Open Geospatial Consortium

Date: 2010-10-12

Reference number of this document: OGC 10-171

Category: Public Discussion Paper

Editor(s): Simon Jirka, Daniel Nüst

OGC® Sensor Instance Registry Discussion Paper

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: NA
Document stage: Approved for Public Release
Document language: English

Contents

Page

Figures

Page

Tables

Page

i. Preface

Suggested additions, changes, and comments on this draft report are welcome and encouraged. Such suggestions may be submitted by email message or by making suggested changes in an edited copy of this document.

The changes made in this document version, relative to the previous version, are tracked by Microsoft Word, and can be viewed if desired. If you choose to submit suggested changes by editing this document, please first accept all the current changes, and then make your suggested changes with change tracking on.

ii. Document terms and definitions

This document uses the standard terms defined in Subclause 5.3 of [OGC 06-121r3], 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 standard.

iii. Submitting organizations

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

a. Institute for Geoinformatics (IfGI), University of Münster (WWU)

b. 52° North Initiative for Geospatial Open Source Software GmbH

iv. Document contributor contact points

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

Name	Organization
Simon Jirka	52° North GmbH
Daniel Nüst	IfGI, University of Münster

v. Revision history

Date	Release	Editor	Primary clauses modified	Description
2010-09-01	0.3.0	Simon Jirka	All	First version of the Discussion Paper
2010-10-08		Carl Reed	Various	Preparation For Public release

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

For the future it is intended to stronger align the SIR specification to other existing SWE service interfaces. Especially the discovery functionality and the access to sensor status information might be realized using operations of the Sensor Observation Service. The subscription to sensor status information might also be covered by operations described in the Sensor Event Service specification.

Furthermore it will be important to keep the SIR specification in line with the developments described in the following OGC Discussion Papers:

- OGC 09-033, OWS-6 SensorML Profile for Discovery Engineering Report: The SIR makes used of a well defined structure of SensorML documents in order to automatically process them. Thus, future versions of SIR implementations shall support future versions of the SensorML Profile for Discovery.

- OGC 09-163r2, SensorML Extension Package for ebRIM Application Profile: The transformation of SensorML based sensor metadata into an ebRIM based Catalog Information Model is based on this Discussion Paper. Thus, future SIR implementations shall be kept up-to-data in order to also support advancements of the SensorML-ebRIM mapping.

- OGC 09-112, Sensor Observable Registry Discussion Paper: The Sensor Observable Registry (SOR) provides means for semantically enhancing sensor discovery requests. Thus, the SIR shall be kept updated in order to make use of the fully range of the SOR functionality.

Foreword

This Discussion paper introduces the Sensor Instance Registry (SIR), a web service interface for managing the metadata and status information of sensors. Furthermore this service is capable of automatically harvesting sensor metadata, transforming the collected metadata sets into a data model compatible to OGC Catalogs and to push harvested metadata into OGC Catalog instances.

With this Discussion Paper the authors aim at presenting a pragmatic solution that may serve as the basis for further discussion. It is expected that the functionality described in this discussion paper can partly be brought in line with other existing SWE specifications. Especially, it shall be investigated how parts of the Sensor Observation Service (SOS) and Sensor Event Service (SES) specifications can be re-used for partially providing the functionality described in this Discussion Paper.

This work was supported by the following EC projects:

- OSIRIS (http://www.osiris-fp6.eu/, contract number 033475, co-funded by the DG Information Society and Media of the European Commission)

- GENESIS (http://genesis-fp7.eu/, contract number 223996, co-funded by the DG Information Society and Media of the European Commission)

- EO2HEAVEN (http://www.eo2heaven.org/, contract number 244100, co-funded by the DG RTD-I (Environment) of the European Commission)

Introduction

This paper introduces the Sensor Instance Registry (SIR), a web service interface for discovering sensors, collecting sensor metadata, handling sensor status information and for automatically inserting sensor metadata into OGC Catalogs.

The SIR is intended to close the gap between the SensorML based metadata model used in the SWE framework and the information models used by OGC Catalogs. Thus, it is not only capable of managing SensorML based sensor metadata but also to transform these metadata into Catalog Information Models (i.e. ebRIM).

Generally the SIR interface can be grouped into four parts:

- sensor discovery
- sensor metadata collection
- sensor status handling
- catalog linking

The discovery part of the SIR interface comprises operations for searching sensors as well as for retrieving SensorML based descriptions of these sensors. Compared to rather static Catalogs, the SIR is capable of including near real-time information into discovery requests and at the same time to use comprehensive sets of sensor metadata encoded in SensorML. By linking to the Sensor Observable Registry described in OGC 09-112 the SIR is furthermore able to exploit semantic relationships in sensor discovery requests.

To collect the sensor metadata necessary for enabling sensor discovery, the sensor metadata collection part of the SIR interface offers according operations. On the one hand the SIR is able to automatically harvest sensor metadata from SWE service instances (i.e. by issueing a sequence of GetCapabilities and DescribeSensor requests). On the other hand it also provides means for directly inserting sensor descriptions as well as information about which sensors are encapsulated by which SWE service instances. SensorML documents which are processed by the SIR shall comply with the SensorML Profile for Discovery as it is defined in OGC 09-163. This ensures the completeness of sensor metadata for discovery purposes as well as a consistent metadata structure that can automatically be processed by a SIR instance.

The sensor status handling handling part provides functionality for handling status information about sensors. Whereas the data contained in a SensorML description is usually more focussed on providing information for discovering sensors and for interpreting their data, the highly dynamic sensor status information is not well covered. Complete SensorML documents are a large overhead for this purpose. Thus a specific set of operations for dealing with the status of sensors is defined. This includes operations

for searching sensors based on their status (e.g. finding all sensors with a critical battery state), retrieving the status of sensors (including only specific parameters), subscribing to sensor status information (e.g. alert if a sensor reaches a critical battery level) and for inserting sensor status information into a SIR instance.

Finally, the catalog link part of the SIR interface offers two operations that allow connecting SIR instances to OGC Catalogs. These operations allow managing the links to OGC Catalogs (i.e. connecting and disconnecting as well as scheduling automatic metadata updates). The functionality provided by the catalog link part of the SIR interface includes transformation mechanisms for translating SensorML based sensor descriptions into OGC Catalog Information Models like ebRIM.

In summary the SIR is intended as a broker mediating between the highly dynamic and SensorML based world of sensor networks and classic spatial data infrastructure concepts, especially OGC Catalog. This link is achieved through a set of functionality comprising sensor metadata collection, sensor discovery, sensor status management and sensor metadata transformation as well as sensor metadata transfer.

For the future it is intended to further align the SIR specification to other existing SWE service interfaces. Especially the discovery functionality and the access to sensor status information might be realized using operations of the Sensor Observation Service. The subscription to sensor status information might also be covered by operations described in the Sensor Event Service specification.

OGC Sensor Instance Registry Discussion Paper

1 Scope

This Discussion paper describes the Sensor Instance Registry (SIR), a web service interface for managing the metadata and status information of sensors. Furthermore this service is capable of automatically harvesting sensor metadata, transforming the collected metadata sets into a data model compatible to OGC Catalogs, and to push harvested metadata into OGC Catalog instances.

This document is intended to stimulate the discussion on mechanisms for enabling sensor discovery. The authors aim at presenting a pragmatic solution that may serve as the basis for further discussion. It is expected that the functionality described in this discussion paper can partly be brought in line with other existing SWE standards.

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 06-121r3, *OpenGIS® Web Services Common Standard*

OGC 07-000, *OpenGIS® Sensor Model Language (SensorML) Implementation Standard*

OGC 07-036, *OpenGIS® Geography Markup Language (GML) Encoding Standard*

OGC 07-036, *OpenGIS® Geography Markup Language (GML) Encoding Standard*

OGC 09-033, *OWS-6 SensorML Profile for Discovery Engineering Report*

OGC 09-112, *Sensor Observable Registry Discussion Paper*

OGC 09-163r2, *SensorML Extension Package for ebRIM Application Profile*

NOTE The OWS Common Standard contains a list of normative references that are also applicable to this Implementation Standard.

3 Terms and definitions

For the purposes of this specification, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r3] shall apply. In addition, the following terms and definitions apply.

observable
Parameter or a characteristic of a phenomenon subject to observation. [OGC 07-022r1]

phenomenon
Characteristic of one or more feature types, the value for which must be estimated by application of some procedure in an observation [OGC 07-022r1].

sensor
An entity capable of observing a phenomenon and returning an observed value. [OGC07-000]

sensor discovery
The process of searching for sensors or for SWE services that encapsulate them.

sensor instance
An individual sensing device.

sensor service
An instance of a SWE service providing access to sensor data, encapsulating sensor instances or providing other sensor related functionality.

4 Conventions

4.1 Abbreviated terms

Most of the abbreviated terms listed in Subclause 5.1 of the OWS Common Implementation Standard [OGC 06-121r3] apply to this document, plus the following abbreviated terms.

Some more frequently used abbreviated terms:

SensorML	Sensor Model Language
SIR	Sensor Instance Registry
SIR	Sensor Observable Registry
SWE	Sensor Web Enablement
URI	Uniform Resource Identifier
URN	Uniform Resource Name

5 Sensor Instance Registry Overview

The specified Sensor Instance Registry is capable of harvesting, managing and transforming sensor metadata. In particular is provides operations for searching sensors, inserting metadata, handling the status of sensors and for linking SIR instances to OGC Catalogs.

The SIR interface (currently) specifies 14 operations that can be requested by a client and performed by a SIR server. These operations are:

a) GetCapabilities (required implementation by servers) – This operation allows a client to receive service metadata (namely Capabilities) documents that describe the abilities of the specific server implementation. This operation also supports negotiation of the standard version being used for client-server interactions.

Operations for searching sensors and retrieving sensor metadata:

b) SearchSensor (required implementation by servers) – This operation allows a client to search for sensor instances. It is possible to specify spatial, temporal and thematic search criteria. The result is the description of the discovered sensors as well as information about the SWE service instances encapsulating these sensors.

c) DescribeSensor (required implementation by servers) – This operation allows a client to retrieve a SensorML based description of a sensor managed by a SIR instance.

Operations for inserting, deleting and updating sensor metadata:

d) HarvestService – This operation allows a client to initiate the harvesting process of sensor metadata from a certain SWE service instance. When executing this request, the SIR will retrieve all sensor metadata contained in the SWE service instance specified by the client and subsequently process and store the information.

e) InsertSensorInfo – This operation allows a client to insert sensor metadata into a SIR instance. This can concern either the complete metadata record of a sensor or relationships between sensor instance and SWE service instances.

f) DeleteSensorInfo – This operation allows a client to delete sensor metadata from a SIR instance. This can concern either the complete metadata record of a sensor or relationships between sensor instance and SWE service instances.

g) UpdateSensorDescription – This operation allows a client to update the description of a sensor already contained in a SIR instance.

Operations for managing sensor status information:

h) GetSensorStatus – This operation allows a client to request status information about a specific sensor. For specifying the status information a client is interested in, the

GetSensorStatus operation offers a broad range of spatial, temporal and thematic search options.

i) SubscribeSensorStatus – This operation allows a client to subscribe to status information about a specific sensor. For specifying the status information a client is interested in, the SubscribeSensorStatus operation offers a broad range of spatial, temporal and thematic criteria.

j) RenewSensorStatusSubscription – This operation allows a client to extend an existing subscription for sensor status information.

k) CancelSensorStatusSubscription – This operation allows a client to stop an existing subscription for sensor status information.

l) InsertSensorStatus – This operation allows a client to insert status information about a sensor contained in the SIR instance.

Operations for managing the links between SIR instances and OGC Catalogs:

m) ConnectToCatalog – This operation allows a client to establish a link between a SIR instance and a specific OGC Catalog server. Using this request a continuous push process of sensor metadata into the OGC Catalog can be initiated.

n) DisconnectFromCatalog – This operation allows a client to stop the process of pushing metadata from a SIR instance into an OGC Catalog server.

Many of these interface aspects are common with other OWSs and thus specified in the OpenGIS® Web Services Common Implementation Standard [OGC 06-121r3]. Hence, many of the common aspects are normatively referenced herein, instead of being repeated in this standard.

Each of the SIR operations is described in more detail in subsequent clauses.

6 SIR Common

6.1 Introduction

This clause specifies common encoding elements that are shared by several operations.

6.2 ServiceReference

The ServiceReference element describes how a sensor can be accessed though a specific SWE service. It contains references to the service as well as information how the sensor is identified within that service. Thus, the ServiceReference element can also be used for uniquely identifying a sensor. The parameter names, meanings, data types, and multiplicity shall be as specified in Table 1.

Table 1 — Parameters in ServiceReference

Names	Definition	Data type and value	Multiplicity and use
ServiceURL	URL pointing to a SWE service instance.	Character string representing a URL	One (mandatory)
ServiceType	String describing the type of the SWE service (i.e. “SOS”, “SPS”, “SAS” or “SES”)	Character String type, not empty	One (mandatory)
ServiceSpecificSensorID	String identifying the sensor within the SWE service instance	Character String type, not empty	One (mandatory)

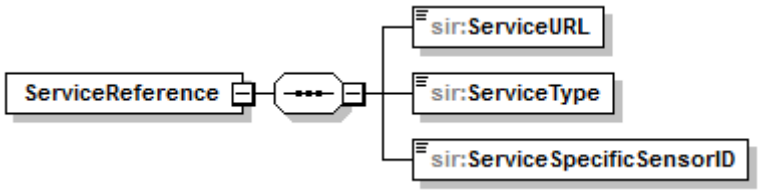

Figure 1: ServiceReference shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of ServiceReference encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:element name="ServiceReference">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="ServiceURL" type="xs:anyURI"/>
```

```
            <xs:element name="ServiceType" type="xs:string"/>
            <xs:element name="ServiceSpecificSensorID"
                        type="xs:string"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

A ServiceReference element can look like this encoded in XML:

```
<sir:ServiceReference>
   <sir:ServiceURL>http://my.service.url/sos</sir:ServiceURL>
   <sir:ServiceType>SOS</sir:ServiceType>
   <sir:ServiceSpecificSensorID>id:017</sir:ServiceSpecificSensorID>
</sir:ServiceReference>
```

6.3 SensorIdentification

The SensorIdentification element is used for providing a unique identifier for a certain sensor when interacting with a SIR server. It consists either of a sensor id specific to the SIR instance of a ServiceReference element. The parameter names, meanings, data types, and multiplicity shall be as specified in Table 2.

Table 2 — Parameters in SensorIdentification

Names	Definition	Data type and value	Multiplicity and use
SensorIDInSIR	String identifying the sensor within the SIR instance	Character String type, not empty	Zero or one (optional)
ServiceReference	Uniqie identification of the sensor through a eference to the sensor within a specific SWE service instance	SIR ServiceReference type	Zero or one (optional)

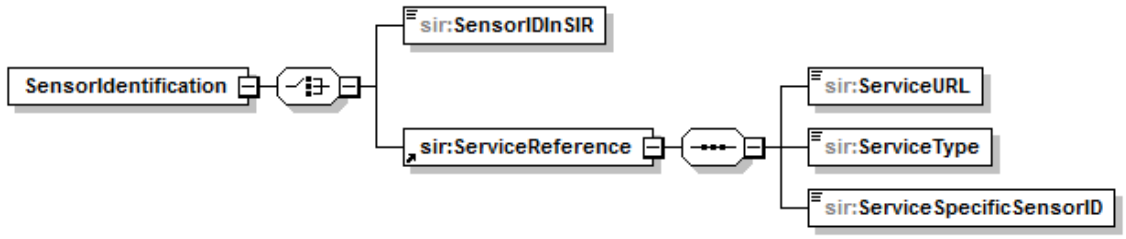

Figure 2: SensorIdentification shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of SensorIdentification encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
```

```
            attributeFormDefault="unqualified">
   <xs:element name="SensorIdentification">
      <xs:complexType>
         <xs:choice>
            <xs:element name="SensorIDInSIR" type="xs:string"/>
            <xs:element ref="sir:ServiceReference"/>
         </xs:choice>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

SensorIdentification elements can look like this encoded in XML:

```
<SensorIdentification>
   <SensorIDInSIR>100</SensorIDInSIR>
</SensorIdentification>

<SensorIdentification>
   <ServiceReference>
      <ServiceURL>http://my.service.url/sps</ServiceURL>
      <ServiceType>SPS</ServiceType>
      <ServiceSpecificSensorID>id:42</ServiceSpecificSensorID>
   </ServiceReference>
</SensorIdentification>
```

6.4 SensorDescription

The SensorDescription element contains a SensorML description of a sensor. The SensorML description shall be compliant to the latest revision of [OGC 09-163]. The parameter names, meanings, data types, and multiplicity shall be as specified in Table 3.

Table 3 — Parameters in SensorDescription

Names	Definition	Data type and value	Multiplicity and use
SensorDescription	SensorML process describing the sensor	SensorML AbstractProcessType	One (mandatory)

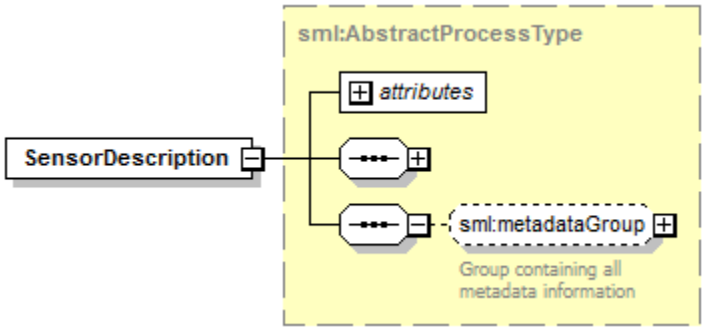

Figure 3: SensorDescription shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of SensorDescription encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           xmlns:sml="http://www.opengis.net/sensorML/1.0.1"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/sensorML/1.0.1"
              schemaLocation=
              "http://schemas.opengis.net/sensorML/1.0.1/process.xsd"/>
   <xs:element name="SensorDescription"
               type="sml:AbstractProcessType"/>
</xs:schema>
```

A SensorDescription element can look like this encoded in XML (shortened example):

```
<SensorDescription xsi:type="sml:SystemType"
                   xmlns:sml="http://www.opengis.net/sensorML/1.0.1">
   <gml:description>Test sensor in the vicinity of Münster,
                    Germany</gml:description>
   <sml:keywords>
       ...
   </sml:keywords>
   ...
   ...
   ...
   <sml:inputs>
       ...
   </sml:inputs>
   <sml:outputs>
       ...
   </sml:outputs>
</SensorDescription>
```

6.5 SimpleSensorDescription

The SimpleSensorDescription element offers a more compact sensor description. Instead of a complete SensorML document it contains a textual description and a URL where the complete SensorML document can be retrieved. The parameter names, meanings, data types, and multiplicity shall be as specified in Table 4.

Table 4 — Parameters in SimpleSensorDescription

Names	Definition	Data type and value	Multiplicity and use
SensorDescriptionURL	URL pointing to the SensorML document describing the sensor	Character string representing a URL	One (mandatory)
DescriptionText	Textual description of the sensor	Character String type, not empty	One (mandatory)

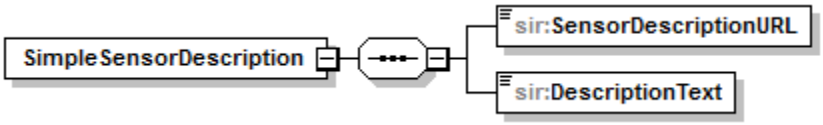

Figure 4: SimpleSensorDescription shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of SimpleSensorDescription encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:element name="SimpleSensorDescription">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="SensorDescriptionURL" type="xs:anyURI"/>
            <xs:element name="DescriptionText" type="xs:string"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

A SimpleSensorDescription element can look like this encoded in XML:

```
<sir:SimpleSensorDescription>
   <sir:SensorDescriptionURL>
      http://giv-genesis.uni-muenster.de:8080/SIR/sir?service=SIR&amp;
      version=0.3.0&amp;REQUEST=DescribeSensor&amp;SENSORIDINSIR=8
   </sir:SensorDescriptionURL>
   <sir:DescriptionText>
      A weather station setup at the Institute for Geoinformatics.
   </sir:DescriptionText>
</sir:SimpleSensorDescription>
```

6.6 ServiceCriteria

The ServiceCriteria element can be used within discovery requests to specify the SWE service instances or types in which a user is interested. The parameter names, meanings, data types, and multiplicity shall be as specified in Table 5.

Table 5 — Parameters in ServiceCriteria

Names	Definition	Data type and value	Multiplicity and use
ServiceURL	URL pointing to a SWE service instance.	Character string representing a URL	Zero or one (optional)
ServiceType	String describing the type of the SWE service (i.e. “SOS”, “SPS”, “SAS” or “SES”)	Character String type, not empty	Zero or one (optional)

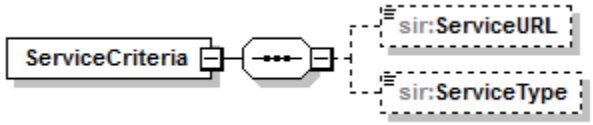

Figure 5: ServiceCriteria shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of ServiceCriteria encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:element name="ServiceCriteria">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="ServiceURL" type="xs:anyURI"
                        minOccurs="0"/>
            <xs:element name="ServiceType" type="xs:string"
                        minOccurs="0"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

A ServiceCriteria element can look like this encoded in XML:

```
<ServiceCriteria>
   <ServiceURL>
      http://v-swe.uni-muenster.de:8080/WeatherSOS
   </ServiceURL>
   <ServiceType>SOS</ServiceType>
</ServiceCriteria>
```

6.7 SearchCriteria

The SearchCriteria element can be used by client to define the search criteria within a discovery request. It contains several properties regarding the sensor, the services encapsultating the sensor as well as a means for specifying if and how the semantics of sensor observables shall be exploited. The parameter names, meanings, data types, and multiplicity shall be as specified in Table 6.

Table 6 — Parameters in SearchCriteria

Names	Definition	Data type and value	Multiplicity and use
ServiceCriteria	Description through which SWE services or SWE service type the sensors the client searches shall	SIR ServiceCriteria type	Zero or more (optional)

Names	Definition	Data type and value	Multiplicity and use
	be encapsulated.		
SearchText	String that shall be contained in the description of the sensors the client searches.	Character String type, not empty	Zero or more (optional)
Phenomenon	Phenomenon for which the sensors the client searches shall provide data	SIR Phenomenon type (see below)	Zero or more (optional)
Uom	Unit of measurement in which the sensor the client searches shall provide data	SWE UoM type	Zero or more (optional)
BoundingBox	BoundingBox for which the sensor the client searches shall provide data	OWS BoundingBox type	Zero or one (optional)
Time	Time for which the sensor the client searches shall provide data	GML AbstractTimeGeometricPrimitive type	Zero or one (optional)

Table 7 — Parameters in Phenomenon

Names	Definition	Data type and value	Multiplicity and use
PhenomenonName	Identifier of the phenomenon	Character String type, not empty	One (mandatory)
SORParameters	Parameters necessary if a SOR instance shall be used for finding related phenomena; if this parameter is not present, then a SOR will not be used.	SIR SORParameters (see below)	Zero or one (optional)

Table 8 — Parameters in SORParameters

Names	Definition	Data type and value	Multiplicity and use
SORURL	URL pointing to a SOR instance.	Character string representing a URL	One (mandatory)
MatchingType	Type of relation between phenomena that shall be used for reasoning and finding matching phenomena	Value can be “SUPER_TYPE”, “EQUIVALENT_TYPE” or “SUB_TYPE”	One (mandatory)
SearchDepth	Number of intermediate step that are allowed in case of transitively related phenomena	xs:int, 0 or higher	One (mandatory)

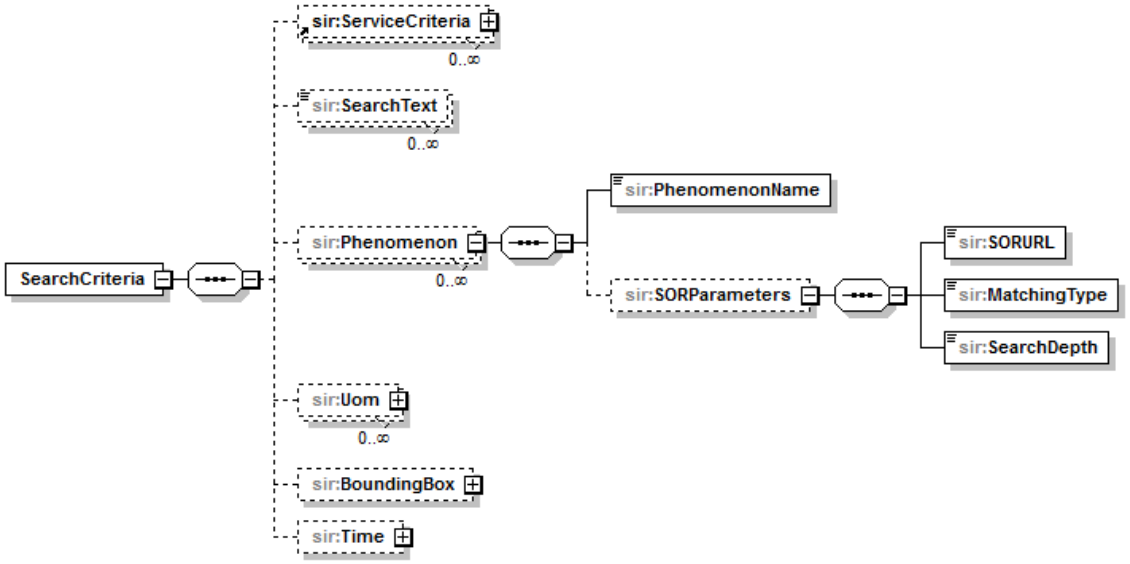

Figure 6: SearchCriteria shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of SearchCriteria encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           xmlns:swe="http://www.opengis.net/swe/1.0.1"
           xmlns:ows="http://www.opengis.net/ows/1.1"
           xmlns:gml="http://www.opengis.net/gml/3.2"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/swe/1.0.1"
              schemaLocation=
                 "http://schemas.opengis.net/sweCommon/1.0.1/swe.xsd"/>
   <xs:import namespace="http://www.opengis.net/ows/1.1"
              schemaLocation=
                 "http://schemas.opengis.net/ows/1.1.0/owsCommon.xsd"/>
   <xs:import namespace="http://www.opengis.net/gml/3.2"
              schemaLocation=
                       "http://schemas.opengis.net/gml/3.2.1/gml.xsd"/>
   <xs:element name="SearchCriteria">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="sir:ServiceCriteria" minOccurs="0"
                        maxOccurs="unbounded"/>
            <xs:element name="SearchText" type="xs:string"
                        minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Phenomenon" minOccurs="0"
                        maxOccurs="unbounded">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="PhenomenonName"
```

```
                        type="xs:anyURI"/>
                <xs:element name="SORParameters" minOccurs="0">
                   <xs:complexType>
                      <xs:sequence>
                         <xs:element name="SORURL"
                                     type="xs:anyURI"/>
                         <xs:element name="MatchingType">
                            <xs:simpleType>
                               <xs:restriction base="xs:string">
                                  <xs:enumeration
                                              value="SUPER_TYPE"/>
                                  <xs:enumeration
                                         value="EQUIVALENT_TYPE"/>
                                  <xs:enumeration
                                               value="SUB_TYPE"/>
                               </xs:restriction>
                            </xs:simpleType>
                         </xs:element>
                         <xs:element name="SearchDepth">
                            <xs:simpleType>
                               <xs:restriction base="xs:int">
                                  <xs:minInclusive value="1"/>
                               </xs:restriction>
                            </xs:simpleType>
                         </xs:element>
                      </xs:sequence>
                   </xs:complexType>
                </xs:element>
             </xs:sequence>
          </xs:complexType>
       </xs:element>
       <xs:element name="Uom" type="swe:UomPropertyType"
                   minOccurs="0" maxOccurs="unbounded"/>
       <xs:element name="BoundingBox" type="ows:BoundingBoxType"
                   minOccurs="0"/>
       <xs:element name="Time"
                   type="gml:AbstractTimeGeometricPrimitiveType"
                   minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

A SearchCriteria element can look like this encoded in XML:

```
<SearchCriteria>
   <ServiceCriteria>
      <ServiceURL>
         http://v-swe.uni-muenster.de:8080/WeatherSOS/sos
      </ServiceURL>
      <ServiceType>SOS</ServiceType>
   </ServiceCriteria>
   <Phenomenon>
      <PhenomenonName>
         urn:x-ogc:def:property:OGC::RelativeHumidity
      </PhenomenonName>
   </Phenomenon>
```

```
   <BoundingBox crs="urn:ogc:def:crs:EPSG:6.14:4326" dimensions="2">
      <ows:LowerCorner>-180.0 -90.0 </ows:LowerCorner>
      <ows:UpperCorner>180.0 90.0 </ows:UpperCorner>
   </BoundingBox>
</SearchCriteria>
```

6.8 PropertyValue

The PropertyValue can be used for describing the value of any property of a sensor. It restricts the possible values to a set of simple data types. The parameter names, meanings, data types, and multiplicity shall be as specified in Table 9.

Table 9 — Parameters in PropertyValue

Names	Definition	Data type and value	Multiplicity and use
PropertyValue	Value of a property	xs:string, xs:integer, xs:double or xs:boolean	One (mandatory)

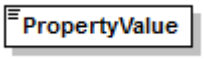

Figure 7: PropertyValue shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of PropertyValue encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:element name="PropertyValue">
      <xs:simpleType>
         <xs:union memberTypes="xs:string xs:integer xs:double
                                xs:boolean"/>
      </xs:simpleType>
   </xs:element>
</xs:schema>
```

A PropertyValue element can look like this encoded in XML:

```
<PropertyValue>sensorIsOperating</PropertyValue>

<PropertyValue>FALSE</PropertyValue>
```

6.9 Status

The Status element can be used for encoding any kind of status information. It comprises the name of the property, the value of the property as well as the unit of measurement and a timestamp. The parameter names, meanings, data types, and multiplicity shall be as specified in Table 10.

Table 10 — Parameters in Status

Names	Definition	Data type and value	Multiplicity and use
PropertyName	Name of the status property	Character String type, not empty	One (mandatory)
PropertyValue	Value of the status property	SIR PropertyValiue type	One (mandatory)
Uom	Unit of measurement in which the status property is provided	SWE UoM type	Zero or more (optional)
Timestamp	Timestamp indicating the time since when the status property value is valid	GML TimeInstant type	Zero or more (optional)

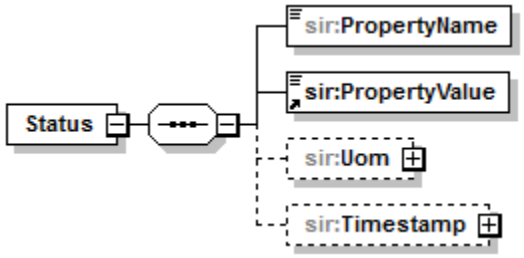

Figure 8: Status shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of Status encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           xmlns:swe="http://www.opengis.net/swe/1.0.1"
           xmlns:gml="http://www.opengis.net/gml/3.2"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/swe/1.0.1"
              schemaLocation="http://schemas.opengis.net/sweCommon/
                 1.0.1/swe.xsd"/>
   <xs:import namespace="http://www.opengis.net/gml/3.2"
              schemaLocation="http://schemas.opengis.net/gml/3.2.1/
                 gml.xsd"/>
   <xs:element name="Status">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="PropertyName" type="xs:anyURI"/>
            <xs:element ref="sir:PropertyValue"/>
            <xs:element name="Uom" type="swe:UomPropertyType"
                        minOccurs="0"/>
            <xs:element name="Timestamp" type="gml:TimeInstantType"
                        minOccurs="0"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

A Status element can look like this encoded in XML:

```
<sir:Status>
   <sir:PropertyName>batteryHoursLeft</sir:PropertyName>
   <sir:PropertyValue>17</sir:PropertyValue>
   <sir:Uom code="hr"/>
</sir:Status>
```

6.10 StatusDescription

The StatusDescription element is used for describing the status of a sensor. It links a set of status elements with the identifier of a specific sensor. The parameter names, meanings, data types, and multiplicity shall be as specified in Table 11.

Table 11 — Parameters in StatusDescription

Names	Definition	Data type and value	Multiplicity and use
SensorINInSIR	Identifier of the sensor for which status information is provided	SIR SensorIdInSIR type	One (mandatory)
Status	Status property of the sensor	SIR Status type	One or more (mandatory)

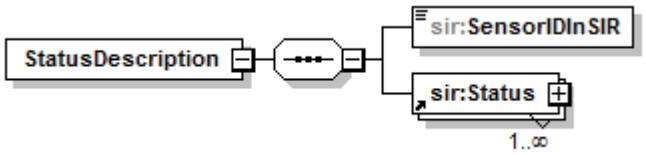

Figure 9: StatusDescription shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of StatusDescription encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:element name="StatusDescription">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="SensorIDInSIR" type="xs:string"/>
             <xs:element ref="sir:Status" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

A StatusDescription element can look like this encoded in XML:

```
<sir:StatusDescription>
```

```
   <sir:SensorIDInSIR>8</sir:SensorIDInSIR>
   <sir:Status>
      <sir:PropertyName>batteryHoursLeft</sir:PropertyName>
      <sir:PropertyValue>17</sir:PropertyValue>
      <sir:Uom code="hr"/>
   </sir:Status>
</sir:StatusDescription>
```

6.11 Constraint

The Constraint element is used for defining value constraints that can be re-used in SIR requests. The parameter names, meanings, data types, and multiplicity shall be as specified in Table 12.

Table 12 — Parameters in Constraint

Names	Definition	Data type and value	Multiplicity and use
IsEqualTo	Property value to which a given one shall be equal to	xs:string, xs:double or xs:boolean	Zero or more (optional)
IsNotEqualTo	Property value to which a given one shall not be equal to	xs:string, xs:double or xs:boolean	Zero or more (optional)
IsGreaterThan	Property value to which a given one shall be geater	xs:double	Zero or more (optional)
IsLessThan	Property value to which a given one shall be less	xs:double	Zero or more (optional)
IsGreaterTahnOrEqualTo	Property value to which a given one shall be geater or equal to	xs:double	Zero or more (optional)
IsLessThanOrEqualTo	Property value to which a given one shall be less or equal to	xs:double	Zero or more (optional)
IsBetween	Property values between which define an interval in which a given value shall lie.	SIR IsBetween type	Zero or more (optional)

Table 13 — Parameters in IsBetween

Names	Definition	Data type and value	Multiplicity and use
LowerBoundary	Property value defining lower boundary of the interval	xs:double	One (mandatory)
UpperBoundary	Property value defining upper boundary of the interval	xs:double	One (mandatory)

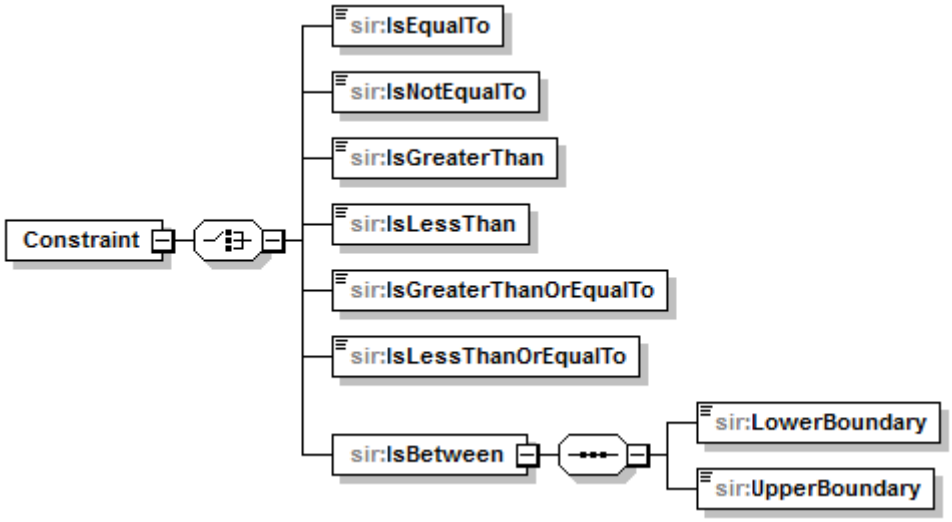

Figure 10: Constraint shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of Constraint encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:element name="Constraint">
      <xs:complexType>
         <xs:choice>
            <xs:element name="IsEqualTo">
               <xs:simpleType>
                  <xs:union memberTypes="xs:string xs:double
                                         xs:boolean"/>
               </xs:simpleType>
            </xs:element>
            <xs:element name="IsNotEqualTo">
               <xs:simpleType>
                  <xs:union memberTypes="xs:string xs:double
                                         xs:boolean"/>
               </xs:simpleType>
            </xs:element>
            <xs:element name="IsGreaterThan" type="xs:double"/>
            <xs:element name="IsLessThan" type="xs:double"/>
            <xs:element name="IsGreaterThanOrEqualTo"
                        type="xs:double"/>
            <xs:element name="IsLessThanOrEqualTo" type="xs:double"/>
            <xs:element name="IsBetween">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="LowerBoundary"
                                 type="xs:double"/>
                     <xs:element name="UpperBoundary"
```

```
                                    type="xs:double"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:choice>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

A Constraint element can look like this encoded in XML:

```
<Constraint>
   <IsLessThan>25.0</IsLessThan>
</Constraint>
```

6.12 PropertyFilter

The PropertyFilter element is used in SIR requests when the requested set of sensors shall be limited to a subset of sensors that possess certain status characteristics. The parameter names, meanings, data types, and multiplicity shall be as specified in Table 14.

Table 14 — Parameters in PropertyFilter

Names	Definition	Data type and value	Multiplicity and use
PropertyName	Name of the status property	Character String type, not empty	One (mandatory)
PropertyConstraint	Constrains that has to be fulfilled by the the PropertyValue	SIR PropertyConstraint type	Zero or one (optional)

Table 15 — Parameters in PropertyConstraint

Names	Definition	Data type and value	Multiplicity and use
Constraint	Constraint that shall be fulfilled by the status property	SIR Constraint type	One (mandatory)
Uom	Unit of measurement in which the constraint is defined	SWE UoM type	Zero or one (optional)

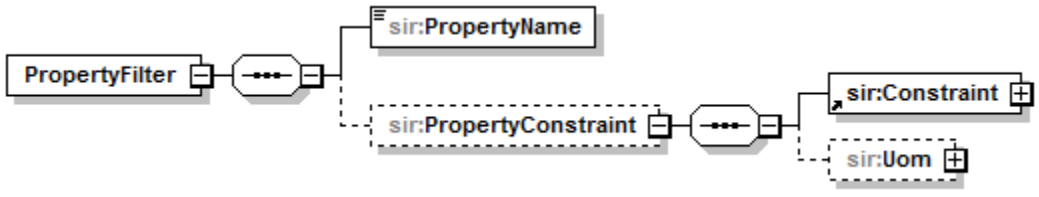

Figure 11: PropertyFilter shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of PropertyFilter encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           xmlns:swe="http://www.opengis.net/swe/1.0.1"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/swe/1.0.1"
              schemaLocation="http://schemas.opengis.net/sweCommon/
                 1.0.1/swe.xsd"/>
   <xs:element name="PropertyFilter">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="PropertyName" type="xs:anyURI"/>
            <xs:element name="PropertyConstraint" minOccurs="0">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="sir:Constraint"/>
                     <xs:element name="Uom" type="swe:UomPropertyType"
                                 minOccurs="0"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

A PropertyFilter element can look like this encoded in XML:

```
<PropertyFilter>
   <PropertyName>batterystate</PropertyName>
   <PropertyConstraint>
      <Constraint>
         <IsEqualTo>55</IsEqualTo>
      </Constraint>
      <Uom code="%"/>
   </PropertyConstraint>
</PropertyFilter>
```

7 GetCapabilities operation (mandatory)

7.1 Introduction

The mandatory GetCapabilities operation allows clients to retrieve service metadata from a server. The response to a GetCapabilities request shall be an XML document containing service metadata about the server. This clause specifies the XML document that a server shall return to describe its capabilities.

7.2 Operation request

The GetCapabilities operation request shall be as specified in Subclauses 7.2 and 7.3 of [OGC 06-121r3]. The value of the "service" parameter shall be "SIR". The allowed set of service metadata (or Capabilities) XML document section names and meanings shall be as specified in Tables 3 and 7 of [OGC 06-121r3].

The "Multiplicity and use" column in Table 1 of [OGC 06-121r3] specifies the optionality of each listed parameter in the GetCapabilities operation request. Table 16 specifies the implementation of those parameters by SIR clients and servers.

Table 16 — Implementation of parameters in GetCapabilities operation request

<table>
<tr><th>Names</th><th>Multiplicity</th><th>Client implementation</th><th>Server implementation</th></tr>
<tr><td>service</td><td>One (mandatory)</td><td rowspan="2">Each parameter shall be implemented by all clients, using specified value</td><td rowspan="2">Each parameter shall be implemented by all servers, checking that each parameter is received with specified value</td></tr>
<tr><td>request</td><td>One (mandatory)</td></tr>
<tr><td>Sections</td><td>Zero or one (optional) [b]</td><td rowspan="3">Each parameter may be implemented by each client [b]
If parameter not provided, shall expect default response
If parameter provided, shall allow default or specified response</td><td rowspan="3">Each parameter may be implemented by each server [a]
If parameter not implemented or not received, shall provide default response
If parameter implemented and received, shall provide specified response</td></tr>
<tr><td>updateSequence</td><td>Zero or one (optional) [b]</td></tr>
<tr><td>AcceptFormats</td><td>Zero or one (optional) [b]</td></tr>
<tr><td colspan="4">a A specific OWS is allowed to make mandatory server implementation of any of these three parameters.
b If a specific OWS makes mandatory server implementation of any of these three parameters, that parameter can also be made mandatory in the operation request, also requiring client implementation of this parameter.</td></tr>
</table>

All SIR servers shall implement HTTP GET transfer of the GetCapabilities operation request, using KVP encoding. Servers may also implement HTTP POST transfer of the GetCapabilities operation request, using XML encoding only.

EXAMPLE 1 To request a SIR capabilities document, a client could issue the following KVP encoded GetCapabilities operation request with near-minimum contents:

```
http://giv-genesis.uni-
muenster.de:8080/SIR/sir?service=SIR&request=getCapabilities
```

EXAMPLE 2 The corresponding GetCapabilities operation request XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<sir:GetCapabilities xmlns:sir="http://swsl.uni-muenster.de/sir"
                     xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                                instance"
                     xmlns:ows="http://www.opengis.net/ows/1.1"
                     service="SIR"
                     xsi:schemaLocation="http://swsl.uni-muenster.de
                        /sir http://giv-genesis/schemas/sir/sirAll.xsd
                        http://www.opengis.net/sensorML/1.0.1
                        http://schemas.opengis.net/sensorML/1.0.1
                        /sensorML.xsd">
</sir:GetCapabilities>
```

7.3 GetCapabilities operation response

The service metadata document shall contain the sections that are defined in the OWS Common standard. For the SIR these sections are not modified. The Contents subsection is specific to the SIR. Depending on the values in the Sections parameter of the GetCapabilities operation request, any combination of these sections can be requested and shall be returned when requested.

Table 17 — Additional Section name values and their meanings

Section name	Contents
Contents	Metadata about the data served by this server. For the SIR, this section shall contain data about the harvested SWE service instances and the OGC Catalog instances that receive sensor metadata from the SIR instance.

The Contents section of a service metadata document contains metadata about the sensor metadata managed by this server. For the SIR, this Contents section shall contain data about data about the harvested SWE service instances and the OGC Catalog instances that receive sensor metadata from the described SIR instance. The Contents section shall include the parameters specified in Table 18.

Table 18 — Parameters included in Contents section

Names	Definition	Data type and values	Multiplicity and use
HarvestedService	Information about a service that has been harvested by the SIR instance	SIR HarvestedService type (see below)	Zero or more (optional)
LinkedCatalog	Information about an OGC Catalog instance that is fed by metadata by the SIR instance.	SIR LinkedCatalog type (see below)	Zero or more (optional)

Table 19 — Parameters included in Harvested section

Names	Definition	Data type and values	Multiplicity and use

Names	Definition	Data type and values	Multiplicity and use
ServiceURL	URL pointing to a SWE service instance.	Character string representing a URL	One (mandatory)
ServiceType	String describing the type of the SWE service (i.e. “SOS”, “SPS”, “SAS” or “SES”)	Character String type, not empty	One (mandatory)

Table 20 — Parameters included in Harvested section

Names	Definition	Data type and values	Multiplicity and use
CatalogURL	URL pointing to an OGC Catalog instance.	Character string representing a URL	One (mandatory)
Status	String indicating the connection status between SIR and Catalog	Character String type, not empty	One (mandatory)
PushIntervalSeconds	Time span (in seconds) describing the interval how often new sensor metadata records are pushed into the Catalog	xs:int, 0 or higher	One (mandatory)

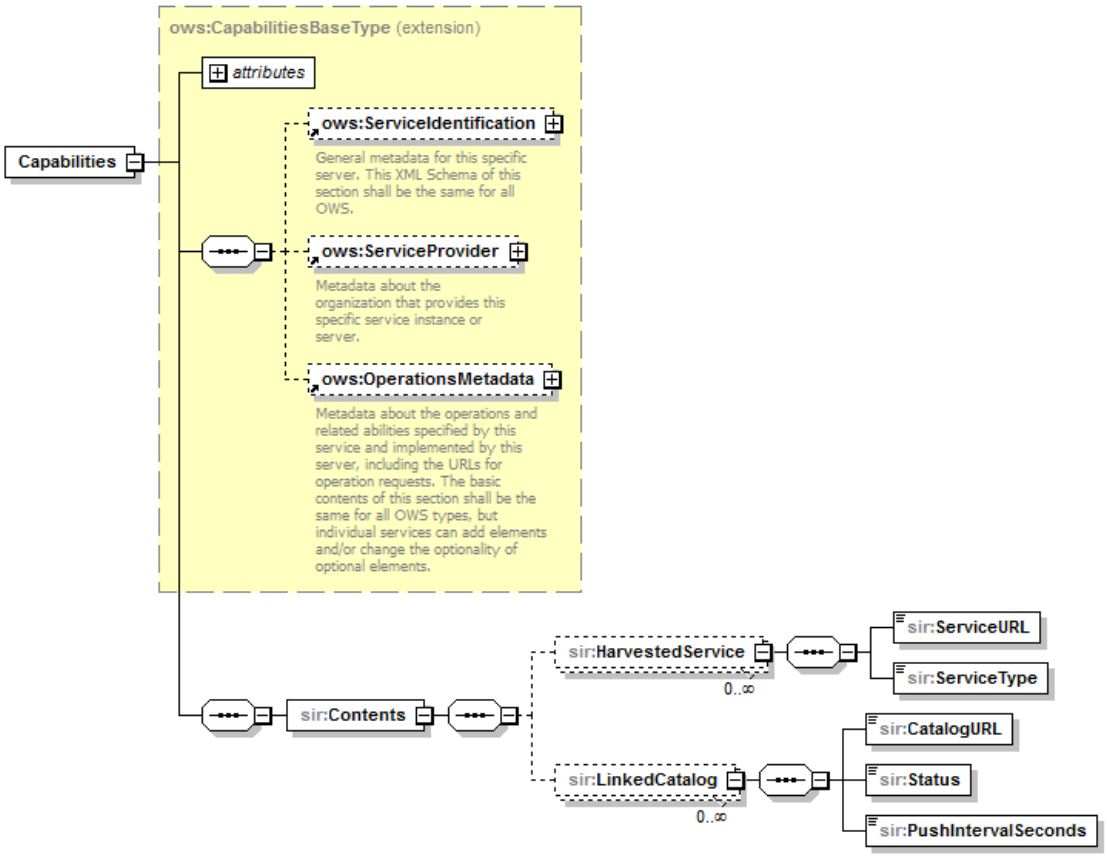

Figure 12: SIR Capabilities shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a SIR Capabilities document, always encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:ogc="http://www.opengis.net/ogc"
        xmlns:ows="http://www.opengis.net/ows/1.1"
        xmlns:sir="http://swsl.uni-muenster.de/sir"
        targetNamespace="http://swsl.uni-muenster.de/sir"
        elementFormDefault="qualified" version="1.0.0" xml:lang="en">
   <import namespace="http://www.opengis.net/ows/1.1"
           schemaLocation="http://schemas.opengis.net/ows/1.1.0
              /owsAll.xsd"/>
   <element name="Capabilities">
      <complexType>
         <complexContent>
            <extension base="ows:CapabilitiesBaseType">
               <sequence>
                  <element name="Contents">
                     <complexType>
                        <sequence>
                           <element name="HarvestedService"
```

```
                                    minOccurs="0"
                                    maxOccurs="unbounded">
                                 <complexType>
                                    <sequence>
                                       <element name="ServiceURL"
                                                type="anyURI"/>
                                       <element name="ServiceType"
                                                type="string"/>
                                    </sequence>
                                 </complexType>
                              </element>
                              <element name="LinkedCatalog"
                                    minOccurs="0"
                                    maxOccurs="unbounded">
                                 <complexType>
                                    <sequence>
                                       <element name="CatalogURL"
                                                type="anyURI"/>
                                       <element name="Status"
                                                type="string"/>
                                       <element name="PushIntervalSeconds"
                                                type="int"/>
                                    </sequence>
                                 </complexType>
                              </element>
                           </sequence>
                        </complexType>
                     </element>
                  </sequence>
               </extension>
            </complexContent>
         </complexType>
      </element>
   </schema>
```

A SIR Capabilities document can look like this encoded in XML (shortened version not showing all operations):

```
<sir:Capabilities version="0.3.0"
                  updateSequence="2010-08-30T19:23:26+0200"
                  xsi:schemaLocation="http://swsl.uni-muenster.de/sir
                     http://giv-genesis/schemas/sir/sirAll.xsd
                     http://www.opengis.net/sensorML/1.0.1
                     http://schemas.opengis.net/sensorML/1.0.1
                     /sensorML.xsd"
                  xmlns:sir="http://swsl.uni-muenster.de/sir"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                     instance">
   <ows:ServiceIdentification
      xmlns:ows="http://www.opengis.net/ows/1.1"
      xmlns:xlink="http://www.w3.org/1999/xlink">
   <ows:Title>WWU SIR</ows:Title>
   <ows:Abstract>SIR Server Operated by WWU</ows:Abstract>
   <ows:ServiceType codeSpace="http://localhost:8080/SIR2/sir">
      OGC:SIR
   </ows:ServiceType>
   <ows:ServiceTypeVersion>0.3.0</ows:ServiceTypeVersion>
```

```
<ows:Fees>NONE</ows:Fees>
<ows:AccessConstraints>NONE</ows:AccessConstraints>
</ows:ServiceIdentification>
<ows:ServiceProvider xmlns:ows="http://www.opengis.net/ows/1.1"
                     xmlns:xlink="http://www.w3.org/1999/xlink">
   <ows:ProviderName>WWU</ows:ProviderName>
   <ows:ProviderSite xlink:href="http://localhost:8080/SIR2/sir"/>
   <ows:ServiceContact>
      <ows:IndividualName>Simon Jirka</ows:IndividualName>
      <ows:ContactInfo>
         <ows:Phone>
            <ows:Voice>222-222-2222</ows:Voice>
            <ows:Facsimile>333-333-3333</ows:Facsimile>
         </ows:Phone>
      </ows:ContactInfo>
   </ows:ServiceContact>
</ows:ServiceProvider>
<ows:OperationsMetadata xmlns:ows="http://www.opengis.net/ows/1.1"
                        xmlns:xlink="http://www.w3.org/1999/xlink">
   <ows:Operation name="GetCapabilities">
      <ows:DCP>
         <ows:HTTP>
            <ows:Get xlink:href="http://giv-genesis.uni-muenster
               .de:8080/SIR/sir?"/>
            <ows:Post xlink:href="http://giv-genesis.uni-muenster
              .de:8080/SIR/sir"/>
         </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="updateSequence">
         <ows:AnyValue/>
      </ows:Parameter>
      <ows:Parameter name="AcceptVersions">
         <ows:AllowedValues>
            <ows:Value>0.3.0</ows:Value>
         </ows:AllowedValues>
      </ows:Parameter>
      <ows:Parameter name="Sections">
         <ows:AllowedValues>
            <ows:Value>ServiceIdentification</ows:Value>
            <ows:Value>ServiceProvider</ows:Value>
            <ows:Value>OperationsMetadata</ows:Value>
            <ows:Value>Contents</ows:Value>
            <ows:Value>All</ows:Value>
         </ows:AllowedValues>
      </ows:Parameter>
      <ows:Parameter name="AcceptFormats">
         <ows:AllowedValues>
            <ows:Value>text/xml</ows:Value>
         </ows:AllowedValues>
      </ows:Parameter>
   </ows:Operation>
   ...
   ...
   ...
   <ows:Operation name="ConnectToCatalog">
      <ows:DCP>
         <ows:HTTP>
            <ows:Post xlink:href="http://giv-genesis.uni-
```

```
                                        muenster.de:8080/SIR/sir"/>
            </ows:HTTP>
         </ows:DCP>
         <ows:Parameter name="CatalogURL">
            <ows:AnyValue/>
         </ows:Parameter>
         <ows:Parameter name="PushIntervalSeconds">
            <ows:AnyValue/>
         </ows:Parameter>
      </ows:Operation>
      ...
      ...
      ...
      <ows:Parameter name="service">
         <ows:AllowedValues>
            <ows:Value>SIR</ows:Value>
         </ows:AllowedValues>
      </ows:Parameter>
      <ows:Parameter name="version">
         <ows:AllowedValues>
            <ows:Value>0.3.0</ows:Value>
         </ows:AllowedValues>
      </ows:Parameter>
   </ows:OperationsMetadata>
   <sir:Contents>
      <sir:HarvestedService>
         <sir:ServiceURL>
            http://v-swe.uni-muenster.de:8080/WeatherSOS/sos
         </sir:ServiceURL>
         <sir:ServiceType>SOS</sir:ServiceType>
      </sir:HarvestedService>
      <sir:HarvestedService>
         <sir:ServiceURL>
            http://mars.uni-muenster.de:8080/sos
         </sir:ServiceURL>
         <sir:ServiceType>SOS</sir:ServiceType>
      </sir:HarvestedService>
      <sir:LinkedCatalog>
         <sir:CatalogURL>
            http://giv-genesis.uni-muenster.de:8080/ergorr/webservice
         </sir:CatalogURL>
         <sir:Status>Last push finished without errors.</sir:Status>
         <sir:PushIntervalSeconds>3600</sir:PushIntervalSeconds>
      </sir:LinkedCatalog>
   </sir:Contents>
</sir:Capabilities>
```

7.3.1 Exceptions

When a SIR server encounters an error while performing a GetCapabilities operation, it shall return an exception report message as specified in Clause 8 of [OGC 06-121r3]. The allowed exception codes shall include those listed in Table 5 of Subclause 7.4.1 of [OGC 06-121r3], if the updateSequence parameter is implemented by the server.

8 SearchSensor operation (mandatory)

8.1 Introduction

The SearchSensor operation allows SIR clients to search for sensors. It can be used for querying sensors based on a broad range of parameters. The client receives an according set of metadata records describing the matching sensors, consisting of SensorML based metadata as well as information about the SWE service instances encapsulating the discovered sensors.

8.2 SearchSensor operation request

8.2.1 SearchSensor request parameters

A request to perform the SearchSensor operation shall include the data structure specified in Table 21.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 21 in Subclause 9.2.1 of [OGC 06-121r3].

Table 21 — Parameters in SearchSensor operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “SIR”	One (mandatory)
request	Operation name	Character String type, not empty Value is “SearchSensor”	One (mandatory)
version	Standard version for operation	Character String type, not empty Value is “0.3.0”	One (mandatory)
SensorIdentification	Identifier of a specific sensors that is explicitly requested in the search request	SIR SensorIdentification type	One or more (optional)
SearchCriteria	Criteria that shall be used when searching for sensors	SIR SearchCriteria type	One or more (optional)
SimpleResponse	Boolean value indicating if a SimpleSensorDescription (true) or a full SensorML document shall be returned (false)	Boolean type	Zero or one (optional)

a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 06-121r3].

8.2.2 SearchSensor request XML encoding (mandatory)

All SIR servers shall implement HTTP POST transfer of the SearchSensor operation request, using XML encoding.

A SearchSensor operation request is encoded in XML as shown in Figure 13.

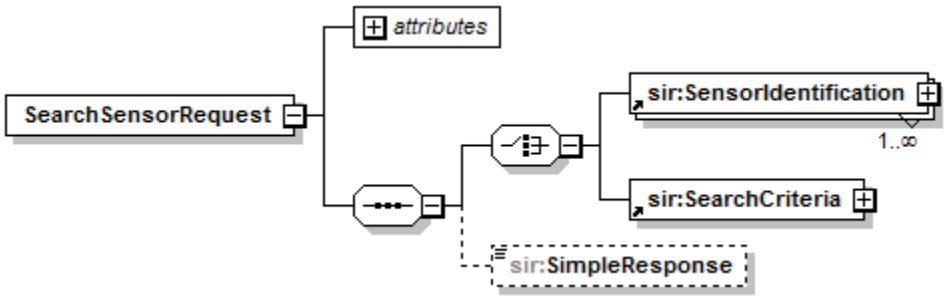

Figure 13: SearchSensor operation request shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a SearchSensor operation request encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           xmlns:ows="http://www.opengis.net/ows/1.1"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/ows/1.1"
              schemaLocation="http://schemas.opengis.net/ows/1.1.0
                 /owsCommon.xsd"/>
   <xs:include schemaLocation="sirCommon.xsd"/>
   <xs:element name="SearchSensorRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:choice>
               <xs:element ref="sir:SensorIdentification"
                           maxOccurs="unbounded"/>
               <xs:element ref="sir:SearchCriteria"/>
            </xs:choice>
            <xs:element name="SimpleResponse" type="xs:boolean"
                        minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="service" type="xs:string" use="required"
                       fixed="SIR"/>
         <xs:attribute name="version" type="xs:string" use="required"
                       fixed="0.3.0"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

EXAMPLE An example SearchSensor operation request XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<SearchSensorRequest service="SIR" version="0.3.0"
                     xmlns="http://swsl.uni-muenster.de/sir"
                     xmlns:ows="http://www.opengis.net/ows/1.1"
                     xmlns:swe="http://www.opengis.net/swe/1.0"
                     xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                     instance"
                     xsi:schemaLocation="http://swsl.uni-muenster.de
                     /sir
                     http://giv-genesis.uni-muenster.de/schemas/sir
                     /sirAll.xsd">
   <SearchCriteria>
      <ServiceCriteria>
         <ServiceType>SOS</ServiceType>
      </ServiceCriteria>
      <Phenomenon>
         <PhenomenonName>
            urn:x-ogc:def:property:OGC::RelativeHumidity
         </PhenomenonName>
      </Phenomenon>
      <BoundingBox crs="urn:ogc:def:crs:EPSG:6.14:4326" dimensions="2">
         <ows:LowerCorner>-180.0 -90.0 </ows:LowerCorner>
         <ows:UpperCorner>180.0 90.0 </ows:UpperCorner>
      </BoundingBox>
   </SearchCriteria>
</SearchSensorRequest>
```

8.3 SearchSensor operation response

8.3.1 Normal response parameters

The normal response to a valid SearchSensor operation request shall be a set of metadata records describing the sensors matching the SearchSensor request. More precisely, a response from the SearchSensor operation shall include a set of SearchResultElements. The parts of a SearchResultElement are listed in Table 22. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 22 — Parts of SearchResultElement operation response

Names	Definition	Data type and values	Multiplicity and use
SensorIDInSIR	String identifying the sensor within the SIR instance	Character String type, not empty	Zero or one (optional)
SensorDescription	Full description of the metadata of the discovered sensor	SIR SensorDescription	One (mandatory)
SimpleSensorDescription	Compact description of the metadata of the discovered sensor	SIR SimpleSensorDescriptionType	One (mandatory)
ServiceReference	Description how the sensor can be accessed through a specific SWE service instance	SIR ServiceReference type	Zero or more (optional)

8.3.2 Normal response XML encoding

A SearchSensor operation response is encoded in XML as shown in Figure 14.

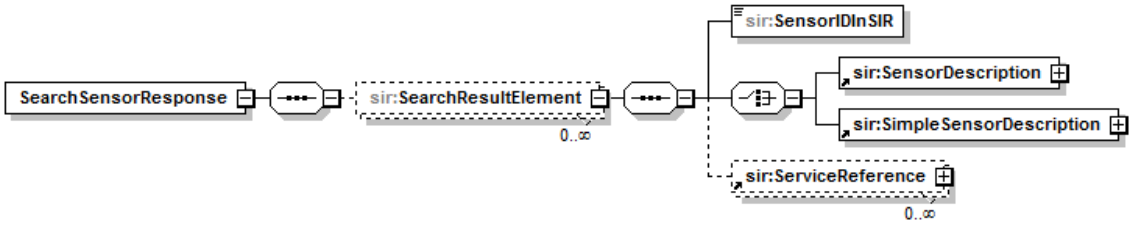

Figure 14: SearchSensor operation response shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a SearchSensor operation response, always encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           xmlns:ows="http://www.opengis.net/ows/1.1"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/ows/1.1"
              schemaLocation="http://schemas.opengis.net/ows/1.1.0
                 /owsCommon.xsd"/>
   <xs:include schemaLocation="sirCommon.xsd"/>
   <xs:element name="SearchSensorResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="SearchResultElement" minOccurs="0"
                        maxOccurs="unbounded">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="SensorIDInSIR"
                                 type="xs:string"/>
                        <xs:choice>
                           <xs:element ref="sir:SensorDescription"/>
                           <xs:element
                                    ref="sir:SimpleSensorDescription"/>
                        </xs:choice>
                     <xs:element ref="sir:ServiceReference"
                                 minOccurs="0" maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

8.3.3 SearchSensor response example

A SearchSensor operation response can look like this encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<sir:SearchSensorResponse xsi:schemaLocation="http://swsl.uni-muenster
                          .de/sir http://giv-genesis/schemas/sir
                          /sirAll.xsd
                          http://www.opengis.net/sensorML/1.0.1
                          http://schemas.opengis.net/sensorML/1.0.1
                          /sensorML.xsd"
                          xmlns:sir="http://swsl.uni-muenster.de/sir"
                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                          instance">
   <sir:SearchResultElement>
      <sir:SensorIDInSIR>9</sir:SensorIDInSIR>
      <sir:SimpleSensorDescription>
         <sir:SensorDescriptionURL>
            http://giv-genesis.uni-muenster.de:8080/SIR/sir
            ?service=SIR&amp;version=0.3.0&amp;
            request=DescribeSensor&amp;sensoridinsir=1
         </sir:SensorDescriptionURL>
         <sir:DescriptionText>
            This is a WS2500 weather station setup at the Institute for
            Geoinformatics.
         </sir:DescriptionText>
      </sir:SimpleSensorDescription>
   </sir:SearchResultElement>
</sir:SearchSensorResponse>
```

9 DescribeSensor operation (mandatory)

9.1 Introduction

The DescribeSensor operation allows SIR clients to request a SensorML document for a specific sensor contained in the SIR instance.

9.2 DescribeSensor operation request

9.2.1 DescribeSensor request parameters

A request to perform the DescribeSensor operation shall include the data structure specified in Table 23.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 21 in Subclause 9.2.1 of [OGC 06-121r3].

Table 23 — Parameters in DescribeSensor operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “SIR”	One (mandatory)
request	Operation name	Character String type, not empty Value is “DescribeSensor”	One (mandatory)
version	Standard version for operation	Character String type, not empty Value is “0.3.0”	One (mandatory)
SensorIDInSIR	String identifying the sensor within the SIR instance	Character String type, not empty	One (mandatory)

a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 06-121r3].

9.2.2 DescribeSensor request KVP encoding (mandatory)

Servers may implement HTTP GET transfer of the DescribeSensor operation request, using KVP encoding. The KVP encoding of the DescribeSensor operation request shall use the parameters specified in Table 24. The parameters listed in Table 24 shall be as specified in Table 23 above.

Table 24 — DescribeSensor operation request URL parameters

Name and example [a]	Optionality and use	Definition and format
service=SIR	Mandatory	Service type identifier
request= DescribeSensor	Mandatory	Operation name
version=0.3.0	Mandatory	Standard and schema version for this operation
sensorIDinSIR= sensor1	Mandatory	String identifying the sensor within the SIR instance

a All parameter names are here listed using mostly lower case letters. However, any parameter name capitalization shall be allowed in KVP encoding, see Subclause 11.5.2 of [OGC 06-121r3].

EXAMPLE An example DescribeSensor operation request KVP encoded for HTTP GET is:

```
http://giv-genesis.uni-muenster.de:8080/SIR/sir?request=DescribeSensor
&service=SIR&version=0.3.0&sensorIDinSIR=1
```

9.2.3 DescribeSensor request XML encoding (mandatory)

All SIR servers shall implement HTTP POST transfer of the DescribeSensor operation request, using XML encoding.

A DescribeSensor operation request is encoded in XML as shown in Figure 15.

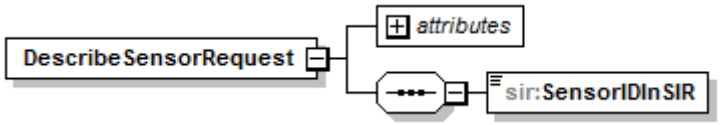

Figure 15: DescribeSensor operation request shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a DescribeSensor operation request encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           xmlns:ows="http://www.opengis.net/ows/1.1"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/ows/1.1"
              schemaLocation="http://schemas.opengis.net/ows/
                 1.1.0/owsCommon.xsd"/>
   <xs:include schemaLocation="sirCommon.xsd"/>
   <xs:element name="DescribeSensorRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="SensorIDInSIR" type="xs:string"/>
         </xs:sequence>
         <xs:attribute name="service" type="xs:string" use="required"
                       fixed="SIR"/>
         <xs:attribute name="version" type="xs:string" use="required"
                       fixed="0.3.0"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

EXAMPLE An example DescribeSensor operation request XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<DescribeSensorRequest service="SIR" version="0.3.0"
                       xmlns="http://swsl.uni-muenster.de/sir"
                       xmlns:swe="http://www.opengis.net/swe/1.0"
```



```
                        xmlns:xsi=http://www.w3.org/2001/XMLSchema-
                        instance
                        xsi:schemaLocation="http://swsl.uni-muenster
                        .de/sir
                        http://giv-genesis.uni-muenster.de/schemas/sir
                        /sirAll.xsd">
   <SensorIDInSIR>1</SensorIDInSIR>
</DescribeSensorRequest>
```

9.3 DescribeSensor operation response

9.3.1 Normal response parameters

The normal response to a valid DescribeSensor operation request shall be a SensorML document describing the requested sensor. In addition this SensorML document shall be compliant to the latest version of [OGC 09-163]

9.3.2 DescribeSensor response example

A DescribeSensor operation response can look like this encoded in XML (the SensorML document contained in the response is shortened):

```
<?xml version="1.0" encoding="UTF-8"?>
<SensorML xmlns="http://www.opengis.net/sensorML/1.0.1"
          xmlns:gml="http://www.opengis.net/gml"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          version="1.0.1"
          xsi:schemaLocation="http://www.opengis.net/sensorML/1.0.1
               http://schemas.opengis.net/sensorML/1.0.1/sensorML.xsd">
   <member>
      <System>
         ...
         ...
         ...
      </System>
   </member>
</SensorML>
```

10 HarvestService operation (mandatory)

10.1 Introduction

The HarvestService operation allows SIR clients to start a harvesting process during which a SIR instance retrieves all available sensor metadata from a specific SWE service instance.

In order to ensure the availability of all necessary metadata and to enable an automatic processing of the harvested metadata, the SWE service instance to be harvested shall comply to the SensorML profile for discovery as it is defined in the latest version of [OGC 09-163].

10.2 HarvestService operation request

10.2.1 HarvestService request parameters

A request to perform the HarvestService operation shall include the data structure specified in Table 25.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 21 in Subclause 9.2.1 of [OGC 06-121r3].

Table 25 — Parameters in HarvestService operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “SIR”	One (mandatory)
request	Operation name	Character String type, not empty Value is “HarvestService”	One (mandatory)
version	Standard version for operation	Character String type, not empty Value is “0.3.0”	One (mandatory)
ServiceURL	URL pointing to a SWE service instance that shall be harvested	Character string representing a URL	One (mandatory)
ServiceType	String describing the type of the SWE service that shall be harvested (i.e. “SOS”, “SPS”, “SAS” or “SES”)	Character String type, not empty	One (mandatory)

a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 06-121r3].

10.2.2 HarvestService request XML encoding (mandatory)

All SIR servers shall implement HTTP POST transfer of the HarvestService operation request, using XML encoding.

A HarvestService operation request is encoded in XML as shown in Figure 16.

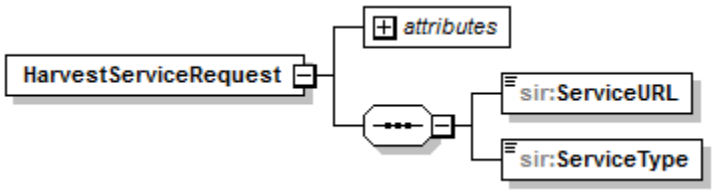

Figure 16: HarvestService operation request shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a HarvestService operation request encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:element name="HarvestServiceRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="ServiceURL" type="xs:anyURI"/>
            <xs:element name="ServiceType" type="xs:string"/>
         </xs:sequence>
         <xs:attribute name="service" type="xs:string" use="required"
                       fixed="SIR"/>
         <xs:attribute name="version" type="xs:string" use="required"
                       fixed="0.3.0"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

EXAMPLE An example HarvestService operation request XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<HarvestServiceRequest service="SIR" version="0.3.0"
                       xmlns="http://swsl.uni-muenster.de/sir"
                       xmlns:swe="http://www.opengis.net/swe/1.0"
                       xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                       instance"
                        xsi:schemaLocation="http://swsl.uni-muenster.
                           de/sir http://giv-genesis.uni-muenster.de/
                           schemas/sir/sirAll.xsd">
   <ServiceURL>
      http://v-swe.uni-muenster.de:8080/WeatherSOS/sos
   </ServiceURL>
   <ServiceType>SOS</ServiceType>
</HarvestServiceRequest>
```

10.3 HarvestService operation response

10.3.1 Normal response parameters

The normal response to a valid HarvestService operation operation request shall be a summary of the changes that were performed in the SIR database.. More precisely, a response from the HarvestService operation shall include the parts listed in Table 26. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 26 — Parts of HarvestService operation response

Names	Definition	Data type and values	Multiplicity and use
ServiceURL	URL pointing to a SWE service instance that was harvested	Character string representing a URL	One (mandatory)
ServiceType	String describing the type of the SWE service that was harvested (i.e. “SOS”, “SPS”, “SAS” or “SES”)	Character String type, not empty	One (mandatory)
NumberOfFoundSensors	Numer of sensors that were found in the harvested service	xs:int	One (mandatory)
NumberOfInsertedSensors	Numer of sensors that were inserted and that were previously not contained in the SIR instance	xs:int	One (mandatory)
NumberOfDeletedSensors	Numer of sensors that were removed from the SIR instance	xs:int	One (mandatory)
NumberOfUpdatedSensors	Numer of sensors that were updated with new metadata	xs:int	One (mandatory)
NumberOfFailedSensors	Numer of sensors that could not successfully harvested	xs:int	One (mandatory)
InsertedSensor	Information about a sensor that was newly inserted into the SIR instance	SIR InsertedSensor type	Zero or more (optional)
DeletedSensor	Information about a sensor that was removed from the SIR instance	SIR DeletedSensor type	Zero or more (optional)

UpdatedSensor	Information about a sensor that updated with new metadata	SIR UpdatedSensor type	Zero or more (optional)
FailedSensor	Information about a sensor that could not successfully beharvested	SIR FailedSensor type	Zero or more (optional)

Table 27 — Parameters of InsertedSensor

Names	Definition	Data type and values	Multiplicity and use
SensorIDInSIR	String identifying the sensor within the SIR instance	Character String type, not empty	One (mandatory)
ServiceSpecificSensorID	String identifying the sensor within the harvested SWE service instance	Character String type, not empty	One (mandatory)

Table 28 — Parameters of DeletedSensor

Names	Definition	Data type and values	Multiplicity and use
SensorIDInSIR	String identifying the sensor within the SIR instance	Character String type, not empty	One (mandatory)
ServiceSpecificSensorID	String identifying the sensor within the harvested SWE service instance	Character String type, not empty	One (mandatory)

Table 29 — Parameters of UpdatedSensor

Names	Definition	Data type and values	Multiplicity and use
SensorIDInSIR	String identifying the sensor within the SIR instance	Character String type, not empty	One (mandatory)
ServiceSpecificSensorID	String identifying the sensor within the harvested SWE service instance	Character String type, not empty	One (mandatory)

Table 30 — Parameters of FailedSensor

Names	Definition	Data type and values	Multiplicity and use
ServiceSpecificSensorID	String identifying the sensor within the harvested SWE service instance	Character String type, not empty	One (mandatory)
FailureDescription	String describing the failure during the harvesting process	Character String type, not empty	One (mandatory)

10.3.2 Normal response XML encoding

A HarvestService operation response is encoded in XML as shown in Figure 17.

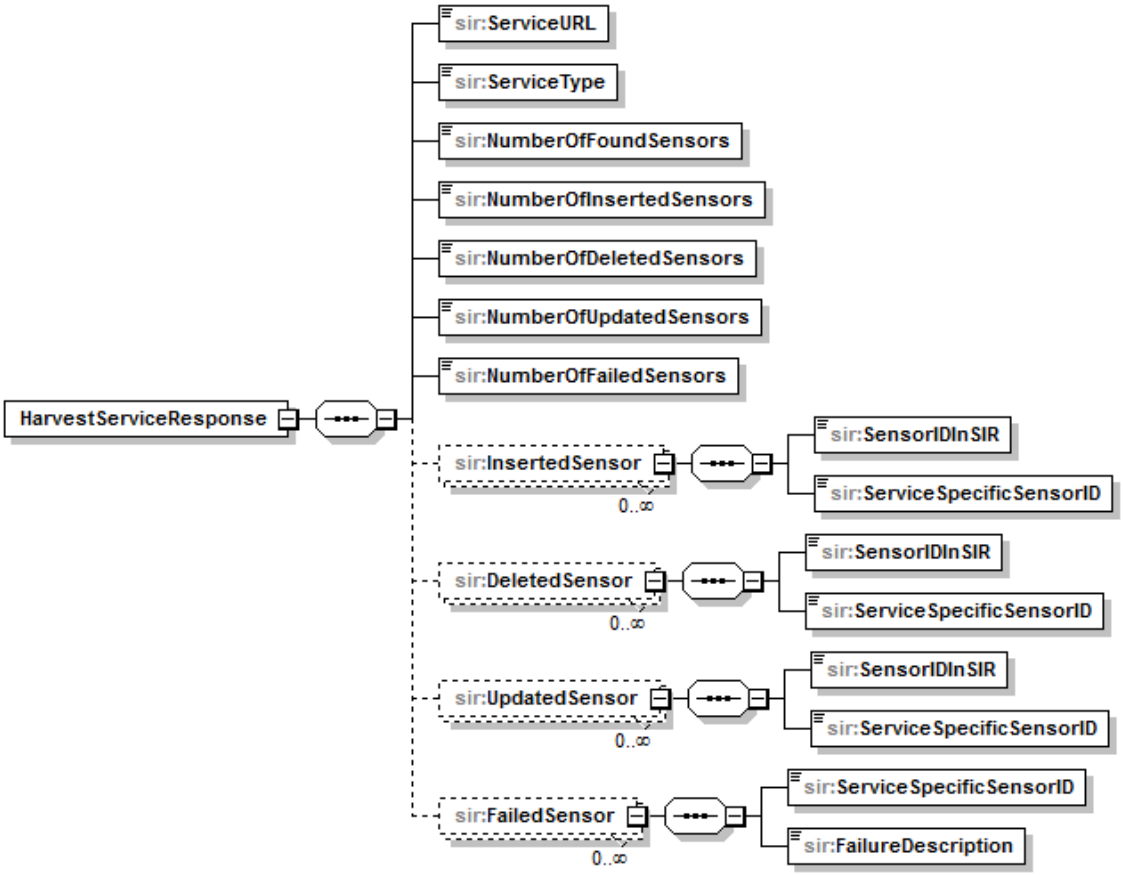

Figure 17: HarvestService operation response shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a HarvestService operation response, always encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:element name="HarvestServiceResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="ServiceURL" type="xs:anyURI"/>
            <xs:element name="ServiceType" type="xs:string"/>
            <xs:element name="NumberOfFoundSensors" type="xs:int"/>
            <xs:element name="NumberOfInsertedSensors" type="xs:int"/>
            <xs:element name="NumberOfDeletedSensors" type="xs:int"/>
            <xs:element name="NumberOfUpdatedSensors" type="xs:int"/>
```

```
            <xs:element name="NumberOfFailedSensors" type="xs:int"/>
            <xs:element name="InsertedSensor" minOccurs="0"
                        maxOccurs="unbounded">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="SensorIDInSIR"
                                 type="xs:string"/>
                     <xs:element name="ServiceSpecificSensorID"
                                 type="xs:string"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="DeletedSensor" minOccurs="0"
                        maxOccurs="unbounded">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="SensorIDInSIR"
                                 type="xs:string"/>
                     <xs:element name="ServiceSpecificSensorID"
                                 type="xs:string"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="UpdatedSensor" minOccurs="0"
                        maxOccurs="unbounded">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="SensorIDInSIR"
                                 type="xs:string"/>
                     <xs:element name="ServiceSpecificSensorID"
                                 type="xs:string"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="FailedSensor" minOccurs="0"
                        maxOccurs="unbounded">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="ServiceSpecificSensorID"
                                 type="xs:string"/>
                     <xs:element name="FailureDescription"
                                 type="xs:string"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

10.3.3 HarvestService response example

A HarvestService operation response can look like this encoded in XML:

```
<sir:HarvestServiceResponse xsi:schemaLocation="http://swsl.uni-
                            muenster.de/sir
```

```
                              http://giv-genesis/schemas/sir/sirAll.xsd
                              http://www.opengis.net/sensorML/1.0.1
                              http://schemas.opengis.net/sensorML/1.0.1
                              /sensorML.xsd"
                              xmlns:sir="http://swsl.uni-muenster.de/sir"
                              xmlns:xsi="http://www.w3.org/2001
                              /XMLSchema-instance">
   <sir:ServiceURL>
      http://v-swe.uni-muenster.de:8080/WeatherSOS/sos
   </sir:ServiceURL>
   <sir:ServiceType>SOS</sir:ServiceType>
   <sir:NumberOfFoundSensors>2</sir:NumberOfFoundSensors>
   <sir:NumberOfInsertedSensors>0</sir:NumberOfInsertedSensors>
   <sir:NumberOfDeletedSensors>0</sir:NumberOfDeletedSensors>
   <sir:NumberOfUpdatedSensors>2</sir:NumberOfUpdatedSensors>
   <sir:NumberOfFailedSensors>0</sir:NumberOfFailedSensors>
   <sir:UpdatedSensor>
      <sir:SensorIDInSIR>2</sir:SensorIDInSIR>
      <sir:ServiceSpecificSensorID>
         urn:ogc:object:feature:OSIRIS-HWS:3d3b239f-7696-4864-9d07-
         15447eae2b93
      </sir:ServiceSpecificSensorID>
   </sir:UpdatedSensor>
   <sir:UpdatedSensor>
      <sir:SensorIDInSIR>1</sir:SensorIDInSIR>
      <sir:ServiceSpecificSensorID>
         urn:ogc:object:feature:OSIRIS-HWS:efeb807b-bd24-4128-a920-
         f6729bcdd111
      </sir:ServiceSpecificSensorID>
   </sir:UpdatedSensor>
</sir:HarvestServiceResponse>
```

11 InsertSensorInfo operation (mandatory)

11.1 Introduction

The InsertSensorInfo operation allows SIR clients to insert a sensor description into the SIR database. Such a sensor description shall be based on SensorML. In order to ensure the completeness of the sensor metadata set and to enable the automatic processing of the metadata, such SensorML documents shall comply to the SensorML profile for discovery as it is defined in the latest version of [OGC 09-163].

11.2 InsertSensorInfo operation request

11.2.1 InsertSensorInfo request parameters

A request to perform the InsertSensorInfo operation shall include a list of the data structure specified in Table 31.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 21 in Subclause 9.2.1 of [OGC 06-121r3].

Table 31 — Parameters in InsertSensorInfo operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “SIR”	One (mandatory)
request	Operation name	Character String type, not empty Value is “InsertSensorInfo”	One (mandatory)
version	Standard version for operation	Character String type, not empty Value is “0.3.0”	One (mandatory)
SensorInfoToBeInserted	Sensor metadata to be inserted into the SIR instance	SIR SensorInfoToBeInserted type (see below)	One or more (mandatory)

a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 06-121r3].

A SensorInfoToBeInserted element can either consist of a SensorDescription and a number of ServiceReferences (in this case a new sensor with according references to SWE services is inserted) or of SensorIDInSIR and a number of ServiceReferences (in this case new references to SWE services for sensors already existing in the SIR are inserted).

Table 32 — Parameters in SensorInfoToBeInserted

Names [a]	Definition	Data type and values	Multiplicity and use
SensorDescription	Full description of the metadata of the discovered sensor	SIR SensorDescription	One (mandatory)
ServiceReference	Description how the sensor can be accessed through a specific SWE service instance	SIR ServiceReference type	Zero or more (optional)
SensorIDInSIR	String identifying the sensor within the SIR instance	Character String type, not empty	One (mandatory)
ServiceReference	Description how the sensor can be accessed through a specific SWE service instance	SIR ServiceReference type	One or more (optional)

11.2.2 InsertSensorInfo request XML encoding (mandatory)

All SIR servers shall implement HTTP POST transfer of the InsertSensorInfo operation request, using XML encoding.

An InsertSensorInfo operation request is encoded in XML as shown in Figure 18.

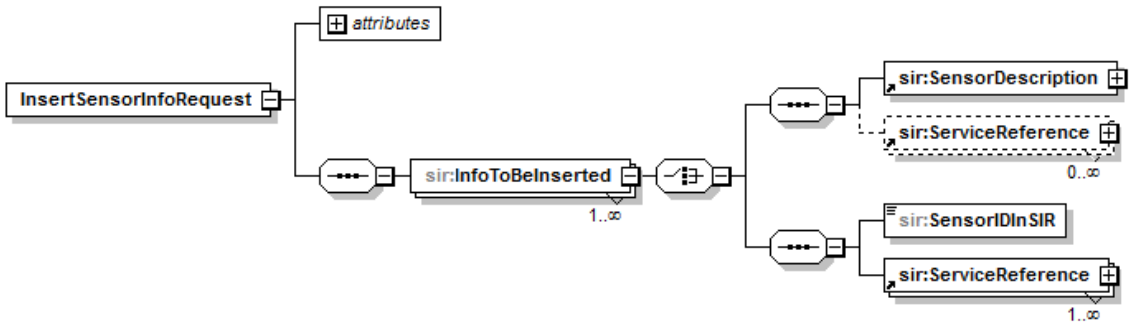

Figure 18: InsertSensorInfo operation request shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of an InsertSensorInfo operation request encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:include schemaLocation="sirCommon.xsd"/>
```

```
   <xs:element name="InsertSensorInfoRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="InfoToBeInserted" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:choice>
                     <xs:sequence>
                        <xs:element ref="sir:SensorDescription"/>
                        <xs:element ref="sir:ServiceReference"
                                    minOccurs="0"
                                    maxOccurs="unbounded"/>
                     </xs:sequence>
                     <xs:sequence>
                        <xs:element name="SensorIDInSIR"
                                    type="xs:string"/>
                        <xs:element ref="sir:ServiceReference"
                                    maxOccurs="unbounded"/>
                     </xs:sequence>
                  </xs:choice>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
         <xs:attribute name="service" type="xs:string" use="required"
                       fixed="SIR"/>
         <xs:attribute name="version" type="xs:string" use="required"
                       fixed="0.3.0"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

EXAMPLE An example InsertSensorInfo operation request for inserting a new sensor into a SIR instance XML encoded for HTTP POST is (SensorML section shortened):

```
<?xml version="1.0" encoding="UTF-8"?>
<InsertSensorInfoRequest service="SIR" version="0.3.0"
                         xmlns="http://swsl.uni-muenster.de/sir"
                         xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                         instance"
                         xmlns:gml="http://www.opengis.net/gml"
                         xmlns:swe="http://www.opengis.net/swe/1.0.1"
                         xsi:schemaLocation="http://swsl.uni-muenster
                         .de/sir
                         http://giv-genesis.uni-muenster.de/schemas/sir
                         /sirAll.xsd http://www.opengis.net/sensorML
                         /1.0.1
                         http://schemas.opengis.net/sensorML/1.0.1
                         /sensorML.xsd">
   <InfoToBeInserted>
      <SensorDescription
                       xmlns:ns="http://www.opengis.net/sensorML/1.0.1"
                       xsi:type="ns:SystemType">
         ...
         ...
         ...
      </SensorDescription>
      <ServiceReference>
         <ServiceURL>http://my.service.url/sos</ServiceURL>
         <ServiceType>SOS</ServiceType>
```

```
         <ServiceSpecificSensorID>
            id:sensor:42
         </ServiceSpecificSensorID>
      </ServiceReference>
   </InfoToBeInserted>
</InsertSensorInfoRequest>
```

EXAMPLE An example InsertSensorInfo operation request for inserting a new service reference for a sensor XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<InsertSensorInfoRequest service="SIR" version="0.3.0"
                         xmlns="http://swsl.uni-muenster.de/sir"
                         xmlns:swe="http://www.opengis.net/swe/1.0"
                         xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                         instance"
                         xsi:schemaLocation="http://swsl.uni-muenster.
                            de/sir http://giv-genesis.uni-muenster.de/
                            schemas/sir/sirAll.xsd
                            http://www.opengis.net/sensorML/1.0.1
                            http://schemas.opengis.net/sensorML/1.0.1/
                            sensorML.xsd">
   <InfoToBeInserted>
      <SensorIDInSIR>1</SensorIDInSIR>
      <ServiceReference>
         <ServiceURL>http://my.service.url/sps</ServiceURL>
         <ServiceType>SPS</ServiceType>
         <ServiceSpecificSensorID>id:017</ServiceSpecificSensorID>
      </ServiceReference>
   </InfoToBeInserted>
</InsertSensorInfoRequest>
```

11.3 InsertSensorInfo operation response

11.3.1 Normal response parameters

The normal response to a valid InsertSensorInfo operation request shall be a summary of the changes that were performed in the SIR database. More precisely, a response from the InsertSensorInfo operation shall include the parts listed in Table 33. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 33 — Parts of InsertSensorInfo operation response

Names	Definition	Data type and values	Multiplicity and use
NumberOfInsertedSensors	Numer of sensors that were inserted and that were previously not contained in the SIR instance	xs:int	One (mandatory)
NumberOfInsertedService References	Numer of new references to SWE service instances that were inserted into the SIR for the sensor	xs:int	One (mandatory)

SensorIDInSIR	String identifying the sensor within the SIR instance	Character String type, not empty	One (mandatory)

11.3.2 Normal response XML encoding

An InsertSensorInfo operation response is encoded in XML as shown in Figure 19.

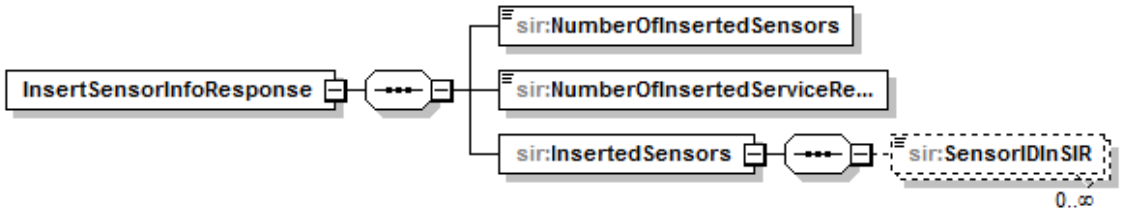

Figure 19: InsertSensorInfo operation response shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of an InsertSensorInfo operation response, always encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:include schemaLocation="sirCommon.xsd"/>
   <xs:element name="InsertSensorInfoResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="NumberOfInsertedSensors" type="xs:int"/>
            <xs:element name="NumberOfInsertedServiceReferences"
                        type="xs:int"/>
            <xs:element name="InsertedSensors">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="SensorIDInSIR" type="xs:string"
                                 minOccurs="0" maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

11.3.3 InsertSensorInfo response example

An InsertSensorInfo operation response can look like this encoded in XML:

```
<sir:InsertSensorInfoResponse xsi:schemaLocation="http://swsl.uni-
                              muenster.de/sir
                              http://giv-genesis/schemas/sir/sirAll.xsd
```

```
                                http://www.opengis.net/sensorML/1.0.1
                                http://schemas.opengis.net/sensorML/1.0.1
                                /sensorML.xsd"
                                xmlns:sir="http://swsl.uni-
                                muenster.de/sir"
                                xmlns:xsi="http://www.w3.org/2001
                                /XMLSchema-instance">
   <sir:NumberOfInsertedSensors>1</sir:NumberOfInsertedSensors>
   <sir:NumberOfInsertedServiceReferences>
      0
   </sir:NumberOfInsertedServiceReferences>
   <sir:InsertedSensors>
      <sir:SensorIDInSIR>23</sir:SensorIDInSIR>
   </sir:InsertedSensors>
</sir:InsertSensorInfoResponse>
```

12 DeleteSensorInfo operation (mandatory)

12.1 Introduction

The DeleteSensorInfo operation allows SIR clients delete specific sensor instances or SWE service instances from a SIR instance.

12.2 DeleteSensorInfo operation request

12.2.1 DeleteSensorInfo request parameters

A request to perform the DeleteSensorInfo operation shall include the data structure specified in Table 34.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 21 in Subclause 9.2.1 of [OGC 06-121r3].

Table 34 — Parameters in DeleteSensorInfo operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “SIR”	One (mandatory)
request	Operation name	Character String type, not empty Value is “DeleteSensorInfo”	One (mandatory)
version	Standard version for operation	Character String type, not empty Value is “0.3.0”	One (mandatory)
InfoToBeDeleted	Sensor metadata to be deleted from the SIR instance	SIR SensorInfoToBeDeleted type (see below)	One or more (mandatory)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 06-121r3].			

An InfoToBeDeleted element consists of an identifier of a sensor within the SIR instance and either a boolean value if the sensor shall be completely deleted (in case the parameter true) or it is the set of references to SWE service instances that shall be deleted (in case the parameter is false).

Table 35 — Parameters in InfoToBeDeleted

Names [a]	Definition	Data type and values	Multiplicity and use
SensorIdentification	Identifier of a specific sensors that is explicitly requested in the search request	SIR SensorIdentification type	One (mandatory)
DeleteSensor	Boolean value indicating if all metadata for the sensor shall be deleted from the SIR	xs:boolean	Zero or one (optional)

ServiceReference	Description how the sensor can be accessed through a specific SWE service instance	SIR ServiceReference type	Zero or one (optional)

12.2.2 DeleteSensorInfo request XML encoding (mandatory)

All SIR servers shall implement HTTP POST transfer of the DeleteSensorInfo operation request, using XML encoding.

A DeleteSensorInfo operation request is encoded in XML as shown in Figure 20.

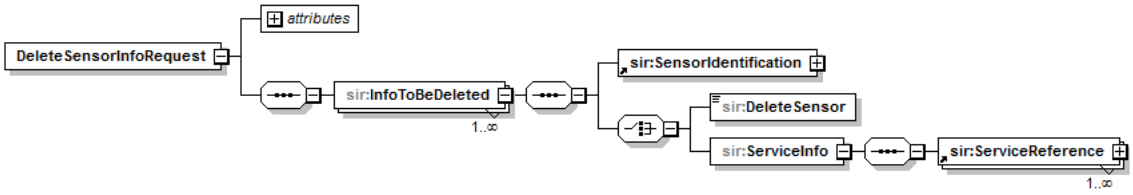

Figure 20: DeleteSensorInfo operation request shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a DeleteSensorInfo operation request encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:include schemaLocation="sirCommon.xsd"/>
   <xs:element name="DeleteSensorInfoRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="InfoToBeDeleted" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="sir:SensorIdentification"/>
                     <xs:choice>
                        <xs:element name="DeleteSensor"
                                    type="xs:boolean"/>
                        <xs:element name="ServiceInfo">
                           <xs:complexType>
                              <xs:sequence>
                                 <xs:element ref="sir:ServiceReference"
                                             maxOccurs="unbounded"/>
                              </xs:sequence>
                           </xs:complexType>
                        </xs:element>
                     </xs:choice>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
```

```
         </xs:sequence>
         <xs:attribute name="service" type="xs:string" use="required"
                       fixed="SIR"/>
         <xs:attribute name="version" type="xs:string" use="required"
                       fixed="0.3.0"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

EXAMPLE An example DeleteSensorInfo operation request for deleting all metadata of a sensor XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<DeleteSensorInfoRequest service="SIR" version="0.3.0"
                         xmlns="http://swsl.uni-muenster.de/sir"
                         xmlns:swe="http://www.opengis.net/swe/1.0"
                         xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                         instance"
                         xsi:schemaLocation="http://swsl.uni-muenster
                         .de/sir
                         http://giv-genesis.uni-muenster.de/schemas/sir
                         /sirAll.xsd">
   <InfoToBeDeleted>
      <SensorIdentification>
         <SensorIDInSIR>3</SensorIDInSIR>
      </SensorIdentification>
      <DeleteSensor>true</DeleteSensor>
   </InfoToBeDeleted>
</DeleteSensorInfoRequest>
```

EXAMPLE An example DeleteSensorInfo operation request for deleting a service reference of a sensor XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<DeleteSensorInfoRequest service="SIR" version="0.3.0"
                         xmlns="http://swsl.uni-muenster.de/sir"
                         xmlns:swe="http://www.opengis.net/swe/1.0"
                         xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                         instance"
                         xsi:schemaLocation="http://swsl.uni-muenster
                         .de/sir
                         http://giv-genesis.uni-muenster.de/schemas/sir
                         /sirAll.xsd">
   <InfoToBeDeleted>
      <SensorIdentification>
         <SensorIDInSIR>1</SensorIDInSIR>
      </SensorIdentification>
      <ServiceInfo>
         <ServiceReference>
            <ServiceURL>http://my.service.url/sos</ServiceURL>
            <ServiceType>SOS</ServiceType>
            <ServiceSpecificSensorID>id:017</ServiceSpecificSensorID>
         </ServiceReference>
      </ServiceInfo>
   </InfoToBeDeleted>
</DeleteSensorInfoRequest>
```

12.3 DeleteSensorInfo operation response

12.3.1 Normal response parameters

The normal response to a valid DeleteSensorInfo operation request shall be a summary of the changes that were performed in the SIR database. More precisely, a response from the DeleteSensorInfo operation shall include the parts listed in Table 36. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 36 — Parts of DeleteSensorInfo operation response

Names	Definition	Data type and values	Multiplicity and use
NumberOfDeletedSensors	Numer of sensors that were removed from the SIR instance	xs:int	One (mandatory)
NumberOfDeletedServiceReferences	Numer of references to SWE service instances that were removed from the SIR	xs:int	One (mandatory)
SensorIDInSIR	String identifying a sensor within the SIR instance that was deleted	Character String type, not empty	One (mandatory)

12.3.2 Normal response XML encoding

A DeleteSensorInfo operation response is encoded in XML as shown in Figure 21.

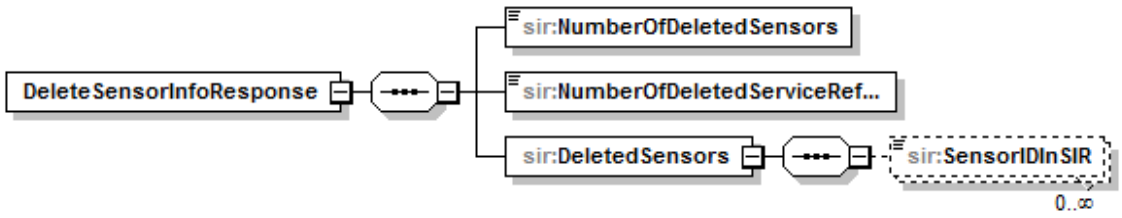

Figure 21: DeleteSensorInfo operation response shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a DeleteSensorInfo operation response, always encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:include schemaLocation="sirCommon.xsd"/>
   <xs:element name="DeleteSensorInfoResponse">
      <xs:complexType>
```

```
         <xs:sequence>
            <xs:element name="NumberOfDeletedSensors" type="xs:int"/>
            <xs:element name="NumberOfDeletedServiceReferences"
                        type="xs:int"/>
            <xs:element name="DeletedSensors">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="SensorIDInSIR" type="xs:string"
                                 minOccurs="0" maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

12.3.3 DeleteSensorInfo response example

A DeleteSensorInfo operation response can look like this encoded in XML:

```
<sir:DeleteSensorInfoResponse xsi:schemaLocation="http://swsl.uni-
                              muenster.de/sir
                              http://giv-genesis/schemas/sir/sirAll.xsd
                              http://www.opengis.net/sensorML/1.0.1
                              http://schemas.opengis.net/sensorML/1.0.1
                              /sensorML.xsd"
                              xmlns:sir="http://swsl.uni-
                              muenster.de/sir"
                              xmlns:xsi="http://www.w3.org/2001
                              /XMLSchema-instance">
   <sir:NumberOfDeletedSensors>1</sir:NumberOfDeletedSensors>
   <sir:NumberOfDeletedServiceReferences>
      0
   </sir:NumberOfDeletedServiceReferences>
   <sir:DeletedSensors>
      <sir:SensorIDInSIR>9</sir:SensorIDInSIR>
   </sir:DeletedSensors>
</sir:DeleteSensorInfoResponse>
```

13 UpdateSensorDescription operation (mandatory)

13.1 Introduction

The UpdateSensorDescription operation allows SIR clients to update the metadata of a sensor or SWE service instance within the SIR.

13.2 UpdateSensorDescription operation request

13.2.1 UpdateSensorDescription request parameters

A request to perform the UpdateSensorDescription operation shall include the data structure specified in Table 37.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 21 in Subclause 9.2.1 of [OGC 06-121r3].

Table 37 — Parameters in UpdateSensorDescription operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “SIR”	One (mandatory)
request	Operation name	Character String type, not empty Value is “UpdateSensorDescription”	One (mandatory)
version	Standard version for operation	Character String type, not empty Value is “0.3.0”	One (mandatory)
SensorDescriptionToBeUpdated	Sensor description (SensorML documents) to be updated in the SIR instance	SIR SensorDescriptionToBeUpdatedtype (see below)	One or more (mandatory)

a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 06-121r3].

Table 38 — Parameters in SensorDescriptionToBeUpdated

Names [a]	Definition	Data type and values	Multiplicity and use
SensorIdentification	Identifier of a specific sensors that is explicitly requested in the search request	SIR SensorIdentification type	One (mandatory)
SensorDescription	Full description of the metadata of the discovered sensor	SIR SensorDescription	One (mandatory)

13.2.2 UpdateSensorDescription request XML encoding (mandatory)

All SIR servers shall implement HTTP POST transfer of the UpdateSensorDescription operation request, using XML encoding.

An UpdateSensorDescription operation request is encoded in XML as shown in Figure 22.

Figure 22: UpdateSensorDescription operation request shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of an UpdateSensorDescription operation request encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:include schemaLocation="sirCommon.xsd"/>
   <xs:element name="UpdateSensorDescriptionRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="SensorDescriptionToBeUpdated"
                        maxOccurs="unbounded">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element ref="sir:SensorIdentification"/>
                     <xs:element ref="sir:SensorDescription"/>
                  </xs:sequence>
                  <xs:attribute name="service" type="xs:string"
                                use="required" fixed="SIR"/>
                  <xs:attribute name="version" type="xs:string"
                                use="required" fixed="0.3.0"/>
               </xs:complexType>
```

```
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

EXAMPLE An example UpdateSensorDescription operation request XML encoded for HTTP POST is (SensorML section shortened):

```
<?xml version="1.0" encoding="UTF-8"?>
<UpdateSensorDescriptionRequest service="SIR" version="0.3.0"
                                xmlns="http://swsl.uni-muenster.de/sir"
                                xmlns:gml="http://www.opengis.net/gml"
                                xmlns:swe="http://www.opengis.net
                                /swe/1.0.1"
                                xmlns:xsi="http://www.w3.org/2001
                                /XMLSchema-instance"
                                xsi:schemaLocation="http://swsl.uni-
                                muenster.de/sir
                                http://giv-genesis.uni-muenster.de
                                /schemas/sir/sirAll.xsd
                                http://www.opengis.net/sensorML/1.0.1
                                http://schemas.opengis.net/sensorML
                                /1.0.1/sensorML.xsd">
   <SensorDescriptionToBeUpdated>
      <SensorIdentification>
         <SensorIDInSIR>1</SensorIDInSIR>
      </SensorIdentification>
      <SensorDescription xsi:type="sml:SystemType"
                         xmlns:sml="http://www.opengis.net
                         /sensorML/1.0.1">
         ...
         ...
         ...
      </SensorDescription>
   </SensorDescriptionToBeUpdated>
</UpdateSensorDescriptionRequest>
```

13.3 UpdateSensorDescription operation response

13.3.1 Normal response parameters

The normal response to a valid UpdateSensorDescription operation request shall be a summary of the changes that were performed in the SIR database. More precisely, a response from the UpdateSensorDescription operation shall include the parts listed in Table 39. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 39 — Parts of UpdateSensorDescription operation response

Names	Definition	Data type and values	Multiplicity and use
NumberOfUpdatedSensors	Numer of sensors that were updated with new metadata	xs:int	One (mandatory)

SensorIDInSIR	String identifying a sensor within the SIR instance that was updated	Character String type, not empty	Zero or more (optional)

13.3.2 Normal response XML encoding

An UpdateSensorDescription operation response is encoded in XML as shown in Figure 23.

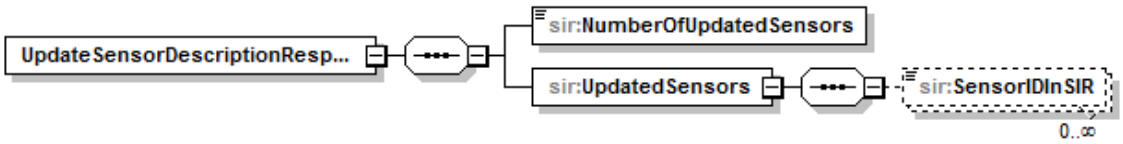

Figure 23: UpdateSensorDescription operation response shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a UpdateSensorDescription operation response, always encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:include schemaLocation="sirCommon.xsd"/>
   <xs:element name="UpdateSensorDescriptionResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="NumberOfUpdatedSensors" type="xs:int"/>
            <xs:element name="UpdatedSensors">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="SensorIDInSIR" type="xs:string"
                                 minOccurs="0" maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

13.3.3 UpdateSensorDescription response example

An UpdateSensorDescription operation response can look like this encoded in XML:

```
<sir:UpdateSensorDescriptionResponse xsi:schemaLocation="
                              http://swsl.uni-muenster.de/sir
                              http://giv-genesis/schemas/sir/sirAll.xsd
                              http://www.opengis.net/sensorML/1.0.1
```

```
                                http://schemas.opengis.net/sensorML
                                /1.0.1/sensorML.xsd"
                                xmlns:sir="http://swsl.uni-
                                muenster.de/sir"
                                xmlns:xsi="http://www.w3.org/2001
                                /XMLSchema-instance">
   <sir:NumberOfUpdatedSensors>1</sir:NumberOfUpdatedSensors>
   <sir:UpdatedSensors>
      <sir:SensorIDInSIR>9</sir:SensorIDInSIR>
   </sir:UpdatedSensors>
</sir:UpdateSensorDescriptionResponse>
```

14 GetSensorStatus operation (optional)

14.1 Introduction

The GetSensorStatus operation allows SIR clients to retrieve information about the status of a specific sensor. This operation offers a broad range of different query parameters for specifying in which sensors a client wants to receice status information.

14.2 GetSensorStatus operation request

14.2.1 GetSensorStatus request parameters

A request to perform the GetSensorStatus operation shall include the data structure specified in Table 40.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 21 in Subclause 9.2.1 of [OGC 06-121r3].

Table 40 — Parameters in GetSensorStatus operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is "SIR"	One (mandatory)
request	Operation name	Character String type, not empty Value is "GetSensorStatus"	One (mandatory)
version	Standard version for operation	Character String type, not empty Value is "0.3.0"	One (mandatory)
SensorIdentification	Identifier of a specific sensors that is explicitly requested in the GetSensorStatus request	SIR SensorIdentification type	Zero or one (optional)
SearchCriteria	Criteria that shall be used when searching for sensors	SIR SearchCriteria type	One or one (optional)
PropertyFilter	Constraints on the status information that shall be considered when returning the status of sensors (only the status of senosrs fulfilling these constraints will be returned)	SIR PropertyFilter type	One or more (optional)

a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 06-121r3].

14.2.2 GetSensorStatus request XML encoding (mandatory)

All SIR servers supporting the GetSensorStatus operation shall implement HTTP POST transfer of the GetSensorStatus operation request, using XML encoding.

A GetSensorStatus operation request is encoded in XML as shown in Figure 24.

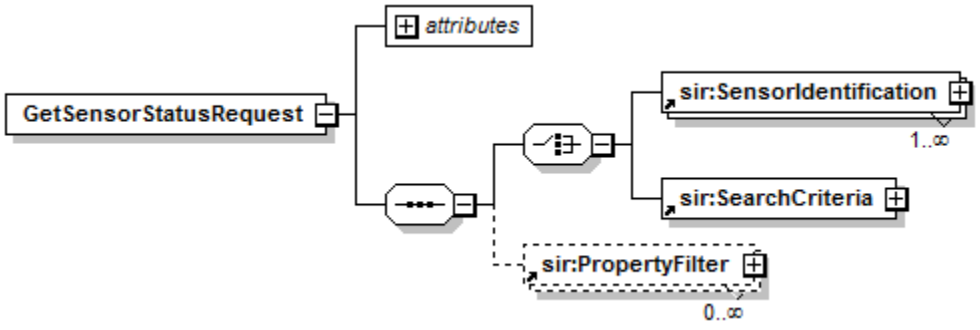

Figure 24: GetSensorStatus operation request shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a GetSensorStatus operation request encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           xmlns:ows="http://www.opengis.net/ows/1.1"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/ows/1.1"
              schemaLocation=
                 "http://schemas.opengis.net/ows/1.1.0/owsCommon.xsd"/>
   <xs:include schemaLocation="sirCommon.xsd"/>
   <xs:element name="GetSensorStatusRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:choice>
               <xs:element ref="sir:SensorIdentification"
                           maxOccurs="unbounded"/>
               <xs:element ref="sir:SearchCriteria"/>
            </xs:choice>
            <xs:element ref="sir:PropertyFilter" minOccurs="0"
                        maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attribute name="service" type="xs:string" use="required"
                       fixed="SIR"/>
         <xs:attribute name="version" type="xs:string" use="required"
                       fixed="0.3.0"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

EXAMPLE An example GetSensorStatus operation request XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetSensorStatusRequest service="SIR" version="0.3.0"
                        xmlns="http://swsl.uni-muenster.de/sir"
                        xmlns:ows="http://www.opengis.net/ows/1.1"
                        xmlns:swe="http://www.opengis.net/swe/1.0"
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                        instance"
                        xsi:schemaLocation="http://swsl.uni-
                        muenster.de/sir
                        http://giv-genesis.uni-muenster.de/schemas/sir
                        /sirAll.xsd">
   <SensorIdentification>
      <SensorIDInSIR>1</SensorIDInSIR>
   </SensorIdentification>
   <PropertyFilter>
      <PropertyName>batterystate</PropertyName>
      <PropertyConstraint>
         <Constraint>
            <IsEqualTo>full</IsEqualTo>
         </Constraint>
      </PropertyConstraint>
   </PropertyFilter>
</GetSensorStatusRequest>
```

14.3 GetSensorStatus operation response

14.3.1 Normal response parameters

The normal response to a valid GetSensorStatus operation request shall be a set of StatusDescription elements describing the status of those sensors matching the GetSensorStatus request. More precisely, a response from a StatusDescription element shall include the parts listed in Table 41. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 41 — Parameters of StatusDescription

Names	Definition	Data type and values	Multiplicity and use
SensorIDInSIR	String identifying a sensor for which the status is described.	Character String type, not empty	One (mandatory)
Status	Status description of the sensor	SIR Status type	One or more (mandatory)

14.3.2 Normal response XML encoding

A GetSensorStatus operation response is encoded in XML as shown in Figure 25.

Figure 25: GetSensorStatus operation response shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a GetSensorStatus operation response, always encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           xmlns:ows="http://www.opengis.net/ows/1.1"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/ows/1.1"
              schemaLocation=
                 "http://schemas.opengis.net/ows/1.1.0/owsCommon.xsd"/>
   <xs:include schemaLocation="sirCommon.xsd"/>
   <xs:element name="GetSensorStatusResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="sir:StatusDescription" minOccurs="0"
                                                    maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

14.3.3 GetSensorStatus response example

A GetSensorStatus operation response can look like this encoded in XML:

```
<sir:GetSensorStatusResponse xsi:schemaLocation="http://swsl.uni-
                             muenster.de/sir
                             http://giv-genesis/schemas/sir/sirAll.xsd
                             http://www.opengis.net/sensorML/1.0.1
                             http://schemas.opengis.net/sensorML
                             /1.0.1/sensorML.xsd"
                             xmlns:sir="http://swsl.uni-muenster.de
                             /sir"
                             xmlns:xsi="http://www.w3.org/2001
                             /XMLSchema-instance"
                             xmlns:ns="http://www.opengis.net/gml/3.2">
   <sir:StatusDescription>
      <sir:SensorIDInSIR>1</sir:SensorIDInSIR>
      <sir:Status>
         <sir:PropertyName>batteryHoursLeft</sir:PropertyName>
         <sir:PropertyValue>42</sir:PropertyValue>
         <sir:Uom code="hr"/>
         <sir:Timestamp ns:id="id161">
            <ns:timePosition>2010-09-01T00:00:00+0200</ns:timePosition>
```

```
            </sir:Timestamp>
         </sir:Status>
      </sir:StatusDescription>
   </sir:GetSensorStatusResponse>
```

15 SubscribeSensorStatus operation (optional)

15.1 Introduction

The SubscribeSensorStatus operation allows SIR clients to subscribe to status information for a selected set of sensors. This operation offers a broad range of different query parameters for specifying in which sensors a client wants to receice status information.

After the subscription request has been received by the SIR server, all status information matching the SubscribeSensorStatus request will automatically transmitted to the notification target specified in the request.

15.2 SubscribeSensorStatus operation request

15.2.1 SubscribeSensorStatus request parameters

A request to perform the SubscribeSensorStatus operation shall include the data structure specified in Table 42.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 21 in Subclause 9.2.1 of [OGC 06-121r3].

Table 42 — Parameters in SubscribeSensorStatus operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is "SIR"	One (mandatory)
request	Operation name	Character String type, not empty Value is "SubscribeSensorStatus"	One (mandatory)
version	Standard version for operation	Character String type, not empty Value is "0.3.0"	One (mandatory)
SubscriptionTarget	Information where the sensor status information shall be sent to	SIR SubscriptionTarget	One (mandatory)
SensorIdentification	Identifier of a specific sensors that is explicitly requested in the SubscribeSensorStatus request	SIR SensorIdentification type	Zero or one (optional)
SearchCriteria	Criteria that shall be used when searching for sensors	SIR SearchCriteria type	One or one (optional)
PropertyFilter	Constraints on the status information that shall be considered when returning the status of sensors (only the status of senosrs fulfilling these constraints will be returned)	SIR PropertyFilter type	One or more (optional)

a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 06-121r3].

Table 43 — Parameters of SubscriptionTarget

Names	Definition	Data type and values	Multiplicity and use
NotificationID	String identifying the receiver of the sensor status information under the NotificationURL	Character String type, not empty	One (mandatory)
NotificationURL	String representing a URL to which notification requests shall be directed (e.g. a WNS instance)	Character string representing a URL	One (mandatory

15.2.2 SubscribeSensorStatus request XML encoding (mandatory)

All SIR servers supporting the SubscribeSensorStatus operation shall implement HTTP POST transfer of the SubscribeSensorStatus operation request, using XML encoding.

A SubscribeSensorStatus operation request is encoded in XML as shown in Figure 26.

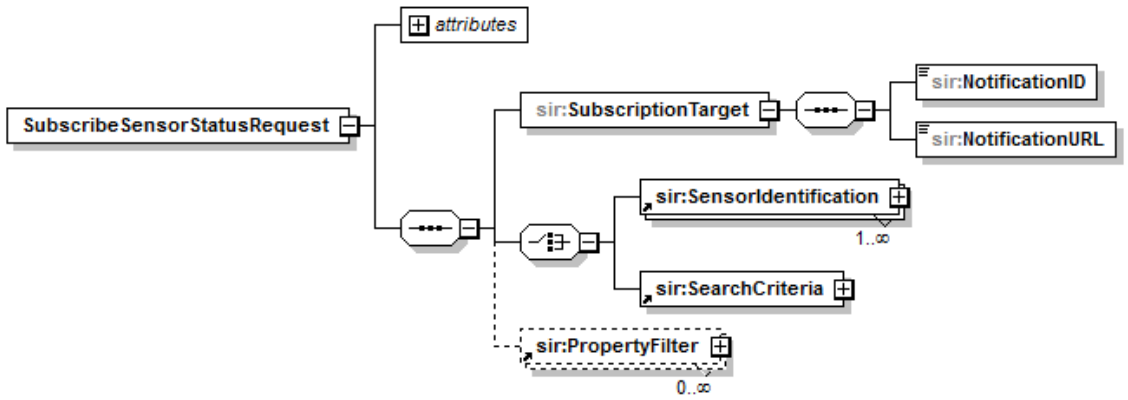

Figure 26: SubscribeSensorStatus operation request shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a SubscribeSensorStatus operation request encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           xmlns:ows="http://www.opengis.net/ows/1.1"
           xmlns:gml="http://schemas.opengis.net/gml/3.2.1/"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/ows/1.1"
              schemaLocation=
                 "http://schemas.opengis.net/ows/1.1.0/owsCommon.xsd"/>
   <xs:include schemaLocation="sirCommon.xsd"/>
   <xs:element name="SubscribeSensorStatusRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="SubscriptionTarget">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="NotificationID"
                                 type="xs:string"/>
                     <xs:element name="NotificationURL"
                                 type="xs:anyURI"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:choice>
               <xs:element ref="sir:SensorIdentification"
                           maxOccurs="unbounded"/>
               <xs:element ref="sir:SearchCriteria"/>
```

```
            </xs:choice>
            <xs:element ref="sir:PropertyFilter" minOccurs="0"
                        maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attribute name="service" type="xs:string" use="required"
                       fixed="SIR"/>
         <xs:attribute name="version" type="xs:string" use="required"
                       fixed="0.3.0"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

EXAMPLE An example SubscribeSensorStatus operation request XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<SubscribeSensorStatusRequest service="SIR" version="0.3.0"
                              xmlns="http://swsl.uni-muenster.de/sir"
                              xmlns:xsi="http://www.w3.org/2001
                              /XMLSchema-instance"
                              xsi:schemaLocation="http://swsl.uni-
                              muenster.de/sir
                              http://giv-genesis.uni-muenster.de
                              /schemas/sir/sirAll.xsd">
   <SubscriptionTarget>
      <NotificationID>001</NotificationID>
      <NotificationURL>
         http://giv-genesis.uni-muenster.de/notificationservice/notify
      </NotificationURL>
   </SubscriptionTarget>
   <SensorIdentification>
      <SensorIDInSIR>1</SensorIDInSIR>
   </SensorIdentification>
</SubscribeSensorStatusRequest>
```

15.3 SubscribeSensorStatus operation response

15.3.1 Normal response parameters

The normal response to a valid SubscribeSensorStatus operation request shall be information about the subscription (i.e. subscription id). More precisely, a response from the SubscribeSensorStatus operation shall include the parts listed in Table 44. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 44 — Parts of SubscribeSensorStatus operation response

Names	Definition	Data type and values	Multiplicity and use
SubscriptionID	Identifier of the subscription (needed for managing the subscription)	Character String type, not empty	One (mandatory)
ExpirationDate	Date when the subscription will expire.	xs:dateTime	One (mandatory)

15.3.2 Normal response XML encoding

A SubscribeSensorStatus operation response is encoded in XML as shown in Figure 27.

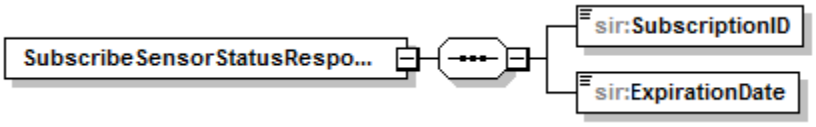

Figure 27: SubscribeSensorStatus operation response shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a SubscribeSensorStatus operation response, always encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           xmlns:ows="http://www.opengis.net/ows/1.1"
           xmlns:gml="http://schemas.opengis.net/gml/3.2.1/"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/ows/1.1"
              schemaLocation=
                 "http://schemas.opengis.net/ows/1.1.0/owsCommon.xsd"/>
   <xs:include schemaLocation="sirCommon.xsd"/>
   <xs:element name="SubscribeSensorStatusResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="SubscriptionID" type="xs:string"/>
            <xs:element name="ExpirationDate" type="xs:dateTime"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

15.3.3 SubscribeSensorStatus response example

A SubscribeSensorStatus operation response can look like this encoded in XML:

```
<sir:SubscribeSensorStatusResponse xsi:schemaLocation="
                              http://swsl.uni-muenster.de/sir
                              http://giv-genesis/schemas/sir/sirAll.xsd
                              http://www.opengis.net/sensorML/1.0.1
                              http://schemas.opengis.net/sensorML/1.0.1
                              /sensorML.xsd"
                              xmlns:sir="http://swsl.uni-muenster.de
                              /sir"
                              xmlns:xsi="http://www.w3.org/2001
                              /XMLSchema-instance">
   <sir:SubscriptionID>001</sir:SubscriptionID>
   <sir:ExpirationDate>2010-09-01T12:00:00</sir:ExpirationDate>
</sir:SubscribeSensorStatusResponse>
```

16 RenewSensorStatusSubscription operation (optional)

16.1 Introduction

The RenewSensorStatusSubscription operation allows SIR clients to renew a subscription for sensor status information before it expires.

16.2 RenewSensorStatusSubscription operation request

16.2.1 RenewSensorStatusSubscription request parameters

A request to perform the RenewSensorStatusDescription operation shall include the data structure specified in Table 45.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 21 in Subclause 9.2.1 of [OGC 06-121r3].

Table 45 — Parameters in RenewSensorStatusSubscription operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is "SIR"	One (mandatory)
request	Operation name	Character String type, not empty Value is "RenewSensorStatusSubscription"	One (mandatory)
version	Standard version for operation	Character String type, not empty Value is "0.3.0"	One (mandatory)
Subscription ID	Identifier of the subscription that shall be extended	Character String type, not empty	One (mandatory)

a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 06-121r3].

16.2.2 RenewSensorStatusSubscription request XML encoding (mandatory)

All SIR servers supporting the RenewSensorStatusSubscription operation shall implement HTTP POST transfer of the RenewSensorStatusSubscription operation request, using XML encoding.

A RenewSensorStatusSubscription operation request is encoded in XML as shown in Figure 28.

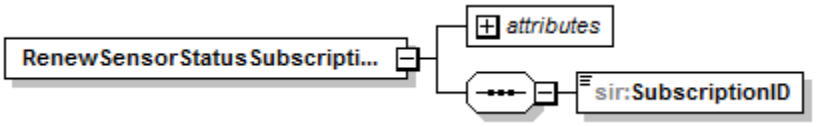

Figure 28: RenewSensorStatusSubscription operation request shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a RenewSensorStatusSubscription operation request encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:element name="RenewSensorStatusSubscriptionRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="SubscriptionID" type="xs:string"/>
         </xs:sequence>
         <xs:attribute name="service" type="xs:string" use="required"
                       fixed="SIR"/>
         <xs:attribute name="version" type="xs:string" use="required"
                       fixed="0.3.0"/>
       </xs:complexType>
   </xs:element>
</xs:schema>
```

EXAMPLE An example RenewSensorStatusSubscription operation request XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<RenewSensorStatusSubscriptionRequest service="SIR" version="0.3.0"
                           xmlns="http://swsl.uni-muenster.de/sir"
                           xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                           instance"
                           xsi:schemaLocation="http://swsl.uni-muenster
                           .de/sir
                           http://giv-genesis.uni-muenster.de/schemas
                           /sir/sirAll.xsd">
   <SubscriptionID>001</SubscriptionID>
</RenewSensorStatusSubscriptionRequest>
```

16.3 RenewSensorStatusSubscription operation response

16.3.1 Normal response parameters

The normal response to a valid RenewSensorStatusSubscription operation request shall be the new expiration date of the subscription. More precisely, a response from the RenewSensorStatusSubscription operation shall include the parts listed in Table 46. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 46 — Parts of RenewSensorStatusSubscription operation response

Names	Definition	Data type and values	Multiplicity and use
SubscriptionID	Identifier of the subscription	Character String type, not empty	One (mandatory)
NewExpirationDate	Date when the subscription will expire.	xs:dateTime	One (mandatory)

16.3.2 Normal response XML encoding

A RenewSensorStatusSubscription operation response is encoded in XML as shown in Figure 29.

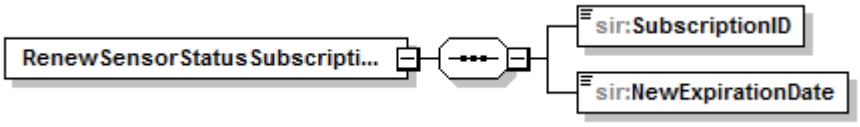

Figure 29: RenewSensorStatusSubscription operation response shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a RenewSensorStatusSubscription operation response, always encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:element name="RenewSensorStatusSubscriptionResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="SubscriptionID" type="xs:string"/>
            <xs:element name="NewExpirationDate" type="xs:dateTime"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

16.3.3 RenewSensorStatusSubscription response example

A RenewSensorStatusSubscription operation response can look like this encoded in XML:

```
<sir:RenewSensorStatusSubscriptionResponse
                    xsi:schemaLocation="http://swsl.uni-muenster.de/sir
                    http://giv-genesis/schemas/sir/sirAll.xsd
                    http://www.opengis.net/sensorML/1.0.1
                    http://schemas.opengis.net/sensorML/1.0.1
                    /sensorML.xsd"
                    xmlns:sir="http://swsl.uni-muenster.de/sir"
```

```
                    xmlns:xsi="http://www.w3.org/2001
                    /XMLSchema-instance">
   <sir:SubscriptionID>001</sir:SubscriptionID>
   <sir:NewExpirationDate>2010-09-01T18:00:00</sir:NewExpirationDate>
</sir:RenewSensorStatusSubscriptionResponse>
```

17 CancelSensorStatusSubscription operation (optional)

17.1 Introduction

The CancelSensorStatusSubscription operation allows SIR clients to cancel a subscription for sensor status information.

17.2 CancelSensorStatusSubscription operation request

17.2.1 CancelSensorStatusSubscription request parameters

A request to perform the CancelSensorStatusSubscription operation shall include the data structure specified in Table 47.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 21 in Subclause 9.2.1 of [OGC 06-121r3].

Table 47 — Parameters in CancelSensorStatusSubscription operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is "SIR"	One (mandatory)
request	Operation name	Character String type, not empty Value is "CancelSensorStatusSubscription"	One (mandatory)
version	Standard version for operation	Character String type, not empty Value is "0.3.0"	One (mandatory)
SubscriptionID	Identifier of the subscription that shall be cancelled	Character String type, not empty	One (mandatory)

a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 06-121r3].

17.2.2 CancelSensorStatusSubscription request XML encoding (mandatory)

All SIR servers supporting the CancelSensorStatusSubscription operation shall implement HTTP POST transfer of the CancelSensorStatusSubscription operation request, using XML encoding.

A CancelSensorStatusSubscription operation request is encoded in XML as shown in Figure 30.

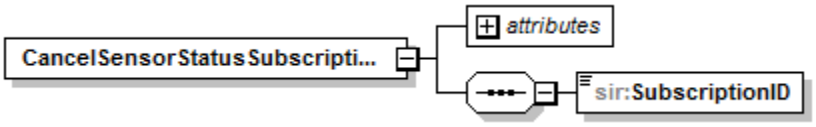

Figure 30: CancelSensorStatusSubscription operation request shown in XMLSpy notation.

The following schema fragmet specifies the contents and structure of a CancelSensorStatusSubscription operation request encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           xmlns:swe="http://www.opengis.net/swe/1.0"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:element name="CancelSensorStatusSubscriptionRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="SubscriptionID" type="xs:string"/>
         </xs:sequence>
         <xs:attribute name="service" type="xs:string" use="required"
                       fixed="SIR"/>
         <xs:attribute name="version" type="xs:string" use="required"
                       fixed="0.3.0"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

EXAMPLE An example CancelSensorStatusSubscription operation request XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<CancelSensorStatusSubscriptionRequest service="SIR" version="0.3.0"
                  xmlns="http://swsl.uni-muenster.de/sir"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://swsl.uni-muenster.de/sir
                  http://giv-genesis.uni-muenster.de/schemas/sir
                  /sirAll.xsd">
   <SubscriptionID>001</SubscriptionID>
</CancelSensorStatusSubscriptionRequest>
```

17.3 CancelSensorStatusSubscription operation response

17.3.1 Normal response parameters

The normal response to a valid CancelSensorStatusSubscription operation request shall be a confirmation that the subscription has successfully been cancelled. More precisely, a response from the CancelSensorStatusSubscription operation shall include the parts listed in Table 48. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 48 — Parts of CancelSensorStatusSubscription operation response

Names	Definition	Data type and values	Multiplicity and use
SubscriptionID	Identifier of the subscription that was cancelled	Character String type, not empty	One (mandatory)

17.3.2 Normal response XML encoding

A CancelSensorStatusSubscription operation response is encoded in XML as shown in Figure 31.

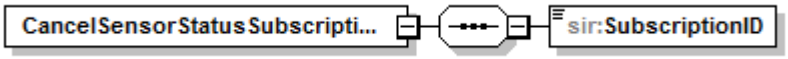

Figure 31: CancelSensorStatusSubscription operation response shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a CancelSensorStatusSubscription operation response, always encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           xmlns:swe="http://www.opengis.net/swe/1.0"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:element name="CancelSensorStatusSubscriptionResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="SubscriptionID" type="xs:string"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

17.3.3 CancelSensorStatusSubscription response example

A CancelSensorStatusSubscription operation response can look like this encoded in XML:

```
<sir:CancelSensorStatusSubscriptionResponse
                    xsi:schemaLocation="http://swsl.uni-muenster.de/sir
                    http://giv-genesis/schemas/sir/sirAll.xsd
                    http://www.opengis.net/sensorML/1.0.1
                    http://schemas.opengis.net/sensorML/1.0.1
                    /sensorML.xsd"
                    xmlns:sir="http://swsl.uni-muenster.de/sir"
                    xmlns:xsi="http://www.w3.org/2001
                    /XMLSchema-instance">
   <sir:SubscriptionID>001</sir:SubscriptionID>
</sir:CancelSensorStatusSubscriptionResponse>
```

18 InsertSensorStatus operation (optional)

18.1 Introduction

The InsertSensorStatus operation allows SIR clients to insert status information about a specific sensor.

18.2 InsertSensorStatus operation request

18.2.1 InsertSensorStatus request parameters

A request to perform the InsertSensorStatus operation shall include the data structure specified in Table 49.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 21 in Subclause 9.2.1 of [OGC 06-121r3].

Table 49 — Parameters in InsertSensorStatus operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “SIR”	One (mandatory)
request	Operation name	Character String type, not empty Value is “InsertSensorStatus”	One (mandatory)
version	Standard version for operation	Character String type, not empty Value is “0.3.0”	One (mandatory)
StatusDescription	Status information that shall be inserted	SIR StatusDescription type	One or more (mandatory)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 06-121r3].			

18.2.2 InsertSensorStatus request XML encoding (mandatory)

All SIR servers supporting the InsertSensorStatus operation shall implement HTTP POST transfer of the InsertSensorStatus operation request, using XML encoding.

An InsertSensorStatus operation request is encoded in XML as shown in Figure 32.

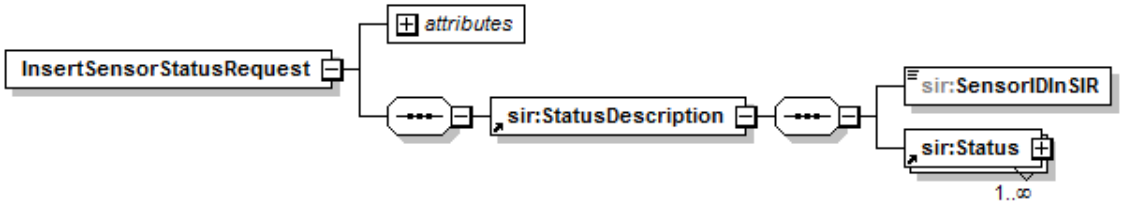

Figure 32: InsertSensorStatus operation request shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of an InsertSensorStatus operation request encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:gml="http://www.opengis.net/gml"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:include schemaLocation="sirCommon.xsd"/>
   <xs:element name="InsertSensorStatusRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="sir:StatusDescription"/>
         </xs:sequence>
         <xs:attribute name="service" type="xs:string" use="required"
                       fixed="SIR"/>
         <xs:attribute name="version" type="xs:string" use="required"
                       fixed="0.3.0"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

EXAMPLE An example InsertSensorStatus operation request XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<InsertSensorStatusRequest service="SIR" version="0.3.0"
                           xmlns="http://swsl.uni-muenster.de/sir"
                           xmlns:gml="http://www.opengis.net/gml/3.2"
                           xmlns:swe="http://www.opengis.net/swe/1.0"
                           xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                           instance"
                           xsi:schemaLocation="http://swsl.uni-
                           muenster.de/sir
                           http://giv-genesis.uni-muenster.de/schemas
                           /sir/sirAll.xsd">
   <StatusDescription>
      <SensorIDInSIR>1</SensorIDInSIR>
      <Status>
         <PropertyName>batterystate</PropertyName>
         <PropertyValue>55</PropertyValue>
         <Uom code="%" />
         <Timestamp gml:id="timestamp_01">
            <gml:timePosition>
               2010-04-16T15:43:06+0200
```

```
            </gml:timePosition>
         </Timestamp>
      </Status>
   </StatusDescription>
</InsertSensorStatusRequest>
```

18.3 InsertSensorStatus operation response

18.3.1 Normal response parameters

The normal response to a valid InsertSensorStatus operation request shall be a confirmation that the status information has been successfully inserted into the SIR database. More precisely, a response from the InsertSensorStatus operation shall include the parts listed in Table 50. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 50 — Parts of InsertSensorStatus operation response

Names	Definition	Data type and values	Multiplicity and use
SensorIDInSIR	String identifying the sensor within the SIR instance for which the status information was inserted	Character String type, not empty	One (mandatory)

18.3.2 Normal response XML encoding

An InsertSensorStatus operation response is encoded in XML as shown in Figure 33.

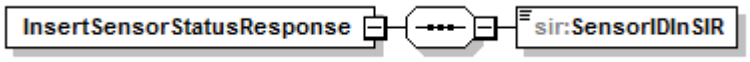

Figure 33: InsertSensorStatus operation response shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of an InsertSensorStatus operation response, always encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:gml="http://www.opengis.net/gml"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:include schemaLocation="sirCommon.xsd"/>
   <xs:element name="InsertSensorStatusResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="SensorIDInSIR" type="xs:string"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

18.3.3 InsertSensorStatus response example

An InsertSensorStatus operation response can look like this encoded in XML:

```
<sir:InsertSensorStatusResponse
                    xsi:schemaLocation="http://swsl.uni-muenster.de/sir
                    http://giv-genesis/schemas/sir/sirAll.xsd
                    http://www.opengis.net/sensorML/1.0.1
                    http://schemas.opengis.net/sensorML/1.0.1
                    /sensorML.xsd"
                    xmlns:sir="http://swsl.uni-muenster.de/sir"
                    xmlns:xsi="http://www.w3.org/2001
                    /XMLSchema-instance">
   <sir:SensorIDInSIR>1</sir:SensorIDInSIR>
</sir:InsertSensorStatusResponse>
```

19 ConnectToCatalog operation (mandatory)

19.1 Introduction

The ConnectToCatalog operation allows SIR clients initiate the transfer of sensor metadata from a SIR instance into an OGC Catalog. This way, the information about sensors contained in the SIR partly becomes discoverable through the very common OGC Catalog.

Catalogs receiving the sensor metadata shall be able to handle metadata according to the sensor metadata model described in [09-163r2].

The push process of sensor metadata can be executed either once or within a continuous automatic repetition interval.

19.2 ConnectToCatalog operation request

19.2.1 ConnectToCatalog request parameters

A request to perform the ConnectToCatalog operation shall include the data structure specified in Table 51.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 21 in Subclause 9.2.1 of [OGC 06-121r3].

Table 51 — Parameters in ConnectToCatalog operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “SIR”	One (mandatory)
request	Operation name	Character String type, not empty Value is “ConnectToCatalog”	One (mandatory)
version	Standard version for operation	Character String type, not empty Value is “0.3.0”	One (mandatory)
CatalogURL	URL of the catalog instance into which the sensor metadata shall be inserted	Character string representing a URI	One (mandatory)
PushIntervalSeconds	Time span (in seconds) describing the interval how often new sensor metadata records are pushed into the catalog	xs:int, 0 or higher	One (mandatory)

a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 06-121r3].

19.2.2 ConnectToCatalog request XML encoding (mandatory)

All SIR servers shall implement HTTP POST transfer of the ConnectToCatalog operation request, using XML encoding.

A ConnectToCatalog operation request is encoded in XML as shown in Figure 34.

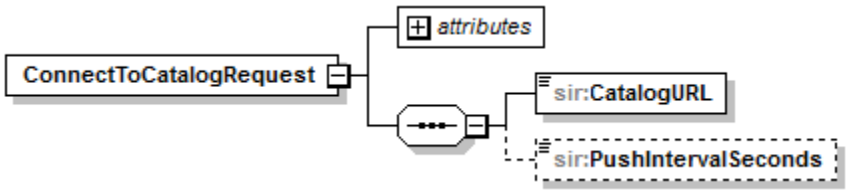

Figure 34: ConnectToCatalog operation request shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a ConnectToCatalog operation request encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:element name="ConnectToCatalogRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="CatalogURL" type="xs:anyURI"/>
            <xs:element name="PushIntervalSeconds" type="xs:int"
                        minOccurs="0"/>
         </xs:sequence>
      <xs:attribute name="service" type="xs:string" use="required"
                    fixed="SIR"/>
      <xs:attribute name="version" type="xs:string" use="required"
                    fixed="0.3.0"/>
       </xs:complexType>
   </xs:element>
</xs:schema>
```

EXAMPLE An example ConnectToCatalog operation request XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<ConnectToCatalogRequest service="SIR" version="0.3.0"
                         xmlns="http://swsl.uni-muenster.de/sir"
                         xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                         instance"
                         xsi:schemaLocation="http://swsl.uni-muenster
                         .de/sir
                         http://giv-genesis.uni-muenster.de/schemas/sir
                         /sirAll.xsd">
   <CatalogURL>
      http://my.catalogserver:8080/ergorr/webservice
   </CatalogURL>
   <PushIntervalSeconds>3600</PushIntervalSeconds>
```

```
</ConnectToCatalogRequest>
```

19.3 ConnectToCatalog operation response

19.3.1 Normal response parameters

The normal response to a valid ConnectToCatalog operation request shall be a confirmation of the successful connection to the catalog instance. More precisely, a response from the ConnectToCatalog operation shall include the parts listed in Table 52. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 52 — Parts of ConnectToCatalog operation response

Names	Definition	Data type and values	Multiplicity and use
CatalogURL	URL of the catalog instance into which the sensor metadata is being inserted	Character string representing a URI	One (mandatory)

19.3.2 Normal response XML encoding

A ConnectToCatalog operation response is encoded in XML as shown in Figure 35.

Figure 35: ConnectToCatalog operation response shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a ConnectToCatalog operation response, always encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:element name="ConnectToCatalogResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="CatalogURL" type="xs:anyURI"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

19.3.3 ConnectToCatalog response example

A ConnectToCatalog operation response can look like this encoded in XML:

```
<sir:ConnectToCatalogResponse
```

```
                    xsi:schemaLocation="http://swsl.uni-muenster.de/sir
                    http://giv-genesis/schemas/sir/sirAll.xsd
                    http://www.opengis.net/sensorML/1.0.1
                    http://schemas.opengis.net/sensorML/1.0.1
                    /sensorML.xsd"
                    xmlns:sir="http://swsl.uni-muenster.de/sir"
                    xmlns:xsi="http://www.w3.org/2001
                    /XMLSchema-instance">
   <sir:CatalogURL>
      http://giv-genesis.uni-muenster.de:8080/ergorr/webservice
   </sir:CatalogURL>
</sir:ConnectToCatalogResponse>
```

20 DisconnectFromCatalog operation (mandatory)

20.1 Introduction

The DisconnectFromCatalog operation allows SIR clients to stop continuous push processes of sensor metadata into OGC Catalogs.

20.2 DisconnectFromCatalog operation request

20.2.1 DisconnectFromCatalog request parameters

A request to perform the DisconnectFromCatalog operation shall include the data structure specified in Table 53.

NOTE 1 To reduce the need for readers to refer to other documents, the first three parameters listed below are largely copied from Table 21 in Subclause 9.2.1 of [OGC 06-121r3].

Table 53 — Parameters in DisconnectFromCatalog operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “SIR”	One (mandatory)
request	Operation name	Character String type, not empty Value is “DisconnectFromCatalog”	One (mandatory)
version	Standard version for operation	Character String type, not empty Value is “0.3.0”	One (mandatory)
CatalogURL	URL of the catalog instance into which the sensor metadata is being inserted	Character string representing a URI	One (mandatory)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 06-121r3].			

20.2.2 DisconnectFromCatalog request XML encoding (mandatory)

All SIR servers shall implement HTTP POST transfer of the DisconnectFromCatalog operation request, using XML encoding.

A DisconnectFromCatalog operation request is encoded in XML as shown in Figure 36.

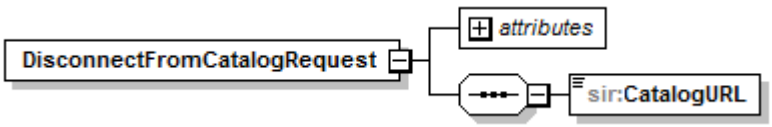

Figure 36: DisconnectFromCatalog operation request shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a DisconnectFromCatalog operation request encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:element name="DisconnectFromCatalogRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="CatalogURL" type="xs:anyURI"/>
         </xs:sequence>
         <xs:attribute name="service" type="xs:string" use="required"
                       fixed="SIR"/>
         <xs:attribute name="version" type="xs:string" use="required"
                       fixed="0.3.0"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

EXAMPLE An example DisconnectFromCatalog operation request XML encoded for HTTP POST is:

```
<?xml version="1.0" encoding="UTF-8"?>
<DisconnectFromCatalogRequest service="SIR" version="0.3.0"
                              xmlns="http://swsl.uni-muenster.de/sir"
                              xmlns:xsi="http://www.w3.org/2001
                              /XMLSchema-instance"
                              xsi:schemaLocation="http://swsl.uni-
                              muenster.de/sir
                              http://giv-genesis.uni-muenster.de
                              /schemas/sir/sirAll.xsd">
   <CatalogURL>http://localhost:8080/ergorr/webservice</CatalogURL>
</DisconnectFromCatalogRequest>
```

20.3 DisconnectFromCatalog operation response

20.3.1 Normal response parameters

The normal response to a valid DisconnectFromCatalog operation request shall be a confirmation that the push process of sensor metadata into the catalog has been stopped. More precisely, a response from the DisconnectFromCatalog operation shall include the parts listed in Table 54. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 54 — Parts of DisconnectFromCatalog operation response

Names	Definition	Data type and values	Multiplicity and use
CatalogURL	URL of the catalog instance to which the transfer of sensor metadata was stopped.	Character string representing a URI	One (mandatory)

20.3.2 Normal response XML encoding

A DisconnectFromCatalog operation response is encoded in XML as shown in Figure 37.

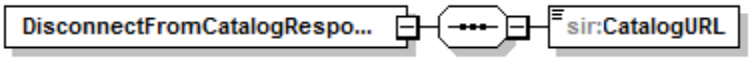

Figure 37: DisconnectFromCatalog operation response shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a DisconnectFromCatalog operation response, always encoded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:sir="http://swsl.uni-muenster.de/sir"
           targetNamespace="http://swsl.uni-muenster.de/sir"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:element name="DisconnectFromCatalogResponse">
       <xs:complexType>
         <xs:sequence>
            <xs:element name="CatalogURL" type="xs:anyURI"/>
         </xs:sequence>
       </xs:complexType>
   </xs:element>
</xs:schema>
```

20.3.3 DisconnectFromCatalog response example

A DisconnectFromCatalog operation response can look like this encoded in XML:

```
<sir:DisconnectFromCatalogResponse
                    xsi:schemaLocation="http://swsl.uni-muenster.de/sir
                    http://giv-genesis/schemas/sir/sirAll.xsd
                    http://www.opengis.net/sensorML/1.0.1
                    http://schemas.opengis.net/sensorML/1.0.1
                    /sensorML.xsd"
                    xmlns:sir="http://swsl.uni-muenster.de/sir"
                    xmlns:xsi="http://www.w3.org/2001
                    /XMLSchema-instance">
   <sir:CatalogURL>
      http://giv-genesis.uni-muenster.de:8080/ergorr/webservice
   </sir:CatalogURL >
</sir:DisconnectFromCatalogResponse>
```

21 Implementation

Within the OSIRIS project a first prototypical implementation of the SIR was performed by the University of Muenster. A significantly enhanced implementation which is made available through the open source initiative 52° North[1] was created within the GENESIS project. The SIR implementation includes also a simple web-based client for comfortable request generation and testing.

One main feature of SIR is the catalog connection to OGC Catalog instances. It is based on a XSLT[2] transformation of the given SensorML sensor descriptions into the ebRIM data structure as defined in OGC 09-163r2. The following fragment shows the wrapping of a SensorML system instance, which is transformed into a so called extrinsic object, which is wrapped into a registry package respectively list:

```
<xsl:element name="rim:RegistryPackage>
   ...
   <xsl:element name="rim:RegistryObjectList">
      <xsl:apply-templates select="sml:System"
                           mode="extrinsic-object"/>
      <xsl:apply-templates select="sml:System"
                          mode="classification-association"/>
      <xsl:apply-templates select="sml:System/sml:contact"/>
      <xsl:apply-templates select="sml:System/sml:components"/>
   </xsl:element>
</xsl:element>
```

The System is processed several times to create both the transformation of itself, the according elements for classification and associations to for example related services, and the elements for nested components. The following fragment is an excerpt of such a transformation:

```
<wrs:ExtrinsicObject   ...>
   <rim:Slot name="urn:ogc:def:slot:OGC-CSW-ebRIM-Sensor::Keywords"
             slotType="urn:oasis:names:tc:ebxml-
                       regrep:DataType:String">
      <rim:ValueList>
         <rim:Value>weather station</rim:Value>
         <rim:Value>precipitation</rim:Value>
         ...
         <rim:Slot name="urn:ogc:def:slot:OGC-CSW-ebRIM-
                         Sensor::Outputs"
                   slotType="urn:oasis:names:tc:ebxml-
                             regrep:DataType:String">
      </rim:ValueList>
      <rim:Value>
         urn:ogc:def:property:OGC::Precipitation1Hour
      </rim:Value>
      ...
```

[1] https://52north.org/

[2] http://www.w3.org/TR/xslt

```
   <rim:Slot name="urn:ogc:def:slot:OGC-CSW-ebRIM-Sensor::ShortName"
             slotType="urn:oasis:names:tc:ebxml-
                       regrep:DataType:String">
      <rim:ValueList>
         <rim:Value>IFGI HWS 1</rim:Value>
      </rim:ValueList>
   </rim:Slot>
...
```

21.1 Examples

21.1.1 Using the SIR for Searching Sensors

The SIR allows providing a reference to a Sensor Observable Registry within its search criteria. This can be a very powerful feature as this example shows using both the SIR and SOR implementations by 52° North. Figure 38 shows the web client interface with the request and response of a SearchSensor operation. The client's input fields to build the request are not shown here for brevity. The response contains no results, because the user searched for the specific phenomenon wind chill identified by the URN *urn:ogc:def:property:OGC::WindChill*.

Build request

```
1 <SearchSensorRequest service="SIR" version="0.3.0" xsi:schemaLocation="http://swsl.uni-
  muenster.de/sir http://giv-genesis/schemas/sir/sirAll.xsd http://www.opengis.net/sensorML
  /1.0.1 http://schemas.opengis.net/sensorML/1.0.1/sensorML.xsd" xmlns="http://swsl.uni-
  muenster.de/sir" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
2   <SearchCriteria>
3     <Phenomenon>
4       <PhenomenonName>urn:ogc:def:property:OGC::WindChill</PhenomenonName>
5     </Phenomenon>
6   </SearchCriteria>
7   <SimpleResponse>true</SimpleResponse>
8 </SearchSensorRequest>
```

Send request

```
1 <sir:SearchSensorResponse xsi:schemaLocation="http://swsl.uni-muenster.de/sir http://giv-
  genesis/schemas/sir/sirAll.xsd http://www.opengis.net/sensorML/1.0.1
  http://schemas.opengis.net/sensorML/1.0.1/sensorML.xsd" xmlns:sir="http://swsl.uni-
  muenster.de/sir" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
2
3
4
5
6
7
8
```

Figure 38: Screenshot of unsuccessful search for a certain phenomenon

The SOR can be used to request equivalent or related phenomena. This feature is utilized in the request shown in the screenshot in Figure 39. The phenomenon information is now complemented by parameters to query a SOR using its GetMatchingDefinitions request (see clause 9 in OGC 09-112r1 for details). These parameters are: a URL of a running

SOR instance, the type of matching and the search depth to be applied in the matching algorithm. This time, the request returns a number of search results of which the first can be seen in the text field in the lower half of the screenshot. The consultation of the SOR resulted in the phenomenon *urn:ogc:def:property:OGC::Temperature* as a valid super type for *urn:ogc:def:property:OGC::WindChill*, and there are several sensors available who provide those kind of measurements.

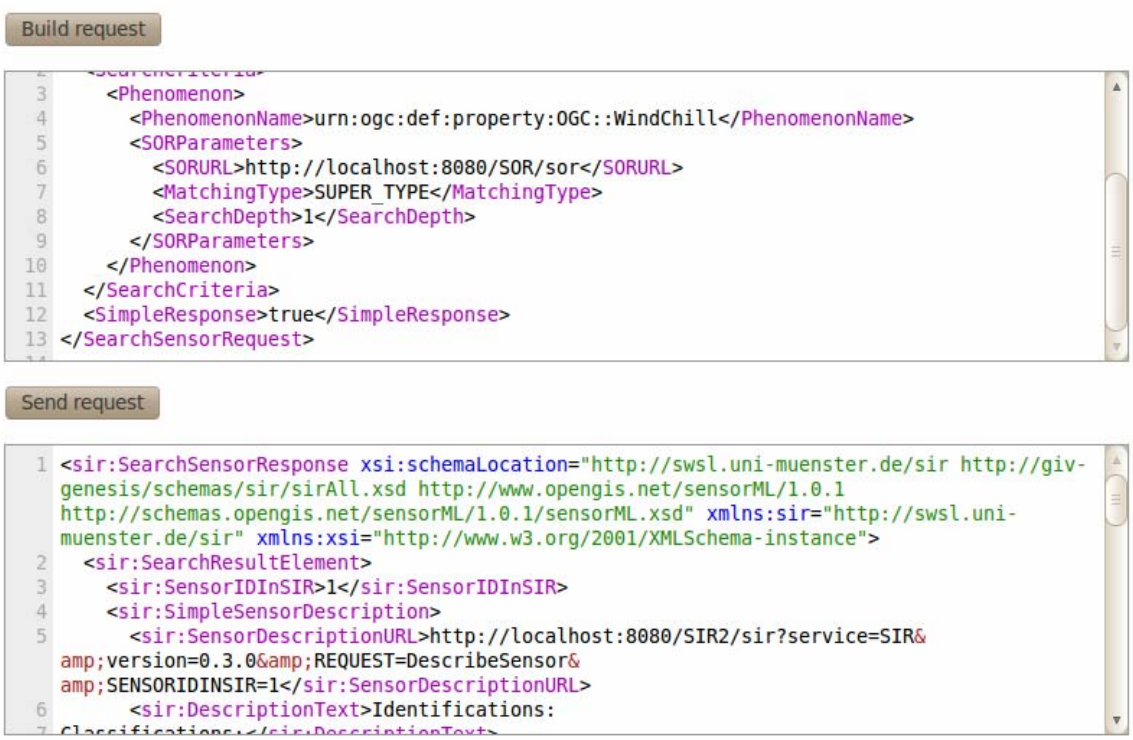

Figure 39: Screenshot of successful search for a phenomenon with semantic matching powered by a SOR

21.1.2 Harvesting a Sensor Observation Service

Figure 40 shows the request and response of a harvesting operation for an OGC Sensor Observation Service. The request was created with the information provided in the top of the screen, namely the URL and type of the service. The button “Build request” created the according XML request for inspection. The request was send using the button “Send request” and the response is shown in the lower text field. It contains the information that the service has been harvested before, because the 2 found sensors were both updated.

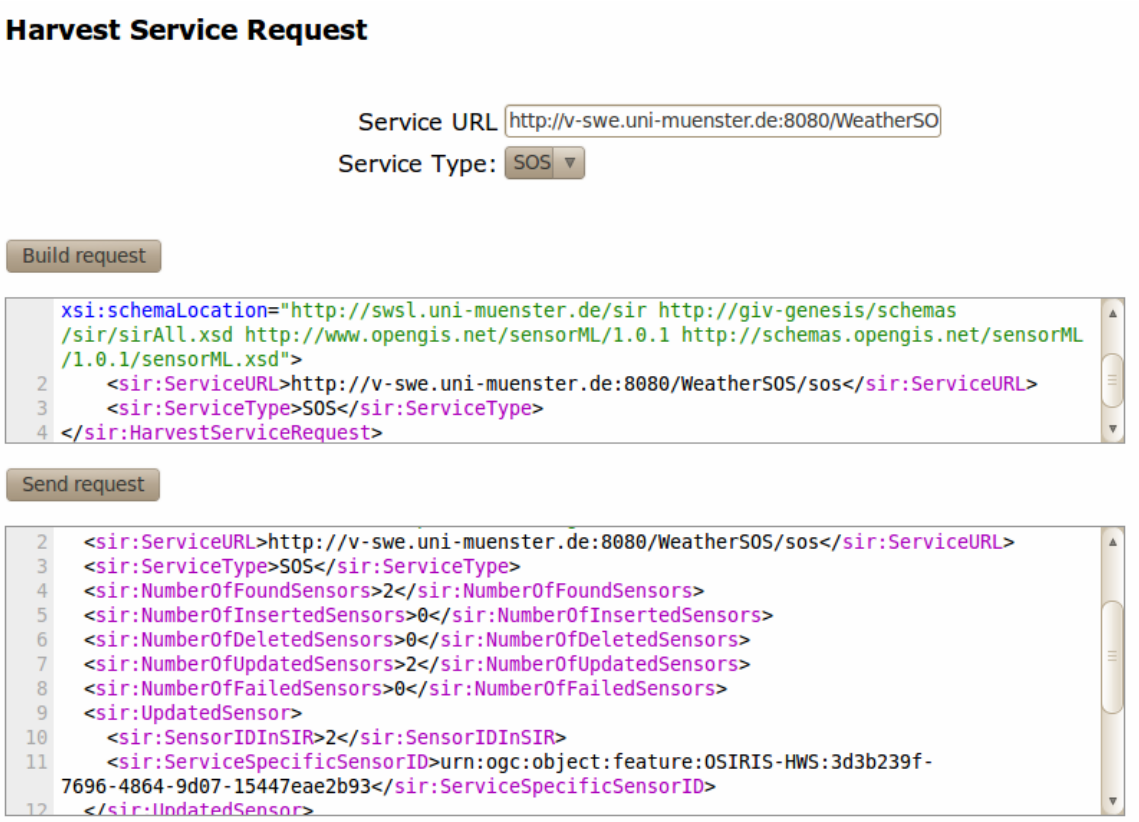

Figure 40: Screenshot of the web client with a harvesting operation's request and response

21.1.3 Requesting Status Information

This example shows the request of a particular status property *batterystate* by using a property filter for all sensors with a service URL *http://v-swe.uni-muenster.de:8080/WeatherSOS/sos* by using a sir criteria element. Figure 41 shows a screenshot of the web based client for this operation, the input fields for creating the request are not visible. The request with the aforementioned settings is visible in the upper text field after clicking “Build request”. The response in the lower text field appears after sending the request (using the “Send request” button) and shows the according information for sensors with ids 1 and 2 that match the request.

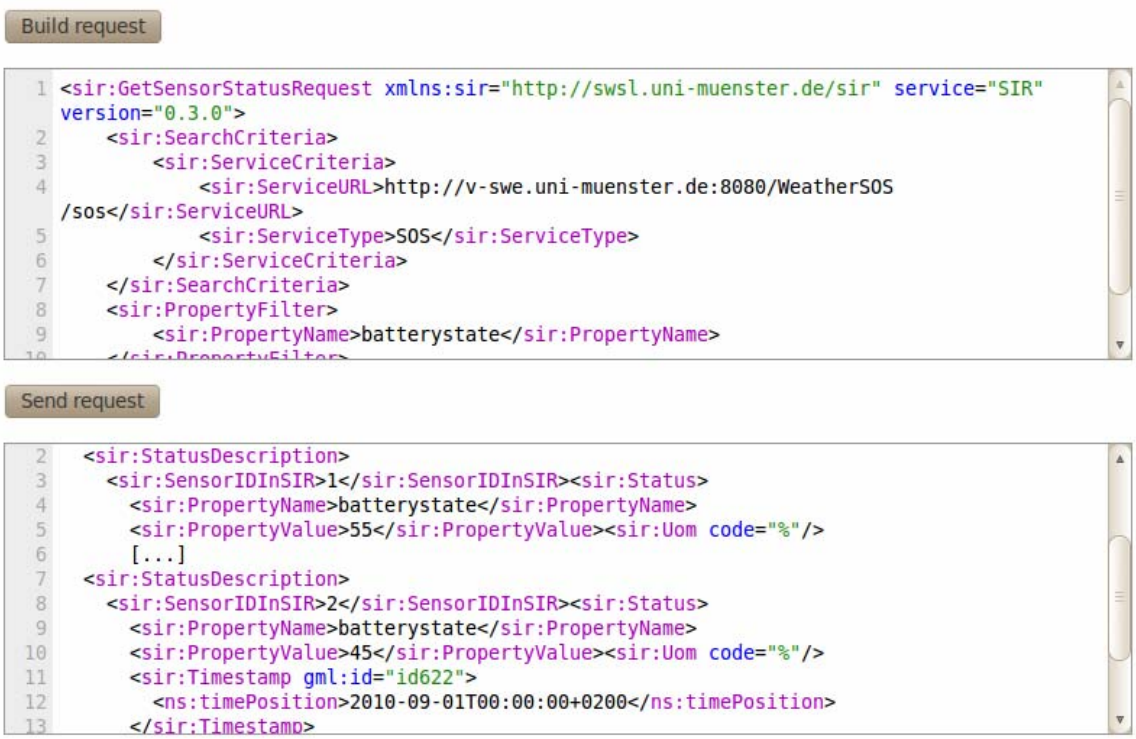

Figure 41: Screenshot of a GetSensorStatus operation

21.1.4 Connecting SIR with an OGC Catalog

This example shows the functionality to establish a permanent link to an OGC Catalog. Figure 42 shows a screenshot of a successful ConnectToCatalog operation. In the top third of the screen the information about the catalog and the desired interval for repetition of synchronization from the SIR into the catalog are entered. The request in the text field in the middle of the screen was created with the “Build request” button. After inspecting and sending the request with the “Send request” button, the response is shown in the text window at the bottom of the screen. It contains a ConnectToCatalog response with the catalog’s URL to show the successful scheduling of the connection.

Figure 42: Screenshot of web client with ConnectToCatalog operation request and non-error response

The actual linking process consists of several steps. First, the capabilities of the given catalog are requested to ensure version and type compatibility. Second, the SIR checks if the required ebRIM classification schemes and slot types are present. If not, the SIR tries to insert them into the Catalog. These insertions are checked for completeness and only if they are successful the SIR starts the push of its data into the catalog.

The SIR allows configuring services that do not need to be checked, because the user is aware about their capabilities, to avoid this possibly resource intensive process.

The insertion of the data itself comprises the transformation of the stored SensorML sensor descriptions into ebRIM data model and wrapping these in insert transaction documents (possibly several sensors in one transaction). These transactions are then sent to the catalog and responses are logged for inspection. After a successful transfer of data a user can explore, query and retrieve the information via the OGC Catalog service interface.

21.1.5 Transform a SensorML document to ebRIM

The transformation of sensor descriptions is based on XSLT. It closely follows the mapping described in OGC 09-163r2. Even though this feature is only used within the service, the test client offers a basic interface (shown in Figure 43) to manually perform such a transformation. The upper part of the picture shows the input text field for the SensorML description (discovery profile conform), the lower part shows a part of the transformed document. Here the output transformation of the generated outputs, a

SensorML identifier for a short name, and a GML description element is visible. The output document is ready for insertion into an OGC Catalog instance.

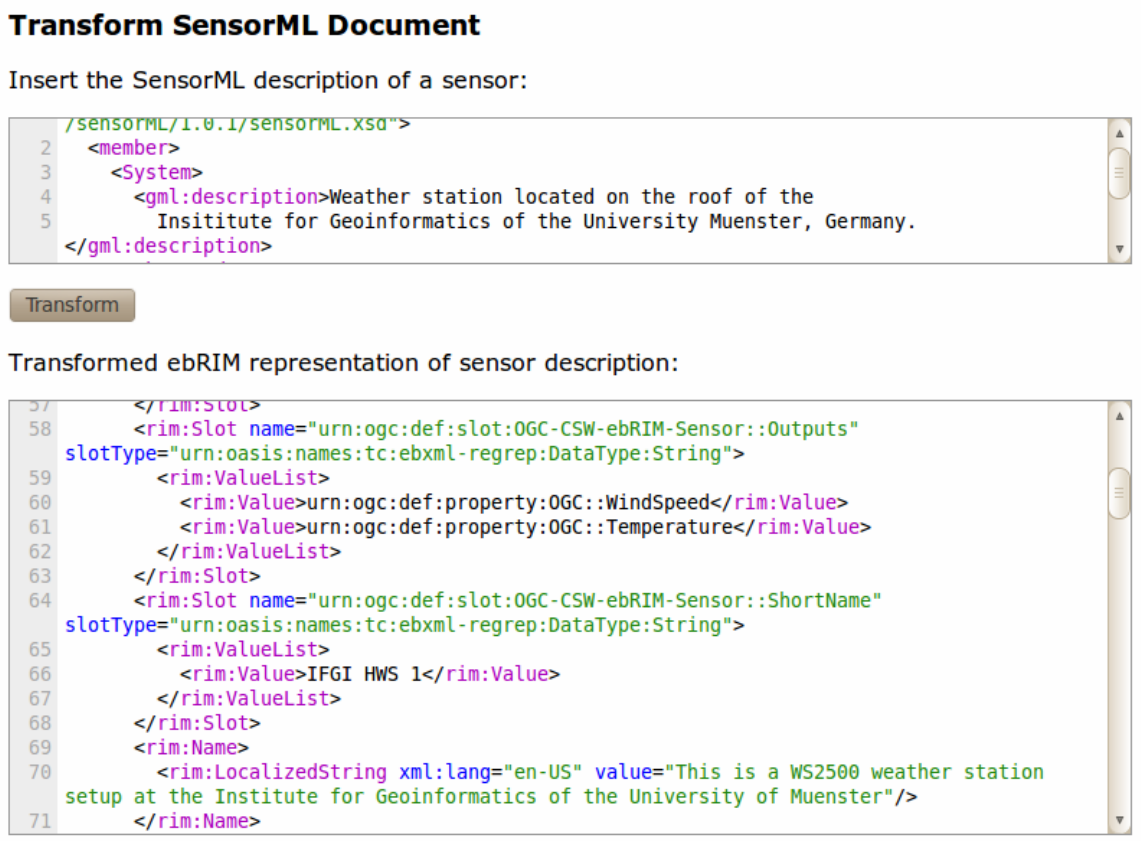

Figure 43: Screenshot showing the transformation (output below input text field) of a SensorML to an ebRIM document

22 Summary and Outlook

In summary the SIR is intended as a broker mediating between the highly dynamic and SensorML based world of sensor networks and classic spatial data infrastructure concepts, especially OGC Catalog. This link is achieved through a set of functionality comprising sensor metadata collection, sensor discovery, sensor status management and sensor metadata transformation as well as sensor metadata transfer. The coupling between SIR and other services, which are providing and cataloguing metadata, making actual observation data available, or offering semantic reasoning, is both flexible and powerful.

For the future it is intended to align the SIR specification more to other existing SWE service interfaces. Especially the discovery functionality and the access to sensor status information might be realized using operations of the Sensor Observation Service. The subscription to sensor status information might also be covered by operations described in the Sensor Event Service specification.

Additionally it will be necessary to keep the SIR specification aligned to progress made in the definition of the SensorML profile for discovery as well as in the specification of the SensorML-ebRIM mapping.

Bibliography

[1] OpenGIS® Sensor Model Language (SensorML) Implementation Standard, OGC document 07-000

[2] OWS-6 SensorML Profile for Discovery Engineering Report, OGC document OGC 09-033

[3] Sensor Observable Registry Discussion Paper, OGC document OGC 09-112

[4] SensorML Extension Package for ebRIM Application Profile, OGC document OGC 09-163r2

[5] Jirka, S.; Bröring, A.; Stasch, C.. 2009. “Discovery Mechanisms for the Sensor Web”. Sensors 9, no. 4: 2661-2681. Online: http://www.mdpi.com/1424-8220/9/4/2661

[6] OSIRIS Consortium. 2008. “Deliverable 6001 Revision A - OSIRIS Architecture Specification and Justification”.

[7] GENESIS Consortium. 2010. “D5200.1 Report On Sensor Network Architecture Assessment Activity”

[8] GENESIS Consortium. 2010. “D5200.2 SWE Demonstrator Explicative Notice”

