>
        <sml:ConnectionList>
```



```xml
            <!-- Instantiation of SID internal interpolation process. Linking of
externally defined process parameters to parameters of interpolation process: -->
            <sml:connection>
                 <sml:Link>
                     <sml:source
    ref="components/barometer/parameters/steadyStateCurve/resistance" />
                     <sml:destination
ref="this/interfaces/mws/transportLayer/decoderComponent/parameters/interpolationParamete
r/resistance" />
                 </sml:Link>
             </sml:connection>
             <sml:connection>
                 <sml:Link>
                     <sml:source
    ref="components/barometer/parameters/steadyStateCurve/radiation" />
                     <sml:destination
ref="this/interfaces/mws/transportLayer/decoderComponent/parameters/interpolationParamete
r/radiation" />
                 </sml:Link>
             </sml:connection>

             <!-- Linking presentationLayer::dataOutputComponents to outputs: -->
            <sml:connection>
                <sml:Link type="urn:ogc:def:link:OGC:sid">
                    <sml:source
ref="this/interfaces/mws/presentationLayer/dataOutputComponents/odl_basis_01/highDoseRadi
ation"/>
                    <sml:destination
ref="this/outputs/highDoseRadiation_output/dataSet"/>
                </sml:Link>
            </sml:connection>

            <sml:connection>
                <sml:Link type="urn:ogc:def:link:OGC:sid">
                    <sml:source
ref="this/interfaces/mws/presentationLayer/dataOutputComponents/odl_basis_01/temp"/>
                    <sml:destination ref="this/outputs/temperature_output/dataSet/"/>
                </sml:Link>
            </sml:connection>

             <sml:connection>
                <sml:Link type="urn:ogc:def:link:OGC:sid">
                    <sml:source
ref="this/interfaces/mws/presentationLayer/dataOutputComponents/odl_basis_01/humidity"/>
                    <sml:destination ref="this/outputs/humidity_output/dataSet/"/>
                </sml:Link>
            </sml:connection>

         </sml:ConnectionList>
    </sml:connections>
</sml:System>
```

Bibliography

[1] M. Botts, G. Percivall, C. Reed, and J. Davidson, “OGC Sensor Web Enablement: Overview and High Level Architecture,” *Lecture Notes In Computer Science*, vol. 4540, pp. 175–190, 2008.

[2] A. Na and M. Priest, “OGC Implementation Specification 06-009r6: OpenGIS Sensor Observation Service (SOS),” 2007.

[3] I. Simonis, “OGC Best Practices 06-028r3: OGC Sensor Alert Service Candidate Implementation Specification,” Open Geospatial Consortium, Tech. Rep., 2006.

[4] J. Echterhoff and T. Everding, “OGC Discussion Paper 08-133: OpenGIS Sensor Event Service Interface Specification,” Open Geospatial Consortium, Tech. Rep., 2008.

[5] I. Simonis, “OGC Implementation Specification 07-014r3: OpenGIS Sensor Planning Service,” Open Geospatial Consortium, Tech. Rep., 2007.

[6] S. Cox, “OGC Implementation Specification 07-022r1: Observations and Measurements - Part 1 - Observation schema,” Open Geospatial Consortium, 2007.

[7] M. Botts, “OGC Implementation Specification 07-000: OpenGIS Sensor Model Language (SensorML),” 2007.

[8] L.-K. Chung, B. Baranski, Y.-M. Fang, Y.-H. Chang, T.-Y. Chou, and B. J. Lee, “A SOA based debris flow monitoring system - Architecture and proof-of-concept implementation,” in *The 17th International Conference on Geoinformatics 2009*, Fairfax, USA, 2009.

[9] S. Jirka, A. Broering, and C. Stasch, “Applying OGC Sensor Web Enablement to Risk Monitoring and Disaster Management,” in *GSDI 11 World Conference, Rotterdam, Netherlands*, June 2009.

[10]C. Stasch, A. C. Walkowski, and S. Jirka, “A Geosensor Network Architecture for Disaster Management based on Open Standards.” in *Digital Earth Summit on Geoinformatics 2008: Tools for Climate Change Research.*, M. Ehlers, K. Behncke, F. W. Gerstengabe, F. Hillen, L. Koppers, L. Stroink, and J. Wächter, Eds., 2008, pp. 54–59.

[11]G. Schimak and D. Havlik, “Sensors Anywhere - Sensor Web Enablement in Risk Management Applications,” *ERCIM News 2009 - The Sensor Web*, no. 76, pp. 40 – 41, 2009.

[12]K. Walter and E. Nash, “Coupling Wireless Sensor Networks and the Sensor Observation Service â€“ Bridging the Interoperability Gap,” in *12th AGILE International Conference on Geographic Information Science 2009*, Hannover, Germany, 2009.

[13]K. Aberer, M. Hauswirth, and A. Salehi, “Middleware support for the Internet of Things,” *5. GI/ITG KuVS Fachgespraech - Drahtlose Sensornetze*, pp. 15 – 19, 2006.

[14]ISO/IEC, “ISO/IEC 7498-1: Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model,” ISO, 1996.

[15]R. Williams. (1993, August) A Painless Guide to CRC Error Detection Algorithms. Rocksoft Ltd. Hazelwood Park, Australia. [Online]. Available: http://www.csm.ornl.gov/-~dunigan/crc.html

[16]A. Broering, T. Foerster, S. Jirka, and C. Priess, “Sensor Bus: An Intermediary Layer for Linking Geosensor Networks and the Sensor Web,” in *COM.Geo 2010: 1st International Conference on Computing for Geospatial Research and Applications*, Washington DC, USA, 21.-30. June 2010 2010; forthcoming.