Open Geospatial Consortium Inc.

Date: 2009-10-13

Reference number of this document: OGC 09-112

Version: 0.3.0

Category: Discussion Paper

Editor(s): Simon Jirka, Arne Bröring

OGC® Sensor Observable 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:	OGC Discussion Paper
Document stage:	Draft
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	IfGI, University of Münster
Arne Bröring	52° North

v. Revision history

Date	Release	Editor	Primary clauses modified	Description
2009-09-05	0.3.0	Simon Jirka	All	First version of the Discussion Paper

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

Improvements in this document are desirable to extend the functionality of the SOR. This includes especially the following aspects:

The SOR addresses the access to ontologies and the execution of reasoning mechanisms. For both of these aspects there might be a more general interest across multiple domains. Thus, it should be investigated how the access to ontologies and the execution of reasoning processes could be encapsulated by separate service interfaces. The pragmatic approach of the SOR can serve as a basis for the approach presented by Janowicz et al. 2009 who propose semantic enablement architecture for SDIs by introducing two new interfaces, the Web Ontology Service (WOS) and highly flexible Web Reasoning Service (WRS). The WOS should be designed as a profile of the CS-W specification (OGC 07-006r1) while the WRS encapsulates semantic web reasoners and should be developed as a profile of the WPS (OGC 05-007r7).

In its current state, the SOR serves only simple reasoning functionality. It allows investigating the hierarchical relationships between phenomena by basic subsumption reasoning. This reasoning capability enables to determine generalization, specialization and equivalency between phenomena. However, a much higher degree of freedom and flexibility regarding reasoning abilities is desirable. In a future approach, reasoning should not be restricted to subsumption, but should include non-standard inference such as finding the most specific concept, the least common subsumer as well as similarity reasoning.

Additionally, the SOR's focus on handling phenomena and their definitions can be broadened in future. It shall be assessed how the SOR concept could be transferred to further application domains.

Foreword

This Discussion paper introduces the Sensor Observable Registry (SOR), a web service interface for managing the definitions of phenomena measured by sensors as well as exploring semantic relationships between these phenomena.

With this Discussion Paper the authors aim at presenting a pragmatic solution that may serve as the basis for further discussion and could serve as input to the creation of a more generic solution for handling semantics within the context of sensor networks or even beyond this domain.

This work was supported by the EC projects OSIRIS (http://www.osiris-fp6.eu/, contract number 033475) and GENESIS (http://genesis-fp7.eu/, contract number 223996) co-funded by the Information Society and Media DG of the European Commission.

Introduction

This Discussion Paper introduces the concept of the Sensor Observable Registry (SOR) which was created within the EC project OSIRIS and further evolved in the EC project GENESIS.

In sensor networks there is a need for handling semantics. This is especially true for handling phenomenon definitions. It is necessary to be able to specify the phenomena that are observed by a sensor so that the discovery of sensors (searching for sensors that observe certain phenomena) and the interpretation of sensor data (determining which phenomenon was observed) become possible.

Within the OSIRIS project an important need was to create a solution for managing the phenomena observed by sensors as well as the definitions and semantics of these phenomena. Basically two requirements lead to the development of the SOR:

- The need for accessing descriptions of phenomena identified by certain URNs
- The enhancement of the sensor discovery process by exploring and investigating the semantics of observed phenomena

Based on the two above mentioned requirements, the following use cases are supported by the SOR:

- If a user wants to search for sensors measuring a certain phenomenon, how can he determine in which way he needs to specify the phenomenon he is interested in. Usually the phenomena measured by sensors are identified by certain URNs. However, as these URNs may vary and as they might even be unknown to the user, he needs some kind of dictionary to look up the available phenomena and their corresponding URNs.
- If users receive metadata about sensor observations, the measured phenomena are identified through URNs. For being able to better understand the meaning of these URNs some kind of dictionary is needed that is capable of resolving the URNs and returning according definitions and explanations.
- Across different domains or even within a single domain the same phenomenon might be identified by different URNs. Thus, for being able to find all phenomena which might be of interest to a user, there must be a solution that is capable of determining all of those URNs which identify the same phenomenon. The foundation for achieving this aim is the exploitation of the semantics that lie behind the URNs identifying phenomenon definitions.

- When searching for a sensor, users sometimes are interested in finding sensors that observe a phenomenon which belongs to a broader thematic field or category of phenomena. For example in case of an accident in a chemical plant a cloud of air pollutants might be released. Directly after the accident users might not know which pollutants have been released. Thus, they need various kinds of air pollutant sensors to determine which chemical substances are contained within the pollutant cloud. One approach might be that the user relies on his knowledge to search for sensors measuring those kinds of air pollutants he knows (e.g. searching separately for sensors measuring CO, NO_2, SO_2, H2CO, etc.). However this approach for searching sensors would be very cumbersome and possesses the risk of forgetting to search for one or more relevant sensor types. Instead a much more efficient approach would be to search for sensors measuring the phenomenon “air pollution”. In this case one search request would return directly all sensor types the user needs. However, for making such a type of discovery possible, it is necessary to exploit the semantics of the phenomena measured by sensors. A sensor discovery solution has to be able to determine that CO, NO_2, SO_2, H_2CO, etc. all are some kind of air pollutant. This functionality is provided through the SOR.

OGC Sensor Observable Registry Discussion Paper

1 Scope

This OGC Discussion Paper describes the Sensor Observable Registry (SOR), a web service interface for providing definitions of phenomena that are observed by sensors, for resolving URNs identifying these definitions and for exploring semantic relationships between phenomena.

This document is intended to stimulate the discussion on ways for handling the semantics of sensor observables. Furthermore the basic concepts of the SOR may be advanced in the future so that not only sensor observables but any kind of definition can be handled.

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 Specification*

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

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

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

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

3 Terms and definitions

For the purposes of this specification, the definitions specified in Clause 4 of the OWS Common Implementation Specification [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.

4 Conventios

4.1 Abbreviated terms

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

Some more frequently used abbreviated terms:

SensorML	Sensor Model Language
SOR	Sensor Observable Registry
SWE	Sensor Web Enablement
URN	Uniform Resource Name

5 Sensor Observable Registry Overview

The specified Sensor Observable Registry is capable of providing definitions of phenomena that are observed by sensors, of resolving URNs identifying these definitions and of exploring semantic relationships between phenomena.

The SOR interface (currently) specifies four operations that can be requested by a client and performed by a SOR server. Those operations are:

a) GetCapabilities (required implementation by servers) – This operation allows a client to request and receive back service metadata (or 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.

b) GetDefinitionURNs (required implementation by servers) – This operation allows a client to retrieve a list of URNs identifying definitions that are contained in a specific SOR instance. Optionally, a client is able to specify a text string to search for URNs which contain it. Furthermore a paging mechanism is supported for retrieving subsets of a longer list of URNs. This functionality is especially intended for supporting the development of client user interfaces as it allows to create lists from which users can select the URNs of those phenomenon definitions they are interested in.

c) GetDefinition (required implementation by servers) – This operation allows a client to retrieve the definition of an observable identified by a given URN. This definition is returned as a GML dictionary entry.

d) GetMatchingDefinitions (required implementation by servers) – This operation allows a client to retrieve the URNs of definitions of observables which are related to another given phenomenon in a certain way. Currently the SOR supports the relations "generalization", "specialization" and "equivalency".

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

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

6 GetCapabilities operation (mandatory)

6.1 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.

6.2 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 “SOR”. 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 1 specifies the implementation of those parameters by SOR clients and servers.

Table 1 — Implementation of parameters in GetCapabilities operation request

Names	Multiplicity	Client implementation	Server implementation
service	One (mandatory)	Each parameter shall be implemented by all clients, using specified value	Each parameter shall be implemented by all servers, checking that each parameter is received with specified value
request	One (mandatory)		
AcceptVersions	Zero or one (optional)	Should be implemented by all software clients, using specified values	Shall be implemented by all servers, checking if parameter is received with specified value(s)
Sections	Zero or one (optional) [b]	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	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
updateSequence	Zero or one (optional) [b]		
AcceptFormats	Zero or one (optional) [b]		

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.

All SOR 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 SOR capabilities document, a client could issue the following KVP encoded GetCapabilities operation request with near-minimum contents:

```
http://v-swe.uni-muenster.de:8080/SOR/SOR?service=SOR&request=getCapabilities
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<sor:GetCapabilities service="SOR" xmlns:sor="http://swsl.uni-
muenster.de/sor" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=" http://swsl.uni-muenster.de/sor
..\sor\sorGetCapabilities.xsd">
</sor:GetCapabilities>
```

6.3 GetCapabilities operation response

The service metadata document shall contain the sections that are defined in the OWS Common specification. For the SOR these sections are not modified. The Contents subsection is specific for the SOR. 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 2 — Additional Section name values and their meanings

Section name	Contents
Contents	Metadata about the data served by this server. For the SOR, this section shall contain data about the number of entries it contains, the domain as well as application area it covers and references to the used ontology repositories.

The Contents section of a service metadata document contains metadata about the data served by this server. For the SOR, this Contents section shall contain data about the the number of entries it contains, the domain as well as application area it covers and references to the used ontology repositories. The Contents section shall include the parameters specified in Table 3.

Table 3 — Parameters included in Contents section

Names	Definition	Data type and values	Multiplicity and use
numberOfEntries	Number of entries contained in the SOR instance	Integer	One (mandatory)
keyword	Keyword describing the content of the SOR instance.	Character string representing a keyword describing the content of the SOR instance	One or more (mandatory)
applicationDomain	Description of the application domain to which the SOR can be applied.	Character string representing the name of an application domain	One or more (mandatory)
ontologyRepository URL	URL pointing to a repository containing an ontology used by the SOR instance.	Character string representing a URL	One or more (mandatory)

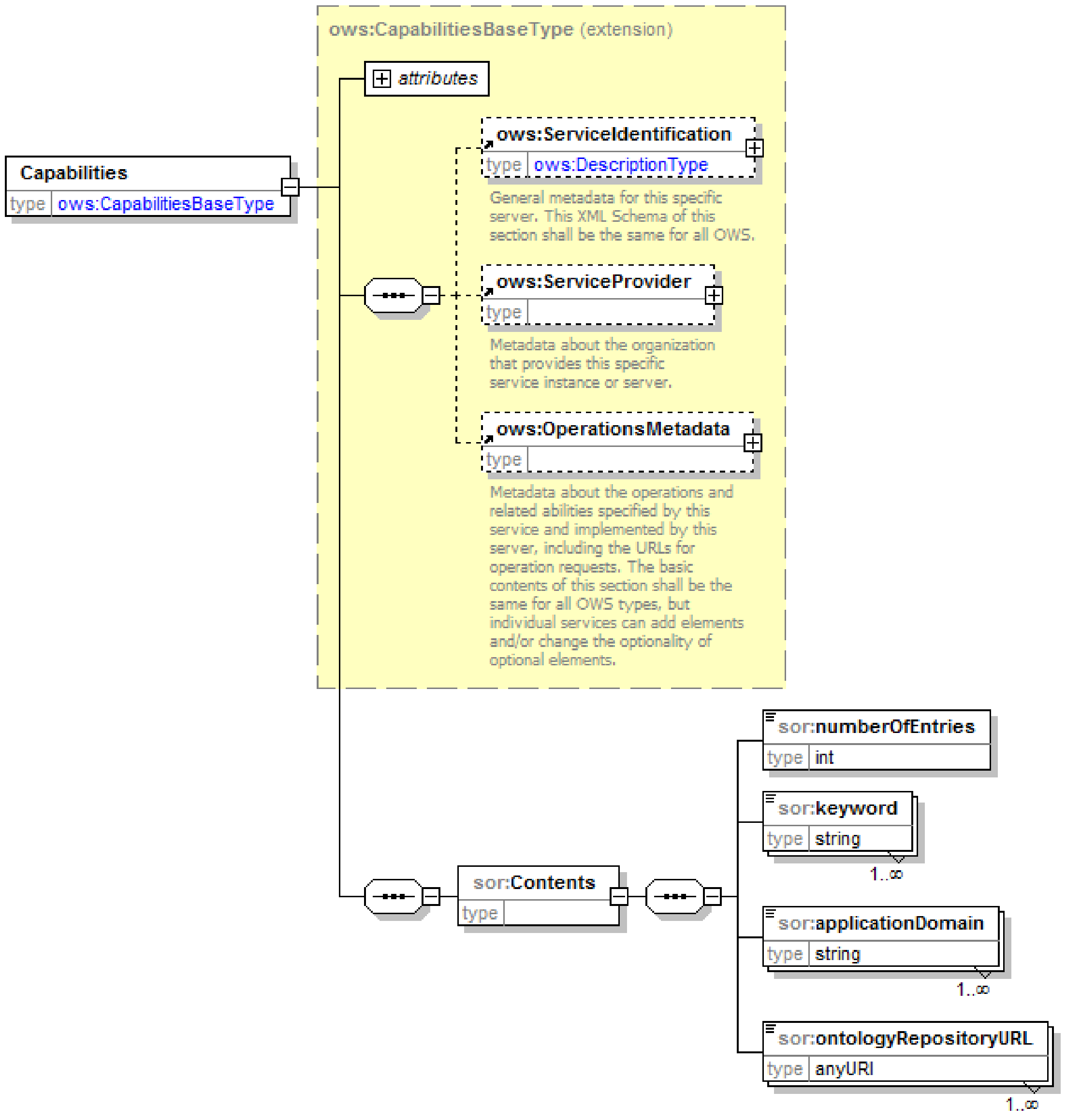

ows:CapabilitiesBaseType (extension)
attributes
Capabilities
type ows:CapabilitiesBaseType
ows:ServiceIdentification
type ows:DescriptionType
General metadata for this specific server. This XML Schema of this section shall be the same for all OWS.
ows:ServiceProvider
type
Metadata about the organization that provides this specific service instance or server.
ows:OperationsMetadata
type
Metadata about the operations and related abilities specified by this service and implemented by this server, including the URLs for operation requests. The basic contents of this section shall be the same for all OWS types, but individual services can add elements and/or change the optionality of optional elements.
sor:Contents
type
sor:numberOfEntries
type int
sor:keyword
type string
1..∞
sor:applicationDomain
type string
1..∞
sor:ontologyRepositoryURL
type anyURI
1..∞

Figure 1: SOR Capabilities shown in XMLSpy notation.

The following schema fragment specifies the contents and structure of a SOR 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:sor="http://swsl.uni-
muenster.de/sor" targetNamespace="http://swsl.uni-muenster.de/sor"
elementFormDefault="qualified" version="1.0.0" xml:lang="en">
    <import namespace="http://www.opengis.net/ows/1.1"
            schemaLocation="../ows/1.1.0/owsAll.xsd"/>
    <element name="Capabilities">
        <complexType>
            <complexContent>
```

```
                <extension base="ows:CapabilitiesBaseType">
                    <sequence>
                        <element name="Contents">
                            <complexType>
                                <sequence>
                                    <element name="numberOfEntries"
                                             type="int"/>
                                    <element name="keyword" type="string"
                                             maxOccurs="unbounded"/>
                                    <element name="applicationDomain"
                                             type="string"
                                             maxOccurs="unbounded"/>
                                    <element name="ontologyRepositoryURL"
                                             type="anyURI"
                                             maxOccurs="unbounded"/>
                                </sequence>
                            </complexType>
                        </element>
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>
</schema>
```

A SOR Capabilities document can look like this encoded in XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<sor:Capabilities xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:sor="http://swsl.uni-muenster.de/sor"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://swsl.uni-muenster.de/sor
./sorGetCapabilities.xsd " updateSequence="2009-08-31T13:12:39+02"
version="1.0.0">
    <ows:ServiceIdentification>
        <ows:Title>WWU SOR</ows:Title>
        <ows:Abstract>SOR Server Operated by WWU</ows:Abstract>
        <ows:ServiceType codeSpace=
            "http://swsl.uni-muenster.de/sor">OGC:SOR</ows:ServiceType>
        <ows:ServiceTypeVersion>0.3.0</ows:ServiceTypeVersion>
        <ows:Fees>NONE</ows:Fees>
        <ows:AccessConstraints>NONE</ows:AccessConstraints>
    </ows:ServiceIdentification>
    <ows:ServiceProvider>
        <ows:ProviderName>WWU</ows:ProviderName>
        <ows:ProviderSite xlink:href="http://swsl.uni-muenster.de"/>
        <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>
    <ows:Operation name="GetCapabilities">
        <ows:DCP>
            <ows:HTTP>
                <ows:Get xlink:href=
                    "http://swsl.uni-muenster.de/sor/GetCapabilities"/>
                <ows:Post xlink:href=
                    "http://swsl.uni-muenster.de/sor/GetCapabilities"/>
            </ows:HTTP>
        </ows:DCP>
        <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:Operation>
    <ows:Operation name="GetDefinition">
        <ows:DCP>
            <ows:HTTP>
                <ows:Get xlink:href=
                   "http://swsl.uni-muenster.de/sor/GetGetDefinition"/>
                <ows:Post xlink:href=
                   "http://swsl.uni-muenster.de/sor/GetGetDefinition"/>
            </ows:HTTP>
        </ows:DCP>
    </ows:Operation>
    <ows:Operation name="GetDefinitionURNs">
        <ows:DCP>
            <ows:HTTP>
                <ows:Get xlink:href=
                  "http://swsl.uni-muenster.de/sor/GetDefinitionURNs"/>
                <ows:Post xlink:href=
                  "http://swsl.uni-muenster.de/sor/GetDefinitionURNs"/>
            </ows:HTTP>
        </ows:DCP>
    </ows:Operation>
    <ows:Operation name="GetMatchingDefinitions">
        <ows:DCP>
            <ows:HTTP>
                <ows:Get xlink:href=
            "http://swsl.uni-muenster.de/sor/GetMatchingDefinitions"/>
                <ows:Post xlink:href=
            "http://swsl.uni-muenster.de/sor/GetMatchingDefinitions"/>
            </ows:HTTP>
        </ows:DCP>
    </ows:Operation>
    <ows:Parameter name="service">
        <ows:AllowedValues>
            <ows:Value>SOR</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>
    <sor:Contents>
       <sor:numberOfEntries>298</sor:numberOfEntries>
       <sor:keyword>weather</sor:keyword>
       <sor:keyword>climate</sor:keyword>
       <sor:keyword>meteorology</sor:keyword>
       <sor:applicationDomain>meteorology</sor:applicationDomain>
    <sor:ontologyRepositoryURL>http://sweet.jpl.nasa.gov/1.1/
    </sor:ontologyRepositoryURL>
    </sor:Contents>
</sor:Capabilities>
```

6.3.1 Exceptions

When a SOR 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.

7 GetDefinitionURNs operation (mandatory)

7.1 Introduction

The GetDefinitionURNs operation allows SOR clients to retrieve a list of URNs identifying definitions that are contained in a specific SOR instance. Optionally, a client is able to specify a text string that shall occur within the returned URNs. Furthermore a paging mechanism is supported for retrieving subsets of a longer list of URNs. This functionality is especially intended for supporting the development of client user interfaces as it allows to create lists from which users can select the URNs of those phenomenon definitions they are interested in.

7.2 GetDefinitionURNs operation request

7.2.1 GetDefinitionURNs request parameters

A request to perform the GetDefinitionURNs operation shall include the data structure specified in Table 4.

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 4 — Parameters in GetDefinitionURNs operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “SOR”)	One (mandatory)
request	Operation name	Character String type, not empty Value is “GetDefinitionURNs”)	One (mandatory)
version	Standard version for operation	Character String type, not empty Value is “0.3.0”.	One (mandatory)
maxNumberOfResults	Number of results (entries) that shall be returned	Integer type, 1 or higher (default is 10)	One (optional)
startResultElement	Number of the first result element that shall be returned	Integer type, 0 or higher (default is 0)	One (optional)
searchString	Substring that shall occur within the definition URNs to be returned.	Character String type, not empty	One (optional)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 06-121r3].			

7.2.2 GetDefinitionURNs request KVP encoding (mandatory)

Servers may implement HTTP GET transfer of the GetDefinitionURNs operation request, using KVP encoding. The KVP encoding of the GetDefinitionURNs operation request shall use the parameters specified in Table 5. The parameters listed in Table 5 shall be as specified in Table 4 above.

Table 5 — GetDefinitionURNs operation request URL parameters

Name and example [a]	Optionality and use	Definition and format
service=SOR	Mandatory	Service type identifier
request= GetDefinitionURNs	Mandatory	Operation name
version=0.3.0	Mandatory	Standard and schema version for this operation
maxNumberOfResults=20	Optional	Number of results (entries) that shall be returned
startResultElement=0	Optional	Number (integer) of the first result element that shall be returned
searchString=temperature	Optional	Substring that shall occur within the definition URNs to be returned.
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 GetDefinitionURNs operation request KVP encoded for HTTP GET is:

```
http://v-swe.uni-
muenster.de:8080/SOR/SOR?service=SOR&request=GetDefinitionURNs&version=0.3.0&
maxNumberOfResults=20&startResultElement=0&searchString=temperature
```

7.2.3 GetDefinitionURNs request XML encoding (mandatory)

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

A GetDefinitionURNs operation request is encoded in XML as shown in Figure 2.

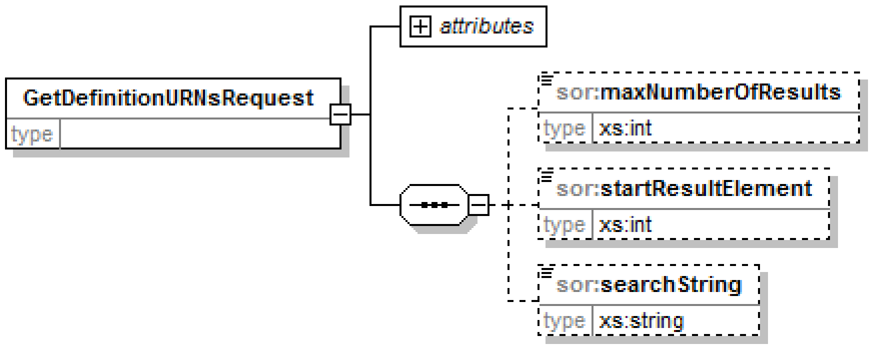


Figure 2: GetDefinitionURNs operation request shown in XMLSpy notation.

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

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sor="http://swsl.uni-muenster.de/sor "
```

```
xmlns:swe="http://www.opengis.net/swe/1.0"
targetNamespace="http://swsl.uni-muenster.de/sor"
elementFormDefault="qualified" attributeFormDefault="unqualified"
service="SOR" version="0.3.0">
   <xs:element name="GetDefinitionURNsRequest">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="maxNumberOfResults" type="xs:int"
                        minOccurs="0"/>
            <xs:element name="startResultElement" type="xs:int"
                        minOccurs="0"/>
            <xs:element name="searchString" type="xs:string"
                        minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="service" type="xs:string" use="required"
                       fixed="SOR"/>
         <xs:attribute name="version" type="xs:string" use="required"
                       fixed="0.3.0"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<GetDefinitionURNsRequest xmlns="http://swsl.uni-muenster.de/sor"
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/sor
sorGetDefinitionURNs.xsd">
   <maxNumberOfResults>20</maxNumberOfResults>
   <searchString>Temp</searchString>
</GetDefinitionURNsRequest>
```

7.3 GetDefinitionURNs operation response

7.3.1 Normal response parameters

The normal response to a valid GetDefinitionURNs operation request shall be a list of URNs identifying observable definitions. More precisely, a response from the GetDefinitionURNs operation shall include the parts listed in Table 6. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 6 — Parts of GetDefinitionURNs operation response

Names	Definition	Data type and values	Multiplicity and use
definitionURN	URNs identifying the definitions of phenomena contained in the SOR instance	Character string representing a URN	Zero or more

7.3.2 Normal response XML encoding

A GetDefinitionURNs operation response is encoded in XML as shown in Figure 3.

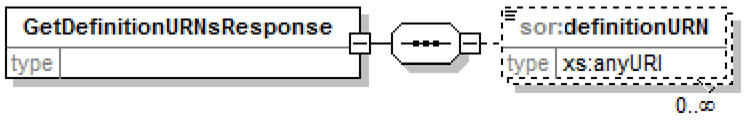


Figure 3: GetDefinitionURNs operation response shown in XMLSpy notation.

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

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sor="http://swsl.uni-muenster.de/sor"
xmlns:swe="http://www.opengis.net/swe/1.0"
targetNamespace="http://swsl.uni-muenster.de/sor"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:element name="GetDefinitionURNsResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="definitionURN" type="xs:anyURI"
                        minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

7.3.3 GetDefinitionURNs response example

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

```
<?xml version="1.0" encoding="UTF-8"?>
<GetDefinitionURNsResponse xmlns="http://swsl.uni-muenster.de/sor"
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/sor
sorGetDefinitionURNs.xsd">
  <definitionURN>urn:x-
               osiris:def:phenomenon:OSIRIS:Temperature</definitionURN>
  <definitionURN>urn:x-
            osiris:def:phenomenon:OSIRIS:AirTemperature</definitionURN>
  <definitionURN>urn:x-
          osiris:def:phenomenon:OSIRIS:WaterTemperature</definitionURN>
</GetDefinitionURNsResponse>
```

8 GetDefinition operation (mandatory)

8.1 Introduction

The GetDefinition operation allows SOR clients to to retrieve the definition of an observable identified by a given URNs. This definition is returned as a GML dictionary entry.

8.2 GetDefinition operation request

8.2.1 GetDefinition request parameters

A request to perform the GetDefinition operation shall include the data structure specified in Table 7.

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 7 — Parameters in GetDefinition operation request

Names [a]	Definition	Data type and values	Multiplicity and use
service	Service type identifier	Character String type, not empty Value is “SOR”)	One (mandatory)
request	Operation name	Character String type, not empty Value is “GetDefinition”)	One (mandatory)
version	Standard version for operation	Character String type, not empty Value is “0.3.0”.	One (mandatory)
inputURN	URN of the phenomenon for which a definition (dictionary) entry shall be retrieved	Character string representing a URN	One (mandatory)
a The name capitalization rules being used here are specified in Subclause 11.6.2 of [OGC 06-121r3].			

8.2.2 GetDefinition request KVP encoding (mandatory)

Servers may implement HTTP GET transfer of the GetDefinition operation request, using KVP encoding. The KVP encoding of the GetDefinition operation request shall use the parameters specified in Table 8. The parameters listed in Table 8 shall be as specified in Table 7 above.

Table 8 — GetDefinition operation request URL parameters

Name and example [a]	Optionality and use	Definition and format
service=SOR	Mandatory	Service type identifier
request= GetDefinition	Mandatory	Operation name

version=0.3.0	Mandatory	Standard and schema version for this operation
inputURN= urn:x-osiris:def:phenomenon:OSIRIS:Temperature	Mandatory	URN of the phenomenon for which a definition (dictionary) entry shall be retrieved
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 GetDefinition operation request KVP encoded for HTTP GET is:

```
http://v-swe.uni-muenster.de:8080/SOR/SOR?service=SOR&request=GetDefinition&
version=0.3.0&inputURN=urn:x-osiris:def:phenomenon:OSIRIS:Temperature
```

8.2.3 GetDefinition request XML encoding (mandatory)

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

A GetDefinition operation request is encoded in XML as shown in Figure 4.

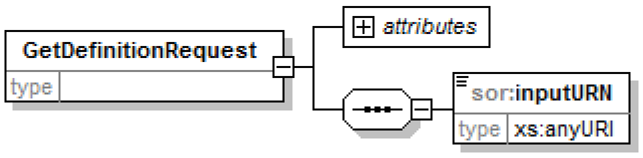


Figure 4: GetDefinition operation request shown in XMLSpy notation.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sor="http://swsl.uni-muenster.de/sor"
xmlns:swe="http://www.opengis.net/swe/1.0/gml32"
targetNamespace="http://swsl.uni-muenster.de/sor"
elementFormDefault="qualified" attributeFormDefault="unqualified"
service="SOR" version="0.3.0">
    <xs:import namespace="http://www.opengis.net/swe/1.0/gml32"
              schemaLocation="../sweCommon/1.0.1_gml32/phenomenon.xsd"/>
    <xs:element name="GetDefinitionRequest">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="inputURN" type="xs:anyURI"/>
            </xs:sequence>
            <xs:attribute name="service" type="xs:string" use="required"
                          fixed="SOR"/>
            <xs:attribute name="version" type="xs:string" use="required"
                          fixed="0.3.0"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<GetDefinitionRequest xmlns="http://swsl.uni-muenster.de/sor"
xmlns:gml="http://www.opengis.net/gml"
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/sor
sorGetDefinition.xsd">
   <inputURN>urn:x-osiris:def:phenomenon:OSIRIS:Temperature</inputURN>
</GetDefinitionRequest>
```

8.3 GetDefinition operation response

8.3.1 Normal response parameters

The normal response to a valid GetDefinition operation request shall be a GML dictionary entry containing the description corresponding to the URN that was submitted in the request. More precisely, a response from the GetDefinition operation shall include the parts listed in Table 9. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 9 — Parts of GetDefinition operation response

Names	Definition	Data type and values	Multiplicity and use
inputURN	URN of the phenomenon for which the definition (dictionary entry) was requested	Character string representing a URN	One (mandatory)
Phenomenon	Dictionary entry containing the description of a phenomenon, a reference to an ontology, the identifier as well as the name of the phenomenon.	SWE Common Phenomenon	One (mandatory)

8.3.2 Normal response XML encoding

A GetDefinition operation response is encoded in XML as shown in Figure 5.

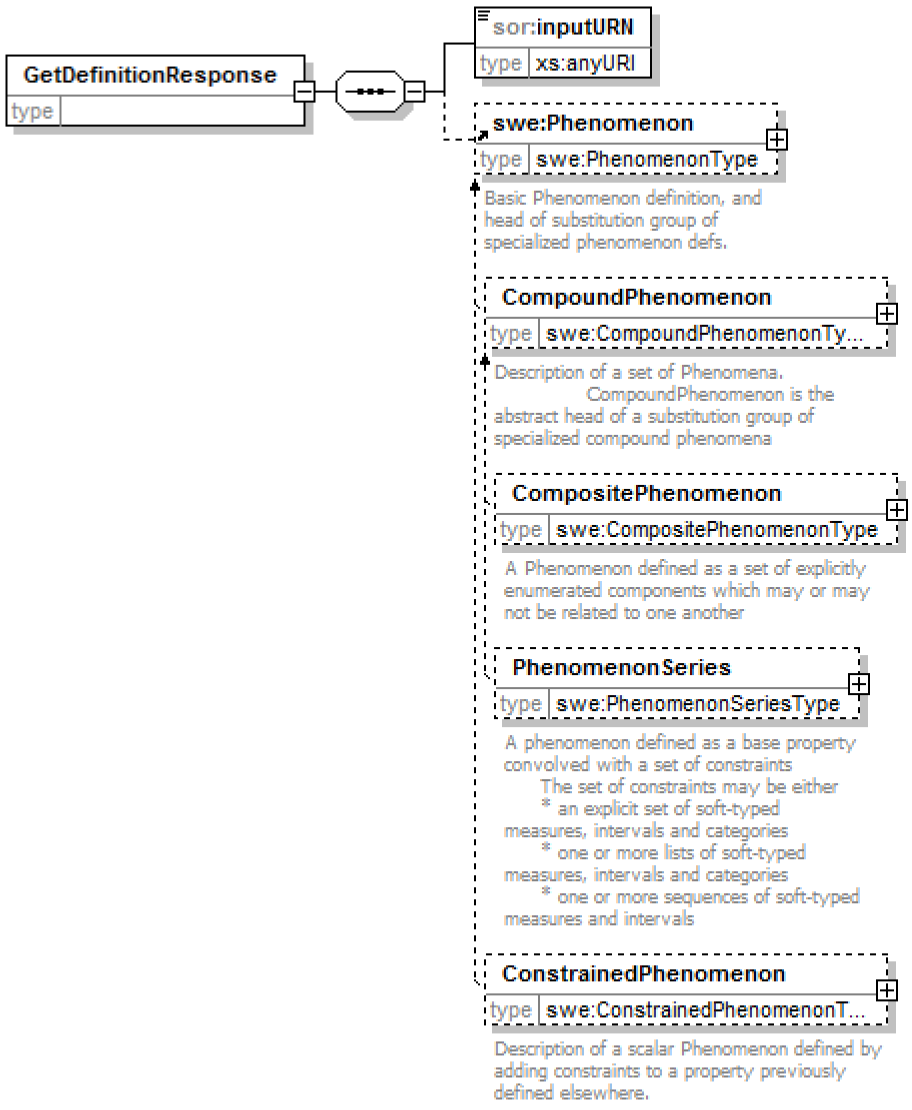

GetDefinitionResponse
type
sor:inputURN
type xs:anyURI
swe:Phenomenon
type swe:PhenomenonType
Basic Phenomenon definition, and head of substitution group of specialized phenomenon defs.
CompoundPhenomenon
type swe:CompoundPhenomenonTy...
Description of a set of Phenomena. CompoundPhenomenon is the abstract head of a substitution group of specialized compound phenomena
CompositePhenomenon
type swe:CompositePhenomenonType
A Phenomenon defined as a set of explicitly enumerated components which may or may not be related to one another
PhenomenonSeries
type swe:PhenomenonSeriesType
A phenomenon defined as a base property convolved with a set of constraints The set of constraints may be either * an explicit set of soft-typed measures, intervals and categories * one or more lists of soft-typed measures, intervals and categories * one or more sequences of soft-typed measures and intervals
ConstrainedPhenomenon
type swe:ConstrainedPhenomenonT...
Description of a scalar Phenomenon defined by adding constraints to a property previously defined elsewhere.

Figure 5: GetDefinition operation response shown in XMLSpy notation.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sor="http://swsl.uni-muenster.de/sor"
xmlns:swe="http://www.opengis.net/swe/1.0/gml32"
targetNamespace="http://swsl.uni-muenster.de/sor"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:import namespace="http://www.opengis.net/swe/1.0/gml32"
            schemaLocation="../sweCommon/1.0.1_gml32/phenomenon.xsd"/>
    <xs:element name="GetDefinitionResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="inputURN" type="xs:anyURI"/>
                <xs:element ref="swe:Phenomenon" minOccurs="0"/>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
</xs:schema>
```

8.3.3 GetDefinition response example

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

```
<?xml version="1.0" encoding="UTF-8"?>
<GetDefinitionResponse xmlns="http://swsl.uni-muenster.de/sor"
xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:swe="http://www.opengis.net/swe/1.0/gml32"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://swsl.uni-muenster.de/sor
sorGetDefinition.xsd">
   <inputURN>urn:x-osiris:def:phenomenon:OSIRIS:Temperature</inputURN>
   <swe:Phenomenon gml:id="Temperature">
      <gml:description xlink:href="http://sweet.jpl.nasa.gov
            /ontology/property.owl#Temperature">The degree of hotness
            or coldness of a body or environment.</gml:description>
      <gml:identifier codeSpace="http://www.osiris-fp6.eu">urn:x-
            osiris:def:phenomenon:OSIRIS:Temperature</gml:identifier>
      <gml:name>Temperature</gml:name>
   </swe:Phenomenon>
</GetDefinitionResponse>
```

9 GetMatchingDefinitions operation (mandatory)

9.1 Introduction

The GetMatchingDefinitions operation allows SOR clients to retrieve the URNs of definitions of observables which are related to another given phenomenon in a certain way. Currently, the SOR supports the relations “generalization”, “specialization” and “equivalency”.

9.2 GetMatchingDefinitions operation request

9.2.1 GetMatchingDefinitions request parameters

A request to perform the GetMatchingDefinitions operation shall include the data structure specified in Table 10.

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 10 — Parameters in GetMatchingDefinitionsRequest operation request

Names [a]	**Definition**	**Data type and values**	**Multiplicity and use**
service service	Service type identifier	Character String type, not empty Value is “SOR”)	One (mandatory)
request request	Operation name	Character String type, not empty Value is “GetMatchingDefinitionsRequest”)	One (mandatory)
version version	Standard version for operation	Character String type, not empty Value is “0.3.0”.	One (mandatory)
inputURN	URN of the phenomenon for which related phenomena shall be retrieved	Character string representing a URN	One (mandatory)
matchingType	Type of relation between phenomena that shall be used for reasoning and finding matching phenomena	Character string Value can be “SUPER_CLASS”, “EQUIVALENT_CLASS” or “SUB_CLASS”	One (mandatory)
searchDepth	Number of intermediate step that are allowed in case of transitively related phenomena	Integer type, 0 or higher (default is 0)	Zero or one (optional)

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

9.2.2 GetMatchingDefinitions request KVP encoding (mandatory)

Servers may implement HTTP GET transfer of the GetMatchingDefinitions operation request, using KVP encoding. The KVP encoding of the GetMatchingDefinitions operation request shall use the parameters specified in Table 11. The parameters listed in Table 11 shall be as specified in Table 10 above.

Table 11 — GetMatchingDefinitions operation request URL parameters

Name and example [a]	Optionality and use	Definition and format
service=SOR	Mandatory	Service type identifier
request= GetMatchingDefinitionsRequest	Mandatory	Operation name
version=0.3.0	Mandatory	Standard and schema version for this operation
inputURN= urn:x-osiris:def:phenomenon:OSIRIS:Temperature	Mandatory	Character string representing a URN
matchingType= SUB_CLASS	Mandatory	Type of relation between phenomena that shall be used for reasoning and finding matching phenomena
searchDepth=1	Optional (default is 0)	Number of intermediate step that are allowed in case of transitively related phenomena
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 GetMatchingDefinitions operation request KVP encoded for HTTP GET is:

```
http://v-swe.uni-muenster.de:8080/SOR/SOR?service=SOR&
request=GetMatchingDefinitions& version=0.3.0&inputURN=urn:x-
osiris:def:phenomenon:OSIRIS:Temperature&matchingType=SUB_CLASS& searchDepth=1
```

9.2.3 GetMatchingDefinitions request XML encoding (mandatory)

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

A GetMatchingDefinitions operation request is encoded in XML as shown in Figure 6.

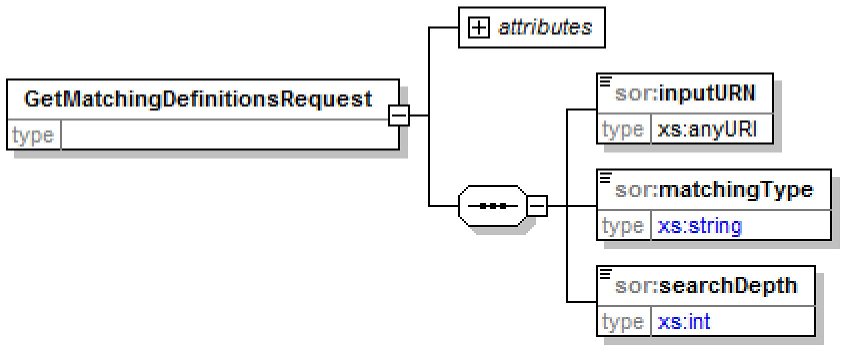


Figure 6: GetMatchingDefinitions operation request shown in XMLSpy notation.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sor="http://swsl.uni-muenster.de/sor"
xmlns:swe="http://www.opengis.net/swe/1.0"
xmlns:mcd="org.n52.reasoner.JenaReasoner."
targetNamespace="http://swsl.uni-muenster.de/sor"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:element name="GetMatchingDefinitionsRequest">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="inputURN" type="xs:anyURI"/>
                <xs:element name="matchingType">
                    <xs:simpleType id="MatchingCode">
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="SUPER_CLASS"/>
                            <xs:enumeration value="EQUIVALENT_CLASS"/>
                            <xs:enumeration value="SUB_CLASS"/>
                        </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:attribute name="service" type="xs:string" use="required"
                          fixed="SOR"/>
            <xs:attribute name="version" type="xs:string" use="required"
                          fixed="0.3.0"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

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

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

```
<GetMatchingDefinitionsRequest xmlns="http://swsl.uni-muenster.de/sor"
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/sor
sorGetMatchingDefinitions.xsd" service="SOR" version="0.3.0">
   <inputURN>urn:x-osiris:def:phenomenon:OSIRIS:Temperature</inputURN>
   <matchingType>SUB_CLASS</matchingType>
   <searchDepth>1</searchDepth>
</GetMatchingDefinitionsRequest>
```

9.3 GetMatchingDefinitions operation response

9.3.1 Normal response parameters

The normal response to a valid GetMatchingDefinitions operation request shall be a list of URNs of all observable definition that match the criteria submitted in the request. More precisely, a response from the GetMatchingDefinitions operation shall include the parts listed in Table 12. This table also specifies the UML model data type plus the multiplicity and use of each listed part.

Table 12 — Parts of GetMatchingDefinitions operation response

Names	Definition	Data type and values	Multiplicity and use
inputURN	URN of the phenomenon for which related phenomena were requested	Character string representing a URN	One (mandatory)
definitionURN	URNs identifying the definitions of phenomena that matched to the parameters specified in the request	Character string representing a URN	Zero or more

9.3.2 Normal response XML encoding

A GetMatchingDefinitions operation response is encoded in XML as shown in Figure 7.

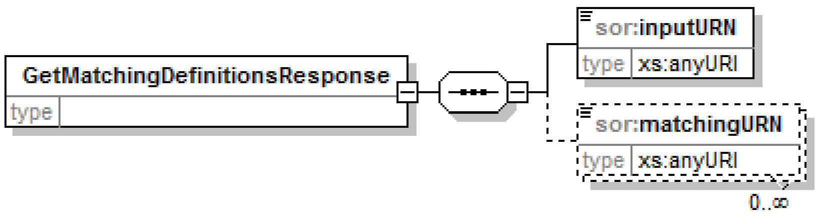


Figure 7: GetMatchingDefinitions operation response shown in XMLSpy notation.

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

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

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sor="http://swsl.uni-muenster.de/sor"
xmlns:swe="http://www.opengis.net/swe/1.0"
xmlns:mcd="org.n52.reasoner.JenaReasoner."
targetNamespace="http://swsl.uni-muenster.de/sor"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:element name="GetMatchingDefinitionsResponse">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="inputURN" type="xs:anyURI"/>
            <xs:element name="matchingURN" type="xs:anyURI"
                        minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

9.3.3 GetMatchingDefinitions response example

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

```
<?xml version="1.0" encoding="UTF-8"?>
<GetMatchingDefinitionsResponse xmlns="http://swsl.uni-muenster.de/sor"
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/sor
sorGetMatchingDefinitions.xsd">
   <inputURN>urn:x-osiris:def:phenomenon:OSIRIS:Temperature</inputURN>
   <matchingURN>urn:x-
         osiris:def:phenomenon:OSIRIS:AirTemperature</matchingURN>
   <matchingURN>urn:x-
         osiris:def:phenomenon:OSIRIS:WaterTemperature</matchingURN>
</GetMatchingDefinitionsResponse>
```

10 Implementation and Practical Application

10.1 Implementation

Within the OSIRIS project a first prototypical implementation was performed by Thales Communications. In addition, after the end of the project a further completely new implementation was created by the University of Muenster which is made available through the open source initiative 52° North[1].

For describing the definitions of observables, their URNs and the links to ontologies, the structure of the phenomenon dictionary as defined in the SWE Common specification is used as a basis. This dictionary provides means for mapping URNs to phenomenon definitions and for establishing the link to the according ontologies. Thus, based on this dictionary the GetDefinitionsURNs as well as GetDefinition can be realized.

In order to enable reasoning mechanisms which form the basis for the GetMarchingDefinitions operation the Jena Semantic Web Framework[2] is used. For being able to execute the reasoning mechanisms the observable definitions were linked to the SWEET ontology[3].

10.2 Example

10.2.1 Using the SOR as an Observable Dictionary

As stated before, one of the core functionalities of the SOR is to resolve URNs identifying the definitions of observables.

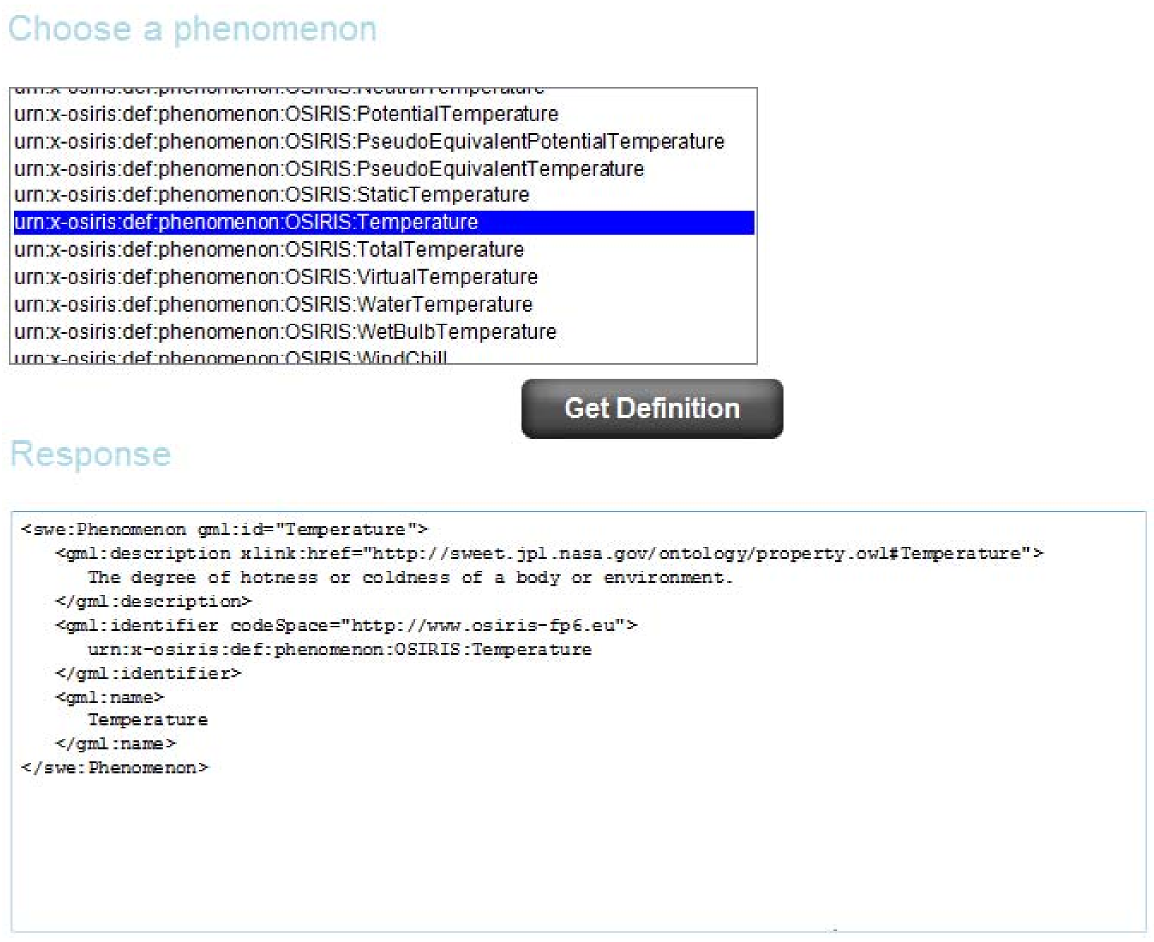
Figure 8 shows a screenshot of a web based client accessing an instance of the 52° North SOR implementation. The list of URNs in the upper half of the screenshot is generated by the client using the GetDefinitionURNs operations and allows users to select the URNs they want to resolve. The text field in the lower half of the screen is used for displaying the requested definitions.

In this example a user has requested the definition of the observable identified by the URN “urn:x-osiris:def:phenomenon:OSIRIS:Temperature”. The returned definition contains besides the id and name of the definition also a short full text explanation as well as a link to an ontology where the semantics of the observable is formally defined.

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

[2] http://jena.sourceforge.net/

[3] http://sweet.jpl.nasa.gov/ontology/



urn:x-osiris:def:phenomenon:OSIRIS:PotentialTemperature
urn:x-osiris:def:phenomenon:OSIRIS:PseudoEquivalentPotentialTemperature
urn:x-osiris:def:phenomenon:OSIRIS:PseudoEquivalentTemperature
urn:x-osiris:def:phenomenon:OSIRIS:StaticTemperature
urn:x-osiris:def:phenomenon:OSIRIS:Temperature
urn:x-osiris:def:phenomenon:OSIRIS:TotalTemperature
urn:x-osiris:def:phenomenon:OSIRIS:VirtualTemperature
urn:x-osiris:def:phenomenon:OSIRIS:WaterTemperature
urn:x-osiris:def:phenomenon:OSIRIS:WetBulbTemperature

```
<swe:Phenomenon gml:id="Temperature">
   <gml:description xlink:href="http://sweet.jpl.nasa.gov/ontology/property.owl#Temperature">
      The degree of hotness or coldness of a body or environment.
   </gml:description>
   <gml:identifier codeSpace="http://www.osiris-fp6.eu">
      urn:x-osiris:def:phenomenon:OSIRIS:Temperature
   </gml:identifier>
   <gml:name>
      Temperature
   </gml:name>
</swe:Phenomenon>
```

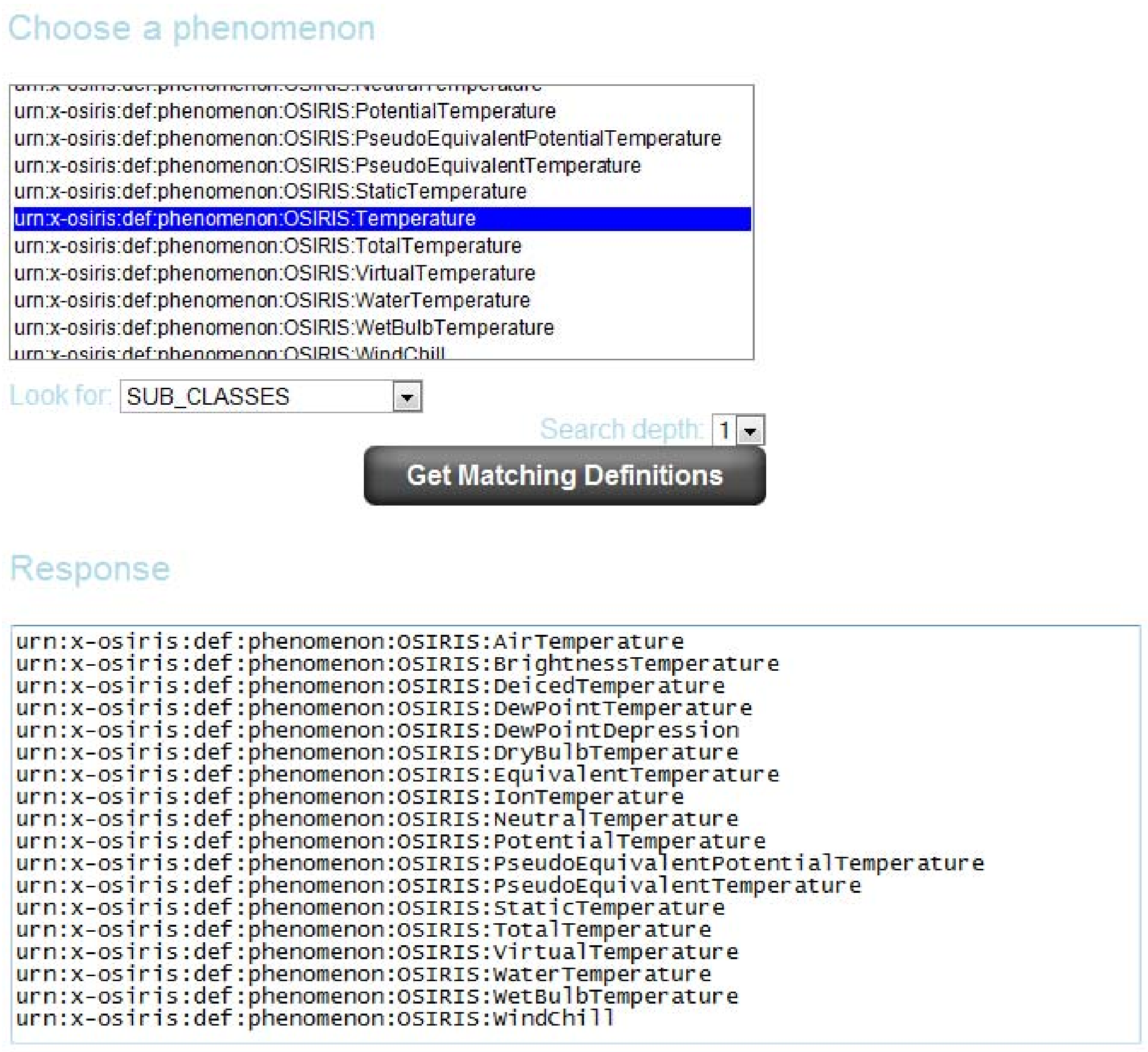
Figure 8: Screenshot showing a GetDefinition operation response

10.2.2 Using the SOR for Discovering Semantically Related Observables

The second main function of the SOR is to exploit the semantics of observables in order to find related observables. A basic web based client supporting this functionality is shown in Figure 9.

Like in the previous example the client offers the user a list of the observable URNs the SOR instance contains. Also in this case this list is generated using the GetDefinitionURNs operation. After having selected a URN the user is able to specify the relation that he is interested in (SUB_CLASSES, SUPER_CLASSES, EQUIVALENT_CLASSES) through a drop-down list and a search depth that specifies the number of allowed intermediate steps if transitively related observables shall be returned.

In the example below the user requests sub classes (with maximum one intermediate step) of the observable identified by the URN "urn:x-osiris:def:phenomenon:OSIRIS:Temperature". The URNs of the matching observable definitions are shown as a list in the lower half of the screenshot.



```
urn:x-osiris:def:phenomenon:OSIRIS:PotentialTemperature
urn:x-osiris:def:phenomenon:OSIRIS:PseudoEquivalentPotentialTemperature
urn:x-osiris:def:phenomenon:OSIRIS:PseudoEquivalentTemperature
urn:x-osiris:def:phenomenon:OSIRIS:StaticTemperature
urn:x-osiris:def:phenomenon:OSIRIS:Temperature
urn:x-osiris:def:phenomenon:OSIRIS:TotalTemperature
urn:x-osiris:def:phenomenon:OSIRIS:VirtualTemperature
urn:x-osiris:def:phenomenon:OSIRIS:WaterTemperature
urn:x-osiris:def:phenomenon:OSIRIS:WetBulbTemperature
```

Look for: SUB_CLASSES

Search depth: 1

```
urn:x-osiris:def:phenomenon:OSIRIS:AirTemperature
urn:x-osiris:def:phenomenon:OSIRIS:BrightnessTemperature
urn:x-osiris:def:phenomenon:OSIRIS:DeicedTemperature
urn:x-osiris:def:phenomenon:OSIRIS:DewPointTemperature
urn:x-osiris:def:phenomenon:OSIRIS:DewPointDepression
urn:x-osiris:def:phenomenon:OSIRIS:DryBulbTemperature
urn:x-osiris:def:phenomenon:OSIRIS:EquivalentTemperature
urn:x-osiris:def:phenomenon:OSIRIS:IonTemperature
urn:x-osiris:def:phenomenon:OSIRIS:NeutralTemperature
urn:x-osiris:def:phenomenon:OSIRIS:PotentialTemperature
urn:x-osiris:def:phenomenon:OSIRIS:PseudoEquivalentPotentialTemperature
urn:x-osiris:def:phenomenon:OSIRIS:PseudoEquivalentTemperature
urn:x-osiris:def:phenomenon:OSIRIS:StaticTemperature
urn:x-osiris:def:phenomenon:OSIRIS:TotalTemperature
urn:x-osiris:def:phenomenon:OSIRIS:VirtualTemperature
urn:x-osiris:def:phenomenon:OSIRIS:WaterTemperature
urn:x-osiris:def:phenomenon:OSIRIS:WetBulbTemperature
urn:x-osiris:def:phenomenon:OSIRIS:WindChill
```

Figure 9: Screenshot showing a GetMatchingDefinitions operation response

10.2.3 Integrating the SOR with a Sensor Registry/Catalogue

Finally, this third example shall illustrate how the SOR functionality, especially the GetMatchingDefinitions operation, can be used for improving sensor discovery mechanism. During the OSIRIS project, the enhancement of sensor discovery mechanisms relying on semantics was the driving factor for developing the SOR.

Figure 10 gives an overview how a sensor discovery service (a sensor registry/catalogue) can be combined with the SOR in order to achieve the previously mentioned sensor discovery enhancements.

In the example a user, who searches for a specific kind of sensor, submits his search request to a sensor registry/catalogue. Within this search request the observable for which the user wants the find sensor data is stated in form of an according URN. If the user enables the use of the SOR by setting according parameters the sensor registry/catalogue directly queries the SOR for the URNs of those observables which are related to the given observable. After the sensor registry/catalogue has received this list of URNs it queries its internal sensor index for all sensors that measure an observable with any of these URNs. Subsequently registry/catalogue entries for all matching sensors are returned.

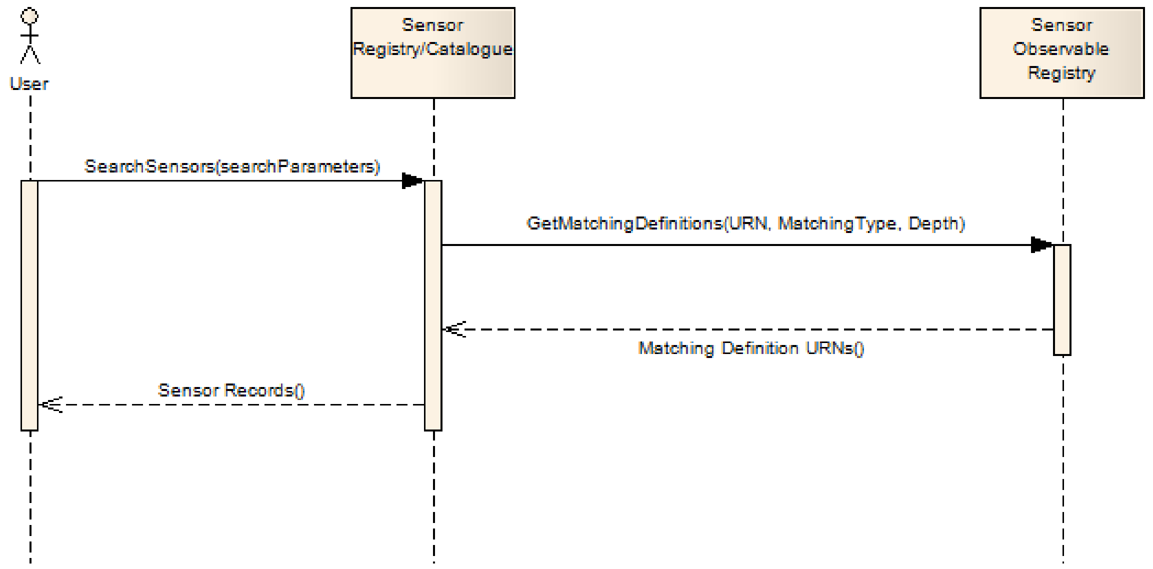

User
Sensor Registry/Catalogue
Sensor Observable Registry
SearchSensors(searchParameters)
GetMatchingDefinitions(URN, MatchingType, Depth)
Matching Definition URNs()
Sensor Records()

Figure 10: Intgration of the SOR into a Sensor Registry/Catalogue

Bibliography

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

[2] OpenGIS® Geography Markup Language (GML) Encoding Standard, OGC document 07-036

[3] Janowicz, K.; Kessler, C.; Bröring, A.; Stasch, C.; Schade, S.. 2009. "Towards Semantic Enablement for Spatial Data Infrastructures". EuroSSC 2009: 4th European Conference on Smart Sensing and Context.

[4] 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

[5] Lutz, M.; Klien, E.. 2004. "Ontology-based retrieval of geographic information". Int. J. Geogr. Inf. Sci. 2004, 20, 233-260.

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

[7] NASA SWEET ontology (http://sweet.jpl.nasa.gov/ontology/)