Project No. 249024

NETMAR

Open service network for marine environmental data

Instrument: *Please tick*	CA	✓**STREP**	IP	NOE

ICT - Information and Communication Technologies Theme

D4.5 – Authoritative specification of semantic framework

Reference: D4.5_Authoritative_specification_of_semantic_framework_r1_20121130

Due date of deliverable (as in Annex 1): M0 + 33
Actual submission date: 30 November 2012

Start date of project: 1 February 2010 Duration: 3 years

Coastal and Marine Resources Centre (CMRC), University College Cork, National University of Ireland

Revision 1

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

European Commission
Information Society and Media

	**NETMAR** Open service network for marine environmental data Project Reference: 249024 Contract Type: Collaborative Project Start/End Date: 01/03/2010 - 31/01/2013 Duration: 36 months
	Coordinator: Prof. Stein Sandven Nansen Environmental and Remote Sensing Center Thormøhlensgate 47, Bergen, Norway Tel.: +47-55205800 Fax. +47 55 20 58 01 E-mail: stein.sandven@nersc.no

Acknowledgements

The work described in this report has been partially funded by the European Commission under the Seventh Framework Programme, Theme ICT 2009.6.4 ICT for environmental services and climate change adaptation.

Consortium

The NETMAR Consortium is comprised of:

- Nansen Environmental and Remote Sensing Center (NERSC), Norway (coordinator).
  Project Coordinator: Prof. Stein Sandven (stein.sandven@nersc.no)
  Deputy Coordinator: Dr. Torill Hamre (torill.hamre@nersc.no)
  Quality Control Manager: Mr. Lasse H. Pettersson (lasse.pettersson@nersc.no)
- British Oceanographic Data Centre (BODC), National Environment Research Council, United Kingdom
  Contact: Dr. Roy Lowry (rkl@bodc.ac.uk)
- Centre de documentation de recherche et d'expérimentations sur les pollutions accidentelles des eaux (Cedre), France.
  Contact: Mr. François Parthiot (Francois.Parthiot@cedre.fr)
- Coastal and Marine Resources Centre (CMRC), University College Cork, National University of Ireland, Cork, Ireland.
  Contact: Mr. Declan Dunne (d.dunne@ucc.ie)
- Plymouth Marine Laboratory (PML), United Kingdom.
  Contact: Mr. Steve Groom (sbg@pml.ac.uk)
- Institut français de recherche pour l'exploitation de la mer (Ifremer), France.
  Contact: Mr. Mickael Treguer (mickael.treguer@ifremer.fr)
- Norwegian Meteorological Institute (METNO), Norway.
  Contact: Mr. Øystein Torget (oysteint@met.no)

Author(s)

- Yassine Lassoued, UCCNUIC (CMRC), y.lassoued@ucc.ie

Document approval

- Document status: Revision 1
- WP leader approval: 2012-11-19
- Quality Manager approval: 2012-11-30
- Coordinator approval: 2012-11-30

Semantic Framework Specification

Version 2.0

Document Version 2012.11.30

Content developed under the EU FP7 NETMAR project
Open Service Network for Marine Environmental Data (NETMAR)
EU FP7 249024

Semantic Framework Specification	Version: 2.0
GEOSS Best Practice Document	2012.11.30

Author(s)

Yassine Lassoued

Coastal and Marine Research Centre (CMRC)
University College Cork
y.lassoued@ucc.ie

Document approval

Document status: First revision
Quality Manager approval:

Acknowledgements

The work described in this document has been partially funded by the European Commission under the Seventh Framework Programme, Theme ICT 2009.6.4 ICT for environmental services and climate change adaptation.

Revision History

Issue	Date	Editor/Author	Change Records
Draft	2012.08.13	Yassine Lassoued	Initial Draft
	2012.11.01	External Expert	Review
Revision 1	2012.11.19	Yassine Lassoued	Revised document according to external expert's review
	2012.11.29	Torill Hamre	Review
Revision 2	2012.11.30	Yassine Lassoued	Revised document according to Torill Hamre's review

Executive Summary

This document provides a formal specification for the semantic framework designed and implemented as part of the NETMAR project for discovering, accessing and chaining (marine) environmental web services and defines a set of practical semantics use cases. The semantic framework specification defines the interface, called ***Semantic Web Service (SWS)***, for accessing and querying semantic resources using HTTP as the distributed computing platform. Via the SWS, a web user or service can use and combine semantic knowledge.

The semantic framework specification defines a set of practical use cases for semantic knowledge, which are:

1. Ontology Browsing: graphically navigate an ontology or a thesaurus in order to understand the meaning of the concepts (ideas represented by terms) contained therein and to find out how these relate to each other (related, narrower or broader concepts, etc.).
2. Product Discovery: improve the pertinence of the search results of catalogue services, by exploiting the semantic relationships between terms (narrower, related, same as, etc.), and/or trying to interpret the meaning of a user free text keyword according to a given thesaurus.
3. Interoperability: Facilitate the interoperability of two or more information systems (e.g., catalogue services, etc.) using heterogeneous data structures and semantics.
4. Service Chaining: Use semantic knowledge to ensure that the inputs and outputs of each component of a service chain are "semantically compatible."

The SWS interface specification builds on existing work and tries to cover the types of operations required by most common use cases and supported by existing vocabulary services (NVS, GEMET, SISSvoc, MMI).

The semantic web service specified in this document supports the following operations:

1. ***GetCapabilities***
   Retrieves service metadata, including supported operations, response formats, available concept schemes, and their supported languages.
2. ***GetConceptSchemes***
   Lists available concept schemes with their annotations (labels, definitions, etc.).
3. ***GetConceptScheme***
   Returns a concept scheme definition given its URI. The response includes the concept scheme's annotations.
4. ***SearchConceptScheme***
   Returns the definition(s) of one or more concept scheme(s) matching a specified free-text keyword.
5. ***GetConceptSchemeContent***
   Returns the content of a given concept scheme (identified by its URI), including its collections and concepts.
6. ***GetCollections***
   Lists available concept collections with their annotations. Collections may be filtered by one or more concept schemes.
7. ***GetCollection***
   Returns a collection definition identified by its URI. The response includes the collection's annotations.
8. ***SearchCollection***
   Returns the definition(s) of one or more collection(s) matching a specified free-text keyword.
9. ***GetCollectionContent***

Returns the content of a given collection (identified by its URI), including member collections and concepts.

10. ***GetConcepts***
    Returns the definitions of the concepts belonging to a specified concept scheme and/or collection.
11. ***GetConcept***
    Returns a concept definition given its URI. The response includes the concept's annotations.
12. ***SearchConcept***
    A search operation that returns the concepts that textually match a given keyword.
13. ***GetRelatedConcepts***
    Returns the concepts related to one or many given concept(s) using one or many given SKOS relationship(s) (e.g., skos:narrower, skos:broader, skos:related, etc.), both from direct assertions and by entailment.
14. ***GetExplicitTopConcepts***
    Returns the concepts that have explicitly been asserted as top concepts of a specified concept scheme.
15. ***GetImplicitTopConcepts***
    Returns the top-level concepts of a specified concept scheme.
16. ***GetConceptHierarchy***
    This operation is suitable for small thesauri, and is useful for ontology browsers. It returns the hierarchy of the concepts within a given concept scheme and/or collection.
17. ***InterpretKeyword***
    Returns the concepts that semantically match a given keyword, within a specified concept scheme and or collection.
18. ***CheckRelation***
    Checks whether two specified concepts are related via a specified SKOS relationship.

The SWS specification aims to pave the road to the specification of a standard semantic web service for spatial data infrastructures.

Contents

1 Introduction

1.1 Context

The content of this document has been developed under the EU FP7 NETMAR[1] project (EU FP7 249024). NETMAR aims to develop a pilot European Marine Information System (EUMIS) for searching, downloading and integrating satellite, in situ and model data from ocean and coastal areas. EUMIS is a user-configurable system offering flexible service discovery, access and chaining facilities using OGC, OPeNDAP and W3C standards. It uses a semantic framework (SF) coupled with ontologies for identifying and accessing distributed data, such as near-real time, forecast and historical data. EUMIS also enables further processing of such data to generate composite products and statistics suitable for decision-making in diverse marine application domains.

1.2 Problem Addressed

Several projects and organisations are developing and maintaining large controlled vocabularies and ontologies, and web services for accessing and querying these valuable and complex resources. Despite the momentum gained by semantic technologies in the scientific and spatial data infrastructure (SDI) communities, two fundamental problems still need to be addressed:

1. The use of the developed vocabularies and ontologies and the exploitation of their full power remain very limited. Despite the advances achieved by the IT and the semantic web communities in semantic web technologies and applications, the use of ontologies in the scientific and SDI communities is often limited to metadata or data semantic annotation and thesaurus browsing. There is an obvious lack of practical ontology use cases, applications, and semantically enabled information systems in SDIs; and ontologies often remain used as mere dictionaries or controlled vocabularies.
2. Several semantic web services currently exist, e.g., the NERC Vocabulary Server[2] (NVS), the General Multilingual Thesaurus[3] (GEMET), the SISS Vocabulary Server[4] (SISSvoc), the Marine Metadata Interoperability (MMI) Semantic Framework[5], etc. They all aim at providing a high-level web interface for interrogating SKOS thesauri. But the lack of a standard specification for this type of services has led to disparate models and web interfaces, which makes the integration of this type of services, in order to build integrated cross-domain semantic knowledge, a challenge. A standard or common web interface specification is needed in order to facilitate the interoperability of semantic web services.

The work described in this document aims to help address the problems outlined above by defining a set of practical use cases for semantic knowledge, and by paving the road to the specification of a standard semantic web service for SDIs. To initiate this process, the specification was submitted to the GEOSS Best Practices Wiki in mid August 2012. Through the interaction with the semantic web and ontology experts in GEOSS, the semantic framework specification will be refined and enhanced according to feedback from the GEOSS Best Practices Wiki. In parallel, the specification is being implemented and tested within running RTD projects. The semantic framework software will be updated to reflect changes to the specification, and serve as a reference implementation for the final semantic framework specification.

[1] http://www.netmar-project.eu/

[2] http://vocab.nerc.ac.uk/

[3] https://svn.eionet.europa.eu/projects/Zope/wiki/GEMETWebServiceAPI

[4] https://www.seegrid.csiro.au/wiki/Siss/SISSvoc30Specification

[5] https://marinemetadata.org/semanticframework

1.3 Objective of this Document

This document provides a formal specification for the semantic framework designed and implemented as part of the NETMAR project for discovering, accessing and chaining (marine) environmental web services and defines a set of practical semantics use cases. The semantic framework specification defines the interface, called ***Semantic Web Service (SWS)***, for accessing and querying semantic resources using HTTP as the distributed computing platform. Via the SWS, a web user or service can use and combine semantic knowledge.

The SWS interface specification builds on existing work and tries to cover the types of operations required by most common use cases (ontology browsing, data and service discovery, interoperability, and service chaining) and supported by existing vocabulary services (NVS, GEMET, SISSvoc, MMI).

1.4 Scope

The NETMAR semantic framework has been tested in four real-world case studies[6]:

1.1. Arctic Sea Ice and Met-ocean Observing System,
1.2. Near real time monitoring and forecasting of oil spill,
1.3. Ocean Colour - Marine Ecosystem, Research and Monitoring,
1.4. International Coastal Atlas Network (ICAN) for coastal zone management.

This said, the specification aims to respond to a variety of generic and practical use cases, such as data and metadata semantic annotation, data and service discovery, and service chaining. It is, therefore, of wider relevance to spatial and non-spatial data infrastructures and information systems.

The NETMAR semantic framework has been designed to support the Simple Knowledge Organization System (SKOS) model. The selection of SKOS as a data model was based on the simplicity of this standard while offering enough expressiveness to respond to the needs of the most common use cases (product discovery, interoperability, service chaining, etc.).

1.5 Related Work

Several open source and commercial ontology frameworks and triple stores exist and integrate, or facilitate the implementation of, ontology web servers, usually based on SPARQL [PS08] as a protocol for querying semantic knowledge. For instance, Sesame[7] includes a REST-ful interface for querying ontologies using SPARQL. AllegroGraph[8] includes a REST-ful interface that supports SPARQL-like queries as input. The Jena ontology framework does not include a built-in server, but supports SPARQL, and therefore allows developers to implement a SPARQL-based web services for querying ontologies. For instance, the NERC Vocabulary Server (NVS)[9] has such a SPARQL-based web interface developed using the Jena framework. The Database Access and Integration Services Working Group (DAIS-WG) of the Open Grid Forum (OGF)[10] has submitted a specification for standard mechanism for accessing RDF(S) data [AAC09], also based on SPARQL.

While SPARQL provides a powerful and expressive protocol for querying general-purpose ontologies regardless of how they are structured and of the models they use, it remains difficult to use by non-computer science users, as it requires technical skills (similar to the use of SQL in database querying).

[6] http://www.netmar-project.eu/content/pilots

[7] http://www.openrdf.org/

[8] http://www.franz.com/agraph/allegrograph/

[9] http://vocab.nerc.ac.uk/

[10] http://www.gridforum.org/

In practice, several projects organise their domain knowledge as SKOS thesauri rather than using ontologies in the broadest sense of the term. The choice of SKOS as an ontology model trades complexity for structure while preserving a sufficient level of expressiveness. For instance, the SeaDataNet and NVS vocabularies, the NASA Global Change Master Directory (GCMD)[11], the USGS thesaurus, and GEMET are all based on SKOS. In such a case, it is possible to provide a simple and high-level query web interface, which would facilitate semantic knowledge querying while preserving a sufficient level of expressiveness. For instance, NVS, GEMET, and SISSvoc provide high-level web interfaces for interrogating SKOS thesauri.

The NETMAR semantic web service specification builds on these efforts and proposes a more complete list of operations that meet the requirements of most common SKOS-based ontology uses cases (c.f., Chapter 2).

The proposed semantic web service may be provided in addition to a REST-ful interface that allows direct access to the ontology resources (concepts, collections, and concepts schemes). While the REST-ful interface provides easy and direct access to the ontology resources themselves, the semantic web service provides access to relatively complex operations over the semantic knowledge, such as hierarchy building, and keyword interpretation. In addition, it allows users to control the amount of information to be returned by the server depending on their needs (brief, summary, full or extended information about resources).

1.6 Terminology

This document uses several key terms that are defined as follows.

Semantic Framework
The key concept in this report is “semantic framework”. In reality, there is no formal or common definition for such a concept.

The term “***semantic framework***” (SF) as used in this report means: a collection of classes, libraries, application programming interfaces (APIs), or applications that can be used to build semantics-aware information systems that integrate, manage, handle or deliver semantic knowledge related to the information system’s data and services.

Operation
Specification of a transformation or query that an object may be called to execute [Pe02]

Interface
A named set of operations that characterise the behaviour of an entity [Pe02]

Service
A distinct part of the functionality that is provided by an entity through interfaces [Pe02]

Service Instance
An actual implementation of a service; service instance is often interchangeable with server

Client
A software component that can invoke an operation from a server

[11] http://gcmd.nasa.gov/

Request
An invocation by a client of an operation

Response
The result of an operation returned from a server to a client

Capabilities XML
Service-level metadata describing the operations and content available at a service instance

Recommendations
For recommendations, the following terms are used.

Rule
Rules SHALL be followed to ensure compatibility and/or conformance with standards, directives or the project objectives. A rule is characterised by the use of the words SHALL and SHALL NOT.

Recommendation
Recommendations consist of advice to implementers that will affect the usability of the final module (here the NETMAR semantic framework). A recommendation is characterised by the use of the words SHOULD and SHOULD NOT.

Permission
Permissions clarify areas of the specification that are not specifically prohibited. Permissions reassure the reader that a certain approach is acceptable and will cause no problem. Permissions are characterised by the use of the word MAY.

SHALL
"SHALL" is a keyword indicating a mandatory requirement. Designers SHALL implement such mandatory requirements in order to ensure conformance with the project objectives. This word is usually associated with a rule.

SHOULD
"SHOULD" is a keyword indicating flexibility of choice with a strongly preferred implementation. This word is usually associated with a recommendation.

MAY
"MAY" is a keyword indicating flexibility of choice with no implied preference. This word is usually associated with permissions.

1.7 Organisation of this Document

This document is organised as follows. Chapter 2 defines common use cases for vocabularies and semantic knowledge. Next, Chapter 3 defines the semantic data model used by the semantic web service and introduces its terminology. Chapter 4 provides an overview of the semantic framework.

Chapter 5 focuses on the basic service requirements, whereas Chapter 6 focuses on the functional requirements and defines the request and response formats for each of the semantic web service operations.

2 Vocabularies and Semantics Use Cases

The NETMAR project defined a set of practical and generic use cases (applications) for ontologies that aim to improve the pertinence of an SDI. Each use case typically requires a set of recurrent semantic operations. This specification tries to cover the common operations and capabilities required by the identified use cases. However, more use case may be defined, which may require additional operations.

2.1 Use Case 1 (UC-1): Ontology Browsing

Ontology browsing is the ability to graphically navigate an ontology or a thesaurus in order to understand the meaning of the concepts (ideas represented by terms) contained therein and to find out how these relate to each other (related, narrower or broader concepts, etc.). Ontology browsing is useful in web atlases as a way of providing educational information about a given domain (domain knowledge). It is also commonly used in product discovery interfaces (c.f., UC-2, section 2.2) as a way to find products by topic (e.g., multi-faceted product browsing). Existing semantic web services usually provide graphical ontology browsers.

Typically, an ontology browser needs to find out:

1.1. What thesauri are delivered by a given semantic service;
1.2. Describe a specified concept scheme;
1.3. What terms or collections of terms are contained within a given thesaurus or term collection;
1.4. What terms are related (same as, narrower, broader, etc.) to a given term;
1.5. What thesauri, term collections, or terms match a given free text keyword.

Additionally, an ontology browser may request the semantic service to:

1.6. Build a graph structure, most commonly hierarchy, of the terms of a given thesaurus or term collection.

2.2 Use Case 2 (UC-2): Product Discovery

Ontologies may be used by product (e.g., data, services, etc.) discovery services (e.g., catalogue services) as a means to improve the pertinence of their search results, by exploiting the semantic relationships between terms (narrower, related, same as, etc.), and/or trying to interpret the meaning of a user free text keyword according to a given thesaurus. For instance, if you search for datasets matching the term “seabed”, you would be able to get those tagged with the keyword “seafloor” (synonym), or if you search for “CTD” (i.e., Conductivity, Temperature, Depth), you would be able to get “Sea Surface Salinity” datasets.

Typically, a semantically enabled product discovery service needs to find out:

2.1. What terms are related to a given term (same as, narrower, related, etc.);
2.2. What terms match a given free text keyword.

2.3 Use Case 3 (UC-3): Interoperability

Database and service interoperability is another common use case of ontologies and semantic knowledge. Typically, in this area, ontologies are used as a mapping mechanism between

A. Two data structures/schemas (structural interoperability);
B. The values of similar properties (attributes) in different databases, using different representations (semantic interoperability).

The former occurs when two information systems, with structurally heterogeneous backends, need to interoperate with each other or with a third party system, e.g., mediator, broker, extract

transform and load (ETL) tool, etc. The latter supposes that structural interoperability has already been achieved and that actual data values need to be mapped or translated from one model, classification scheme, or terminology, to another. This is the typical case of distributed catalogue services using different vocabularies, possibly from different domains or in different natural languages, for product metadata values, e.g., descriptive keywords, units of measure, parameter names, organisation names, etc.

Structural and semantic interoperability, in the general case, may require complex ontology models and mappings, able to capture database primary and foreign keys, or how to convert attribute values from one model into another (e.g., degrees Celsius to Fahrenheit, etc.). However, in several practical cases (e.g., catalogue service interoperability), a simple semantic model relying on basic semantic relationships, such as broader, narrower, and related, may be sufficient to perfectly interoperate two semantically or structurally heterogeneous systems. In such a case, the interoperable agent (ETL, mediator, etc.) typically requires information such as:

3.1. What terms are related (same as, narrower, broader, etc.) to a given term; this is useful to translate one attribute value from one model into another for example;
3.2. Find all concepts from a given concept scheme or collections that semantically match (equivalent to or narrower than) a given keyword or concept scheme;
3.3. What terms best match a keyword; this is useful for an ETL or a schema matcher to find out which table or attribute from a destination database schema matches a table or attribute from a data source schema.

2.4 Use Case 4 (UC-4): Service Chaining

A service chain is defined in ISO 19119 as “a sequence of services where, for each adjacent pair of services, occurrence of the first action is necessary for the occurrence of the second action” [ISO05]. From a user perspective, service chaining is the linking together of standardised data and processing services into a workflow to produce results that are not predefined by the service providers. The defined workflows will then be passed to a workflow engine for execution.

Typically, a processing service requires input data with given structure and semantics and outputs data also with given structure and semantics. While chaining services, one must make sure that the data output by a service and fed into another conform to the required structure and semantics of the latter.

It is possible to use semantic knowledge to ensure that the inputs and outputs of each component of a service chain are “semantically compatible.” This can be achieved by using ontologies as a way to represent the semantics of data and service input and output parameters. The service-chaining engine needs then to check whether an input dataset parameter is “semantically compatible” with a processing service input parameter (in terms of the parameter type, dimension, unit of measure, etc.). In this way, correct connections between components can be enforced (e.g., do not send chlorophyll data to a component that only knows how to process sea surface temperature).

Checking the semantic compatibility of two parameters in most practical cases is a matter of checking whether two concepts are the same, or whether a concept is narrower/broader than the other, or whether both concepts share a common broader concept. Therefore, it is commonly possible to express parameter compatibility as a combination of one or more of the following operations:

4.1. What concepts (parameters, dimensions, units, etc.) are narrower or broader than a given concept (parameter, dimension, unit, etc.);
4.2. Is a given concept (parameter, dimension, unit, etc.) narrower than another given concept (parameter, dimension, unit, etc.)?

2.5 Summary

Based on the above functional requirements identified above, we have identified a list of generic operations to be supported by the semantic web service as specified in the table below.

Functional requirements	Operations matching the functional requirements
1.1. What thesauri are delivered by a given semantic service	Get concept schemes (GetConceptSchemes)
1.2. Describe a specified concept scheme	Get concept scheme (GetConceptScheme)
1.3. What terms or collections of terms are contained within a given thesaurus or concept scheme	Get concepts (GetConcepts), with possible filtering by target concept scheme or collection Get collections (GetCollections) with possible filtering by target concept scheme
1.4, 2.1, 3.1. What terms are related (same as, narrower, broader, etc.) to a given term 4.1. What concepts are narrower or broader than a given concept	Get related concepts (GetRelatedConcepts), relationship to be specified as part of the request, e.g., narrower, broader, related, narrower transitive, etc.
1.5. What thesauri, term collections, or terms match a given free text keyword 2.2. What terms match a given free text keyword 3.3. What terms best match a keyword	Search concept scheme (SearchConceptScheme) Search collection (SearchCollection) Search concept (SearchConcept)
1.6. Build a hierarchy of the terms of a given thesaurus or term collection	Get concept hierarchy (GetConceptHierarchy)
1.6. Build a hierarchy of the terms of a given thesaurus or term collection	Get explicitly asserted or implicit top level concepts of a given concept scheme, i.e., root nodes of the hierarchy (GetExplicitTopConcepts, GetImplicitTopConcepts)
3.2. Find all concepts from a given concept scheme or collections that semantically match (equivalent to or narrower than) a given keyword	Interpret keyword (InterpretKeyword)
4.2. Is a given concept narrower than another given concept	Check relation (CheckRelation), source and target concepts, and relationship to be specified as part of the request
Other useful general-purpose operations: Get service metadata, get information about a specified collection or concept, get the content of a given collection	Get capabilities (GetCapabilities) Get collection (GetCollection) Get concept (GetConcept) Get collection content (GetCollectionContent)

3 Semantic Data Model

The semantic web service specified in this document is primarily intended to respond to most common/recurring queries over SKOS-based ontologies.

In this chapter, we introduce basic SKOS constructs. As SKOS is available in OWL (Web Ontology Language [DS04]), we firstly define a few useful OWL constructs, such as classes and properties.

This chapter does not intend to provide a full SKOS documentation. Rather it defines basic notions required to understand the remaining of the document. For further details about SKOS, readers are referred to the SKOS reference [MB09].

3.1 Terminology

3.1.1 *Resources*

A resource is an abstract term that refers to any of the notions defined in the subsequent subsections. We refer to resources using the notation <namespace>:<resourceName>, where <namespace> refers to the namespace of the vocabulary or XML schema in which the resource is defined, and <resourceName> refers to the local name of the resource. For instance, owl:Class refers to the construct "Class" defined in the OWL language.

In this chapter, in addition to SKOS and OWL, we use the following vocabularies:

- RDF, the Resource Description Framework [MM04], a language for representing information about resources in the World Wide Web;
- RDF-S, i.e., RDF Schema [BG04], which extends RDF by adding more modelling primitives such as classes, inheritance, domains, etc.

Below is the list of namespaces used in this chapter.

```
rdf:  http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs: http://www.w3.org/2000/01/rdf-schema#
owl:  http://www.w3.org/2002/07/owl#
skos: http://www.w3.org/2004/02/skos/core#
```

3.1.2 *Classes & Instances (Individuals)*

The term "*class*", as used in this document, is interchangeable with "*OWL class*" (owl:Class) [DS04]. In OWL, classes provide an abstraction mechanism for grouping resources that share similar characteristics. For instance, Instrument or Parameter can be regarded as classes. A resource belonging to a given class is called *instance* of that class or also *individual*. For example, Thermometer is an instance of Instrument.

A class may be a *subclass* of another, in which case the latter would be called *super-class* of the former. Inheritance relationships (sub-class and super-class) are transitive. In OWL, all classes are direct or indirect sub-classes of the OWL Thing class (owl:Thing).

3.1.3 *Properties*

RDF properties (rdf:Property) provide a mechanism for creating relationships amongst resources or between resources and data values. OWL refines RDF properties into four types of properties:

- Object properties (owl:ObjectProperty), which link individuals to individuals; for instance an Instrument ***measures*** a Parameter; therefore *measures* may be considered as an object property;
- Data type properties (owl:DatatypeProperty), which link individuals to data values (also called atomic values or literals); for instance: a Place may have a *northernmost latitude* which can be defined as a data type property;
- Annotation properties (owl:annotationProperty), which link resources (classes, individuals, properties, etc.) to data values or resources; for instance the RDFS label property (skos:label) and the SKOS preferred label property (skos:prefLabel) are annotation properties;
- Ontology properties (owl:OntologyProperty), which link ontologies to resources or data values; for instance owl:backwardCompatibleWith (backward compatible with) is an ontology property.

According to RDFS, a property may be a sub-property of another (see examples below from SKOS). OWL provides constructs for specifying mathematical properties of properties, such as transitivity, symmetry, inverse property, etc.

3.1.4 *SKOS Concept Schemes*

In SKOS, vocabularies MAY be organised into concept schemes (skos:ConceptScheme). A concept scheme can be regarded as a knowledge organisation system, a thesaurus or a classification scheme.

In SKOS, concept schemes SHALL be implemented as instances of the SKOS *ConceptScheme* class (skos:ConceptScheme). Below is an example of a concept scheme definition.

```
<skos:ConceptScheme rdf:ID="http://vocab.ndg.nerc.ac.uk/list/P081/3">
  <skos:definition xml:lang="en">
    Terms used to classify SeaDataNet Agreed Parameter Groups to provide
    topic/theme level terms in a hierarchical parameter discovery interface
  </skos:definition>
  <skos:prefLabel xml:lang="en">
    SeaDataNet Parameter Disciplines
  </skos:prefLabel>
  <!--...-->
</skos:ConceptScheme>
```

3.1.5 *SKOS Concepts*

A concept is defined in the SKOS reference [MB09] as “an idea or notion; a unit of thought.” SKOS concepts are instances of the SKOS Concept class (skos:Concept). As we are only supporting OWL DL (and consequently OWL Lite) [MH04], concepts SHALL NOT be classes. Nevertheless they MAY be organised internally into OWL classes. For instance, one MAY define a class *Place* for containing place concepts, and a sub-class of that named *ICESDivision for ICES divisions*.

Concepts SHALL be linked to concept schemes using the skos:inScheme object property as shown in the example below.

```
<skos:Concept rdf:ID="http://vocab.ndg.nerc.ac.uk/list/P081/3/DS10">
  <skos:inScheme rdf:resource="http://vocab.ndg.nerc.ac.uk/list/P081/3"/>
  <skos:prefLabel xml:lang="en">Environment</skos:prefLabel>
  <skos:definition xml:lang="en">
    The domain documenting the activities of man that have an effect on the
    Earth System
  </skos:definition>
</skos:Concept>
```


IF a concept is a top-level concept of a given concept scheme then it SHOULD be defined as such using the skos:topConceptOf object property (the inverse of which is skos:hasTopConcept). A concept is a top concept of a concept scheme if it has no broader concept within the concept scheme in question[12].

```
<skos:ConceptScheme rdf:ID="http://vocab.ndg.nerc.ac.uk/list/P081/3">
  <!—Concept scheme definition...-->
  <skos:hasTopConcept>
    <skos:Concept rdf:ID="http://vocab.ndg.nerc.ac.uk/list/P081/3/DS10">
      <!--...-->
    </skos:Concept>
  </skos:hasTopConcept>
</skos:ConceptScheme>
```

Please note that skos:topConceptOf is a sub-property of skos:inScheme (c.f. Figure 2.2). Therefore, if a concept is defined as a top concept of a given concept scheme then there is no need to define it as belonging to it.

```
skos:inScheme
 |
 +- skos:topConceptOf <--> (inverse of skos:hasTopConcept)
```

Figure 3.1. Concept-Concept Scheme Relationships

3.1.6 *SKOS Collections*

Another useful notion in SKOS is that of collections (skos:Collection). SKOS collections are groups of concepts that share something in common and which can be labelled (skos:Collection) or ordered (skos:OrderedCollection); c.f., Figure 2.7.

```
skos:Collection [A meaningful collection of concepts]
 |
 +- skos:OrderedCollection [An ordered collection of concepts, where both the grouping and the
                            ordering are meaningful]
```

Figure 3.2. SKOS Collections

SKOS collections are useful in general as a way to define simple classification schemes within a thesaurus (concept scheme) or as a way to order concepts. For instance, concepts representing remote sensing instruments may be grouped into one collection. A collection may contain concepts and/or other collections.

Collection memberships are expressed using the skos:member and skos:memberList properties. The former is used to ascertain that a collection contains a concept or another collection. The latter is used to list the elements of an ordered collection in an rdf:list [MM04].

Below is an example that defines two collections: *RemoteSensingInstruments* and *ActiveRemoteSensingInstruments*. The latter contains two concepts, MultibeamEchosounder and

[12] Please note that, as per the SKOS reference, this is not a strict rule in SKOS.

SingleBeamEchosounder, and is a member of the former. For the sake of readability, we do not use URIs as identifiers in the subsequent examples.

```
<skos:Collection rdf:ID="RemoteSensingInstruments">
  <skos:prefLabel xml:lang=”en”>Remote Sensing Instruments</skos:prefLabel>
  <!--Collection definition here: labels, definition, etc.-->
  <skos:member>
    <skos:Collection rdf:ID="ActiveRemoteSensingInstruments">
      <!--Collection definition here: labels, definition, etc.-->
      <skos:member>
        <skos:Concept rdf:ID="MultieamEchosounder">
          <!--Concept definition here: labels, definition, etc.-->
        </skos:Concept>
      </skos:member>
      <skos:member>
        <skos:Concept rdf:ID="SingleBeamEchosounder">
          <!--Concept definition here: labels, definition, etc.-->
        </skos:Concept>
      </skos:member>
    </skos:Collection>
  </skos:member>
</skos:Collection>
```

The example below defines an ordered collection, called MarineStrata, which consists of four concepts: SubSeabed, Seabed, WaterColumn, and WaterSurface, ordered vertically from the lowest to the highest.

```
<skos:OrderedCollection rdf:ID="MarineStrata">
  <skos:prefLabel xml:lang=”en”>Marine Strata</skos:prefLabel>
  <!--Collection definition here: labels, definition, etc.-->
  <skos:memberList>
    <rdf:List>
      <rdf:first>
        <skos:Concept rdf:ID="SubSeabed"/>
      </rdf:first>
      <rdf:rest>
        <rdf:List>
          <rdf:first>
            <skos:Concept rdf:ID="Seabed"/>
          </rdf:first>
          <rdf:rest>
            <rdf:List>
              <rdf:first>
                <skos:Concept rdf:ID="WaterColumn"/>
              </rdf:first>
              <rdf:rest>
                <rdf:List>
                  <rdf:first>
                    <skos:Concept rdf:ID="WaterColumn"/>
                  </rdf:first>
                </rdf:List>
              </rdf:rest>
            </rdf:List>
          </rdf:rest>
        </rdf:List>
      </rdf:rest>
    </df:List>
  </skos:memberList>
</skos:OrderedCollection>
```


3.1.7 *SKOS Annotations*

SKOS defines a set of annotation properties for annotating resources (classes, properties, concepts, etc.). SKOS annotation properties are divided into two groups: lexical labels and documentation properties (or note properties).

3.1.7.1 Lexical Labels

Lexical labels, the values of which are UNICODE characters in a given natural language, are used to associate various types of labels with concepts or resources in general, such as preferred label or alternative labels, etc.

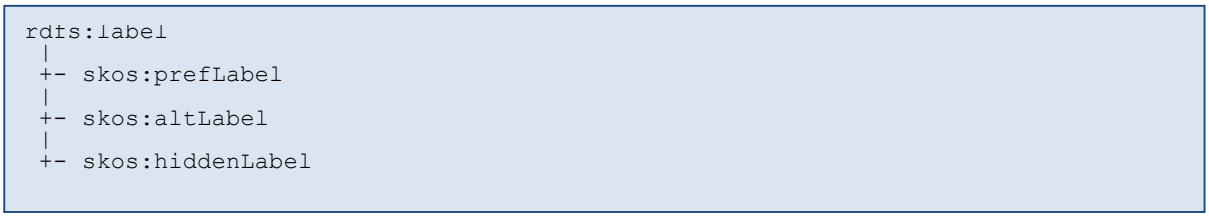

Figure 3.3. SKOS Lexical Labels

As shown in Figure 2.4, all SKOS lexical labels are sub-properties of the RDFS label annotation property (rdfs:label). Preferred and alternative labels (skos:prefLabel and skos:altLabel) are human-readable representations of concepts or resources. A hidden label MAY be a popular misspelled label, which may be useful for improving vocabulary search. For instance, one may define “Tide Guage” (which is a very common misspelling of “Tide Gauge”) as a hidden label for the concept “Tide Gauge.” This would help users find the term “tide gauge” event when they misspell it as “tide guage”.

Below is an example of a valid concept definition using the three SKOS lexical labels (preferred, alternative and hidden labels).

```
<skos:Concept rdf:ID=”TideGauge”>
  <skos:prefLabel xml:lang=”en”>Tide Gauge</skos:prefLabel>
  <skos:altLabel xml:lang=”en”>Tide Gage</skos:altLabel>
  <skos:hiddenLabel xml:lang=”en”>Tide Guage</skos:hiddenLabel>
  <!--...-->
</skos:Concept>
```

Preferred, alternative and hidden labels in a given language SHALL be all mutually exclusive. Therefore, the same term SHALL NOT be used simultaneously as alternative label and preferred label (or as preferred label and hidden label, or as alternative label and hidden label) within one given language.

Every concept SHOULD at least have a label (preferred label) per language, even if the label is the same as the concept ID (c.f. example above). And, as per the SKOS specification [MB09], a resource SHALL NOT have more than one preferred label (skos:prefLabel value) per language.

Finally, as all SKOS lexical labels are sub-properties of the RDFS label, ontology administrators SHOULD avoid defining RDFS labels for the ontology resources. Rather they SHALL use the SKOS lexical labels, which are more specific. RDFS labels can be inferred from SKOS labels.

3.1.7.2 Documentation Properties

SKOS defines a set of documentation properties (also called note properties) designed to provide information about SKOS concepts. The top level SKOS documentation property is skos:note which

may be used directly, or as a super-property for more specific note types. The SKOS documentation properties are illustrated in Figure 2.5. The main of them is skos:definition which provides a statement or formal explanation of the meaning of a concept. Definitions are very useful as they explain to users what the concepts within a given ontology actually mean, thus allowing for the disambiguation of terms. Therefore, every concept SHALL have at least one definition per supported language.

Please note that there is no restriction on the nature of the information provided by SKOS documentation properties. This can be plain text, hypertext, image, or even an object (individual). Nevertheless, in this version of the semantic framework, we only support plain text literals.

```
skos:note [A general note, for any purpose]
 |
 +- skos:changeNote [A note about a modification to a concept]
 |
 +- skos:definition [A statement or formal explanation of the meaning of a concept]
 |
 +- skos:editorialNote [A note for an editor, translator or maintainer of the vocabulary]
 |
 +- skos:example [An example of the use of a concept]
 |
 +- skos:historyNote [A note about the past state/use/meaning of a concept]
 |
 +- skos:scopeNote [A note that helps to clarify the meaning and/or the use of a concept]
```

Figure 3.4. SKOS Documentation Properties

3.1.8 *SKOS Semantic Relations*

SKOS provides a set of object properties, known as semantic relations, useful to link concepts **within a concept scheme**. All SKOS semantic relations are sub-properties of skos:semanticRelation. Figure 2.6 shows the SKOS semantic relations, which are represented in black (red ones correspond to mapping properties, c.f., subsection 2.1.8).

```
skos:semanticRelation [Links a concept to a concept related by meaning]
 |
 +- skos:broaderTransitive [Transitive super-property of skos:broader]
 |   |
 |   +- skos:broader [Relates a concept to a concept that is more general in meaning]
 |       |
 |       +- skos:broadMatch [See below]
 |
 +- skos:narrowerTransitive [Transitive super-property of skos:narrower]
 |   |
 |   +- skos:narrower [Relates a concept to a concept that is more specific in meaning]
 |       |
 |       +- skos:narrowMatch [See below]
 |
 +- skos:related [A statement or formal explanation of the meaning of a concept]
 |   |
 |   +- skos:relatedMatch [See below]
 |
 +- skos:mappingRelation [Relates two concepts coming, by convention, from different schemes]
     |
     +- skos:broadMatch [Hierarchical mapping between two concepts in different schemes]
     |
     +- skos:narrowMatch [Hierarchical mapping between two concepts in different schemes]
     |
     +- skos:relatedMatch [Associative mapping between two concepts in different schemes]
     |
     +- skos:closeMatch [Links two concepts that are similar enough to be used interchangeably]
         |
         +- skos:exactMatch [Links two concepts with a high degree of similarity]
```

Figure 2.6. SKOS Semantic Relations

By convention, those semantic relations written in italic SHALL NOT be used directly to make assertions. Rather they SHOULD be used to draw inferences about the hierarchical structure of relationships or their transitivity.

For instance, assume you have two concepts *Geology* and *MarineGeology*, the former being broader then the latter. Then, you SHOULD NOT make the assertion

MarineGeology skos:broaderTransitive Geology,

or the assertion

MarineGeology skos:semanticRelation Geology.

Rather you SHOULD make the following assertion (from which the above assertions can be inferred if requested by a user)

MarineGeology skos:broader Geology.

3.1.9 *SKOS Mapping Properties*

SKOS provides a set of mapping properties intended for linking concepts **from different concept schemes**. The SKOS mapping properties are all sub-properties of skos:mappingRelation. SKOS mapping properties are represented in red in Figure 2.6.

When mappings concepts from different schemes one SHOULD use SKOS mapping properties rather than SKOS semantic relations.

3.2 Multilinguality

As per the best practices and guidelines for multilinguality defined by the European Environment Information and Observation Network (EIONET)[13]:

1. Language codes SHALL follow the ISO 639-1 international standard;
2. In order to be able to display Latin, Greek, Armenian, Georgian and Cyrillic on the same page, we SHALL use a character set that contains all these alphabets. Therefore, we SHALL use UTF-8 as the character code for content transmitted to the web browser (or semantic framework clients in general).

In RDF and OWL, labels and definitions of resources SHALL declare their languages using the xml:lang attribute. An example of a SKOS multilingual concept is shown below.

```
<?xml version="1.0" encoding="utf-8"?>
...
<skos:Concept rdf:ID=”Geology”>
  <skos:prefLabel xml:lang=”en”>Geology</skos:prefLabel>
  <skos:prefLabel xml:lang=”fr”>Géologie</skos:prefLabel>
  <skos:definition xml:lang=”en”>The scientific study of the origin, history,
structure, and composition of the earth</skos:definition>
  <skos:definition xml:lang=”fr”>Domaine scientifique qui étudie l'origine de la
Terre, son histoire, sa forme, les matériaux qui la composent et les processus
qui ont agi sur elle ou qui agissent encore</skos:definition>
  ...
</skos:Concept>
```

[13] http://www.eionet.europa.eu/

As per the SKOS specification [MB09], a resource SHALL NOT have more than one preferred label (skos:prefLabel value) per language tag.

3.3 General OWL and SKOS Rules and Recommendations

When working with SKOS, one must bear in mind that concepts, collections, and concept schemes SHALL be mutually disjoint. Therefore, a concept cannot be a concept scheme or a collection, and a collection cannot be a concept scheme. In the same way, you SHALL NOT use SKOS semantic relations or mapping properties to link a collection to a concept (or vice versa) or a concept scheme to a collection or a concept, and so on. In fact, SKOS semantic relations and mapping properties SHALL only link concepts to concepts.

More rules and recommendations related to SKOS can be found in the SKOS reference [MB09].

The semantic knowledge internal to the semantic web service MAY benefit from the rich capabilities and expressiveness of OWL. For instance it MAY organise concepts internally in a hierarchy of OWL classes and MAY use inference rules, class restrictions, OWL and OWL 2 constructs, etc. However this SHOULD be transparent to the SWS users. Data delivered by the SWS SHOULD be SKOS-compliant.

3.4 Semantic Resources Identification

Semantic resources SHOULD be identified using their URIs.

It is very common that the URI of a semantic resource within a concept scheme or ontology is of the form: <namespace>#<resource local name>, where <namespace> is an ontology or concept scheme URI. However, this is not a general rule. For instance, the NERC Vocabulary Server (NVS)[14] uses URIs of the form:

- http://vocab.ndg.nerc.ac.uk/list/<listReference>/<listVersion> for lists (i.e., concept schemes),
- http://vocab.ndg.nerc.ac.uk/term/<listReference>/<listVersion>/<entryReference> for entry terms (i.e., concepts).

[14] http://www.bodc.ac.uk/products/web_services/vocab/

4 Semantic Framework - Overview

The typical architecture of the proposed semantic framework is illustrated in Figure 4.1. As shown in this diagram, the SWS uses semantic knowledge that may be stored and managed in a variety of ways, e.g., triple store, database, ontology files (RDF, OWL, etc.), external web service, etc.

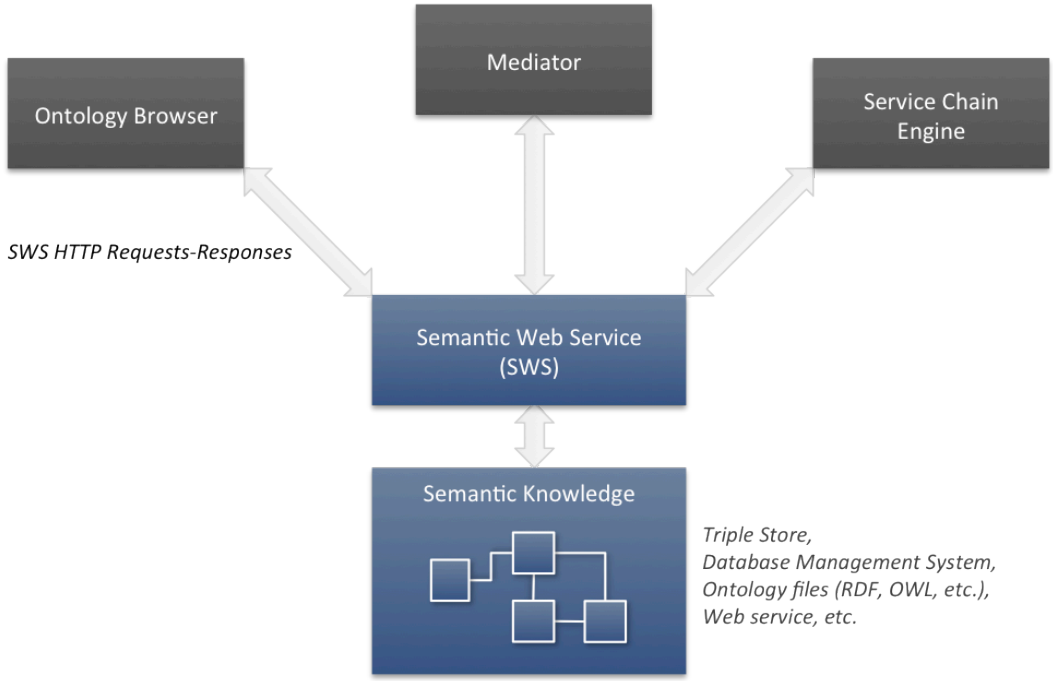

Figure 4.1. Typical Semantic Framework Architecture

The SWS is made accessible on the web to "semantically-enabled" applications, such as ontology browsers, semantic mediators, or service chain engines. These interact with the SWS using SWS requests and responses over HTTP.

4.1 SWS Operations

The SWS aims to provide a web interface for querying SKOS thesauri through a standardised set of high-level and easy-to-use operations required by most common semantically-enabled clients such as data and metadata mediators, service chaining engines, vocabulary browsers and data and service discovery interfaces. The SWS supports the following operations.

1. ***GetCapabilities***
   Retrieves service metadata, including supported operations, response formats, available concept schemes, and their supported languages.

2. ***GetConceptSchemes***
   Lists available concept schemes with their annotations (labels, definitions, etc.).

3. ***GetConceptScheme***
   Returns a concept scheme definition given its URI. The response includes the concept scheme's annotations.

4. ***SearchConceptScheme***
   Returns the definition(s) of one or more concept scheme(s) matching a specified free-text keyword.

5. ***GetConceptSchemeContent***
   Returns the content of a given concept scheme (identified by its URI), including its collections and concepts.

6. ***GetCollections***
   Lists available concept collections with their annotations. Collections may be filtered by one or more concept schemes.

7. ***GetCollection***
   Returns a collection definition identified by its URI. The response includes the collection's annotations.

8. ***SearchCollection***
   Returns the definition(s) of one or more collection(s) matching a specified free-text keyword.

9. ***GetCollectionContent***
   Returns the content of a given collection (identified by its URI), including member collections and concepts.

10. ***GetConcepts***
    Returns the definitions of the concepts belonging to a specified concept scheme and/or collection.

11. ***GetConcept***
    Returns a concept definition given its URI. The response includes the concept's annotations.

12. ***SearchConcept***
    A search operation that returns the concepts that textually match a given keyword.

13. ***GetRelatedConcepts***
    Returns the concepts related to one or many given concept(s) using one or many given SKOS relationship(s) (e.g., skos:narrower, skos:broader, skos:related, etc.), both from direct assertions and by entailment.

14. ***GetExplicitTopConcepts***
    Returns the concepts that have explicitly been asserted as top concepts of a specified concept scheme.

15. ***GetImplicitTopConcepts***
    Returns the top-level concepts of a specified concept scheme.

16. ***GetConceptHierarchy***
    This operation is suitable for small thesauri, and is useful for ontology browsers. It returns the hierarchy of the concepts within a given concept scheme and/or collection.

17. ***InterpretKeyword***

Returns the concepts that semantically match a given keyword, within a specified concept scheme and or collection.

18. ***CheckRelation***
    Checks whether two specified concepts are related via a specified SKOS relationship.

4.2 SWS Client-Service Interaction

The interaction between an SWS client and a server relies on a series of SWS operation calls and responses. In this section we provide two examples of SWS client-service interactions: vocabulary (or ontology) browsing, and catalogue service mediation.

4.2.1 *Vocabulary Browsing (UC-1)*

The protocol diagram of Figure 4.2 outlines the typical protocol to be followed in order to process semantic web service requests involved in vocabulary browsing.

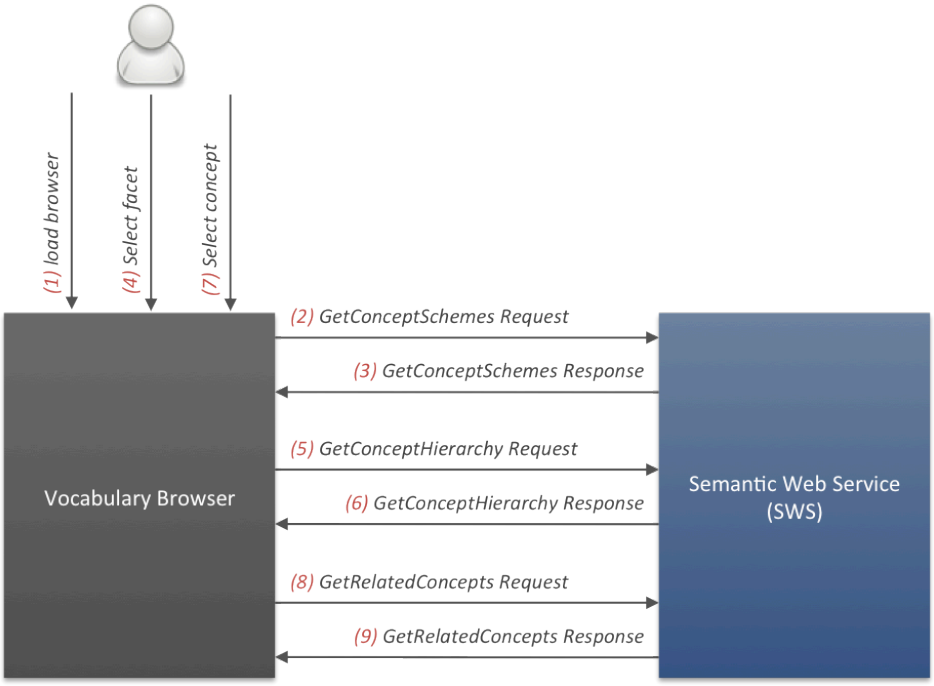

Figure 4.2. Protocol Diagram for an SWS Client-Service Interaction involved in Vocabulary Browsing

This protocol diagram above involves 3 actors:

- The user of the system,
- The vocabulary browser,
- The semantic web service.

The typical component interactions illustrated in this figure are explained below.

- (1) The user loads the vocabulary browser (VB).
- (2) While loading, the VB submits a *GetConceptSchemes* request to the SWS in order to get all information (definitions, labels, etc.) related to the thesauri (concept schemes).

- (3) The SWS returns the list of concept schemes together with their definitions and labels, which are used then displayed by the browser.
- (4) The user selects the concept scheme of interest.
- (5) The VB sends a *GetConceptHierarchy* request to the SWS with the aim to retrieve a hierarchy of the concepts contained within the concept scheme selected by the user.
- (6) The SWS returns the concept hierarchy for the concept scheme selected by the user and the VB displays it.
- (7) The user selects a concept of interest from the concept hierarchy.
- (8) In order to display the list of concepts related to the selected concept, the VB sends a *GetRelatedConcepts* request to the SWS.
- (9) The SWS responds back and the OB displays the related concepts

The interactions can continue from step 4 (user selects a different concept scheme, or selects a related concept from a different concept schemes) or step 7 (use selects a related concept from the current concept scheme).

4.2.2 *Catalogue Service Mediation (UC-2 and 3)*

The protocol diagram of Figure 4.3 outlines the typical protocol to be followed in order to process semantic web service requests involved in the mediation of OGC catalogue services (CSW) [NW05].

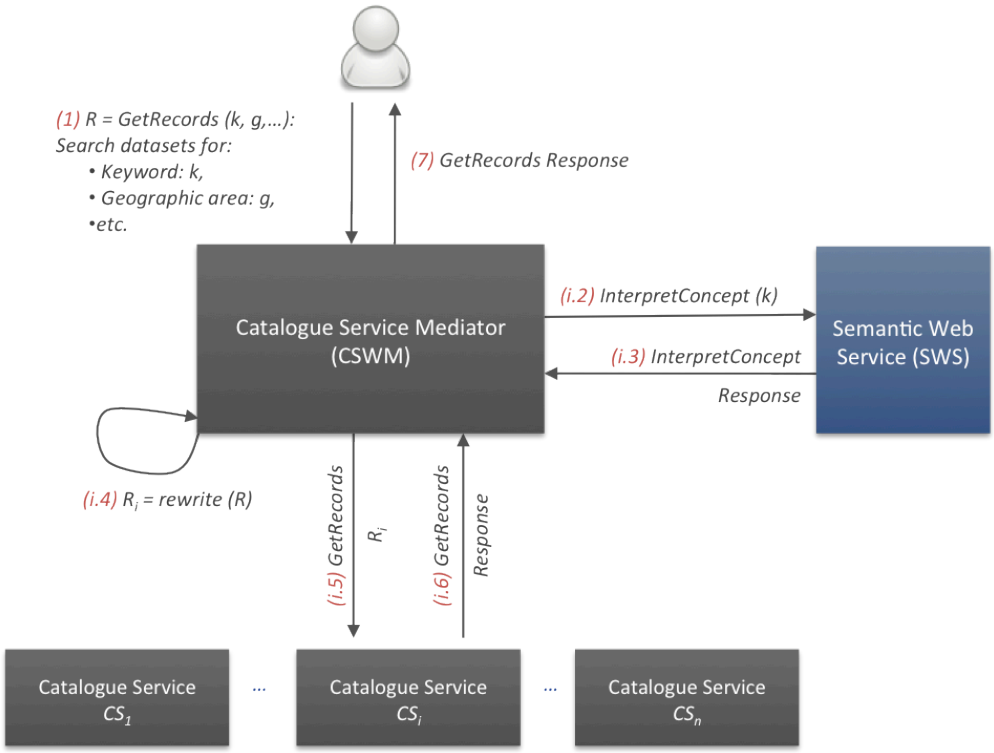

Figure 4.3. Protocol Diagram for an SWS Client-Service Interaction involved in Catalogue Service Mediation

This protocol diagram involves 4 actors:

- The user of the system,
- The catalogue service mediator (CSWM),
- The semantic web service,
- A set of distributed catalogue services (CSW).

The typical component interactions illustrated in this figure are explained below.

- (1) The user sends a GetRecords request to the catalogue service mediator to search datasets for a given keyword *k* and a geographic extent *g*.
- The CSWM parses the user query *R*, and for each catalogue service CS_i to be involved in the query (by default all the catalogue nodes are involved unless otherwise specified in the user request R) it does the following (in parallel):
  - (i.2) The CSWM extracts the keyword of interest *k* and submits an InterpretKeyword request to the SWS with *k* as the keyword parameter value.
  - (i.3) The SWS responds to the CSW by sending the list of concepts semantically covered by the user keyword *k* (i.e., best matches of the keyword and their narrower concepts).
  - (i.4) The CSWM, now, has all the concepts covered by the user's keyword. It rewrites the user's request into a request supported by catalogue node CS_i and translates the original keyword using the concepts obtained in step (i.4). Let R_i denote the so-obtained query.
  - (i.5) The CSWM submits the rewritten query R_i to catalogue node CS_i.
  - (i.6) Catalogue node CS_i returns a response for the CSWM's request R_i.
- (7) The CSWM mediator collects all the catalogue node responses, wrap them in a CSWM GetRecords request and sends them back to the user.

4.2.3 *Service Chaining (UC-4)*

The protocol diagram of Figure 4.4 outlines the typical protocol to be followed in order to check the semantic validity of a service chain by a service chain editor or engine.

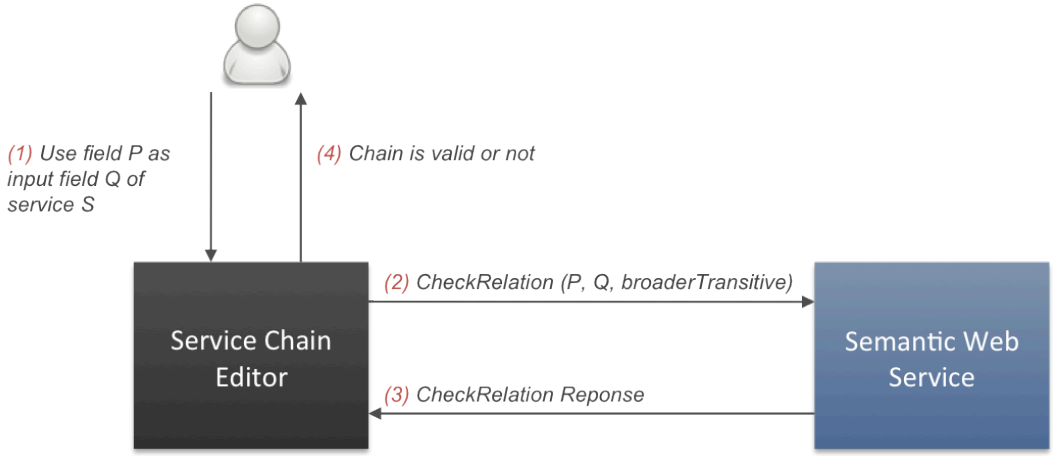

Figure 4.4. Protocol Diagram for an SWS Client-Service Interaction involved in Service Chaining

This service chaining protocol diagram involves 3 actors:

- The user of the system,
- The service chain editor,
- The semantic web service.

The typical component interactions illustrated in this figure are explained below.

- (1) The user sends requests that the service chain editor use a given dataset field representing a parameter *P* as input for a processing service *S*, the input representing a given parameter Q.

- (2) The service chain editor sends a CheckRelation request to the SWS to check whether parameter P (represented by the input field) is compatible with parameter Q (requested by the service), in other terms to check whether P is a narrower term than Q. For instance, “Sea Surface Temperature” is narrower than “Temperature”, therefore a dataset field tagged as “Sea Surface Temperature” can be accepted as input to a service requiring “Temperature” as an input.
- (3) The SWS sends back a positive or negative response to the service chain editor.
- (4) Depending on the SWS response the service chain editor decides whether the service chain is valid or not and warns the user accordingly.

5 Basic Service Requirements

This chapter specifies basic service requirements for the semantic framework, version 2.0.

5.1 Version Numbering

This section specifies the way the semantic web service versioning should be dealt with. Please note that versioning here is relative to the SWS only and not to the vocabularies within. A versioning mechanism for the SWS will be very useful, as the SWS specification will evolve.

5.1.1 *Version Number Format*

The version number format for the semantic web service contains two positive integers, separated by a decimal point, in the form “x.y”, e.g., “2.0”.

5.1.2 *Version Changes*

The semantic resource server software version number shall be changed with each revision. The number shall increase monotonically and shall comprise no more than two integers separated by decimal points, with the first integer being the most significant. There may be gaps in the numerical sequence. Some numbers may denote experimental or interim versions. Service instances and their clients need not support all defined versions, but must obey the negotiation rules below.

5.1.3 *Version Number Negotiation*

Version number negotiation SHALL occur as follows:

1. If the server implements the requested version number, the server must send that version.
2. If the client request is for an unknown version greater than the lowest version that the server understands, the server SHALL send the highest version less than the requested version.
3. If the client request is for a version lower than any of those known to the server, then the server SHALL send the lowest version it knows.
4. If the client does not understand the new version number sent by the server, it may either cease communicating with the server or send a new request with a new version number that the client does understand, but which is less than that sent by the server (if the server had responded with a lower version).
5. If the server had responded with a higher version (because the request was for a version lower than any known to the server), and the client does not understand the proposed higher version, then the client may send a new request with a version number higher than that sent by the server.

5.2 General HTTP Request Rules

5.2.1 *HTTP GET*

The service SHALL respond to HTTP GET KVP requests. These MAY come from client applications or web browsers.

5.2.2 *HTTP POST*

The service SHALL accept requests encoded either as XML requests compliant with the XML schemas defined herein, or SOAP messages.

5.2.3 *HTTPS*

HTTPS is not required for the basic semantic resources; however any editing functionality for the semantic resource SHALL be secured using HTTPS (see section 6.2 below).

5.3 Request Encoding

The service SHALL accept requests encoded either as XML requests or SOAP requests.

This document defines two methods of encoding HTTP requests. The first uses XML as the encoding language, and is intended for use with the HTTP POST method. The second encoding uses keyword-value pairs (KVP) to encode the various parameters of a request and is intended for use with HTTP GET.

5.4 Response Encoding

This document mandates the use of XML as an encoding for the semantic web service responses. In addition, semantic resources described within the responses SHALL be encoded in RDF/XML [BM04] and use the SKOS vocabulary where appropriate. The use of other response formats (e.g., JSON) or ontology languages (e.g., TURTLE) is optional.

5.5 Namespaces

Standard namespaces SHALL be used where appropriate, i.e. SKOS and RDFS for the delivery of semantic resources.

Below is the list of namespaces to be considered and which are used in this document.

```
xml:        http://www.w3.org/XML/1998/namespace
xs:         http://www.w3.org/2001/XMLSchema#
dct:        http://purl.org/dc/terms/
rdf:        http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:       http://www.w3.org/2000/01/rdf-schema#
owl:        http://www.w3.org/2002/07/owl#
skos:       http://www.w3.org/2004/02/skos/core#
sparql:     sparql xmlns="http://www.w3.org/2005/sparql-results#
soap:       http://www.w3.org/2003/05/soap-envelope
sws:        http://cmrc.ucc.ie/sws/2.0
```

5.6 Simple Object Access Protocol (SOAP)

A SOAP interface to the SWS MAY be provided. The following XML fragment illustrates a skeleton SOAP message.


```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap=http://www.w3.org/2003/05/soap-envelope
  soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
  <soap:Header>
    <!--...-->
  </soap:Header>
  <soap:Body>
    <!--.....>
    <soap:Fault>
      <!--...-->
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

When a client interacts with the SWS using SOAP, it SHALL embed the SWS XML request in the <soap:Body> element in the request message.

The SWS SHALL then respond by generating a SOAP message where the response to the client's request is the content of the <soap:Body> element.

IF an exception is encountered while processing a SWS request encoded in a SOAP envelope, THEN the SWS SHALL generate a SOAP response message where the content of the <soap:Body> element is a <soap:Fault> element. Below is the skeleton XML of such a response.

```
<soap:Body>
  <soap:Fault>
    <soap:faultcode>soap:Server</soap:faultode>
    <soap:faultstring>A server exception was encountered.</soap:faultstring>
    <soap:faultactor><!--URL of SWS server--></soap:faultactor>
    <soap:detail>
      <sws:ExceptionReport>
        <!--...-->
      </sws:ExpetionReport>
    </soap:detail>
  </soap:Fault>
</soap:Body>
```

5.7 Exception Reporting

Upon encountering an exception, the service SHALL generate an XML document stating that an exception has occurred and providing an intelligible traceback report of the error for debugging purposes.

The format of the XML error responses SHALL validate against the XML schema for SWS exceptions available at http://netmar.ucc.ie/xsd/sws/2.0/exceptions.xsd.

An HTML documentation for the SWS exceptions XML schema is available at the following URL: http://netmar.ucc.ie/xsd/2.0/doc/exceptions.html.

Exceptions are reported using the <ExceptionReport> element, which SHALL contain one or more exceptions, expressed using the <Exception> element. Exception messages SHALL be contained within the <ExceptionMessage> element, within an <Exception> element.

Each type of exception SHALL have a unique code, specified using the code attribute of the <Exception> element. The exception codes for this version of SWS (2.0) are defined in table 3.1.

Table 3.1. SWS Exception Codes

Exception code	Description
InternalError	Internal error to the SWS
InvalidParameterValue	The parameter value is not valid
InvalidRequest	The request message is either invalid or is not well-formed
MissingParameter	A parameter is missing
NoApplicableCode	There is no applicable code to this exception
NotImplemented	The (abstract) operation has not been implemented
NotSupported	A service option or capability is not supported
NullResourceValue	A requested resource has a null value
NullValue	Null value exception: a required parameter or variable is null
ResourceNotFound	The requested resource does not exist or could not be found
ResourceTypeMismatch	The requested resource does not have the required type
UnknownError	The error type is unknown

In addition to a code, an exception SHOULD have a locator that identifies the origin of the exception (e.g. the invalid or missing parameter or value, etc.). The locator is specified using the locator attribute of the <Exception> element.

The XML fragment below illustrates an exception report generated by the SWS due to an unknown concept scheme specified in a *GetConceptScheme* operation (c.f., subsection 5.3.3).

```
<sws:ExceptionReport xmlns:sws="http://cmrc.ucc.ie/sws/2.0"
 xmlns:xml=http://www.w3.org/XML/1998/namespace xml:lang="en" version="1.0">
  <sws:Exception exceptionCode="ResourceNotFound"
   locator="http://netmar.ucc.ie/ont/20110901/geoscience.owl#BadCS">
    <sws:ExceptionText>
      Resource "http://geodi.ucc.ie/ont/20110901/geoscience.owl#BadCS": No such
      concept scheme or resource.
    </sws:ExceptionText>
  </sws:Exception>
</sws:ExceptionReport>
```


6 Functional Requirements

This chapter specifies the operations to be supported by the NETMAR semantic web service, version 2.0. Table 6.1 defines the SWS 2.0 operations and specifies their use: mandatory (M) or optional (O). SWS operations are split into two functional interfaces: discovery interface (M), and semantic interface (M). The former provides the operations required for requesting service metadata. The latter supports the most common semantic operations required by SWS clients.

Table 6.1. SWS Operations

Discovery Interface			
1	*GetCapabilities*	Retrieves service metadata, including supported operations, response formats, available concept schemes, and their supported languages.	M
Semantic Interface			
2	*GetConceptSchemes*	Lists available concept schemes with their annotations (labels, definitions, etc.).	M
3	*GetConceptScheme*	Returns a concept scheme definition given its URI.	M
4	*SearchConceptScheme*	Returns the definition(s) of one or more concept scheme(s) matching a specified free-text keyword.	M
5	*GetConceptSchemeContent*	Returns the content of a specified concept scheme identified by its URI, including its collections and concepts.	O
6	*GetCollections*	Lists available concept collections with their annotations. Collections may be filtered by one or more concept schemes.	O
7	*GetCollection*	Returns a collection definition identified by its URI.	O
8	*SearchCollection*	Returns the definition(s) of one or more collection(s) matching a specified free-text keyword.	O
9	*GetCollectionContent*	Returns the content of a given collection identified by its URI, including member collections and concepts.	O
10	*GetConcepts*	Returns the definitions of the concepts belonging to a specified concept scheme and/or collection.	M
11	*GetConcept*	Returns a concept definition given its URI. The response includes the concept's annotations.	M
12	*SearchConcept*	A smart search operation that returns the concepts matching a given keyword.	M
13	*GetRelatedConcepts*	Returns the concepts related to one or many given concept(s) using one or many given SKOS relationship(s) both from direct assertions and by entailment.	M
14	*GetExplicitTopConcepts*	Returns the concepts that have explicitly been asserted as top concepts of a specified concept scheme.	O
15	*GetImplicitTopConcepts*	Returns the concepts of one concept scheme that have no broader concepts within the latter.	M
16	*GetConceptHierarchy*	Returns the hierarchy of the concepts within a given concept scheme and/or collection.	O
17	*InterpretKeyword*	Returns the concepts that semantically match a given keyword, within a specified concept scheme and or collection.	O
18	*CheckRelation*	Checks whether two specified concepts are related via a specified SKOS relationship.	O

This chapter specifies the operations listed in Table 6.1, and defines their request and response formats. Request formats are provided in both KVP and XML. Fragments of XML schemas are used to describe the XML structures of the SWS requests. The full XML schema for the SWS requests is

available at http://netmar.ucc.ie/xsd/sws/2.0/requests.xsd. An HTML documentation for this XML schema is available at the following URL: http://netmar.ucc.ie/xsd/2.0/doc/requests.html.

6.1 Common SWS Request Parameters

All SWS operation requests, except for GetCapabilities, SHALL include the following four parameters.

- Service type, this should be "SWS" for the semantic web service;
- Semantic web service version;
- Request name (e.g., GetConceptSchemes, GetConcept, etc.);
- Response format (e.g., XML, JSON, etc.)

In addition to these four common parameters, the semantic interface operations 2 to 17 share the following two parameters.

- The response language parameter, which specifies the language to be used for the lexical labels and documentation properties in the server response. If the language parameter is omitted, the server shall retrieve lexical labels and documentation in all available languages.
- The element set parameter, which specifies the amount of information (detail level) to be returned by the SWS. The element set parameter has the following five possible values, called *element set names*.
  - "***abstract***": only the resource URI SHALL be returned.
  - "***brief***": abstract information and preferred label SHALL be returned.
  - "***summary***": brief information, associated concept schemes (if applicable), and definitions SHALL be returned.
  - "***full***": summary information and alternate and hidden labels SHALL be returned.
  - "***extended***": full information and any other information (e.g., mappings) SHALL be returned.

The response detail levels for the element set names above are defined using the RDF graph templates specified in Table 6.2.

Table 6.2. RDF Graph Template for each Element Set Name

Element Set Name	RDF Graph Template	Comment
abstract	?resource rdf:type ?type .	
brief	?resource rdf:type ?type ; skos:prefLabel ?pl .	
summary	?resource rdf:type ?type ; skos:prefLabel ?pl ; skos:definition ?def ; skos:inScheme ?cs .	skos:inScheme only applicable to concepts and collections
full	?resource rdf:type ?type ; skos:prefLabel ?pl ; skos:definition ?def ; skos:inScheme ?cs ; skos:altLabel ?al ; skos:hiddenLabel ?hl .	skos:inScheme only applicable to concepts and collections
extended	?resource ?predicate ?object .	

6.1.1 *KVP Encoding of the Common Parameters*

In KVP encoding, the above common request parameters are encoded as specified in Table 6.3. Note that the parameter names in all KVP encodings SHALL be handled in a **case insensitive** manner.

Table 6.3. Common SWS Request Parameters

Parameter	Cardinality	Definition	Example
service	1	Type of service requested ***Possible values:*** "SWS"	service=SWS
version	*	Service version accepted by the client ***Possible values:*** "2.0", "1.0"	version=2.0
request	1	Operation requested by the client ***Possible values:*** see Table 6.1	request=GetConcept
acceptFormat	0..1	Response format expected by the client ***Possible values:*** "text/xml", "application/json"	AcceptFormat=text/xml
responseLanguage	0..1	Response language ***Type:*** ISO 639-1 two-letter language code	responseLanguage=en
elementSet	0..1	Level of resource details returned by the SWS ***Possible values and meanings:*** • "abstract": *only URI SHALL be returned* • "brief": *abstract information and preferred label SHALL be returned* • "summary": *brief information, associated concept schemes (if applicable), and definitions SHALL be returned* • "full": *summary information and alternate labels SHALL be returned* • "extended": *full information and hidden labels SHALL be returned*	elementSet=full

6.1.2 *XML Encoding of the Common Parameters*

In XML encoding, all SWS operation request elements, except for *GetCapabilities and CheckRelation*, extend the abstract sws:SWSRequestType which is defined by the following XML schema fragment. For the full definition of the sws:SWSRequestType type and all the SWS XML requests, see the XML schema at http://netmar.ucc.ie/xsd/sws/2.0/requests.xsd.


```
<!--SWSRequestType is an abstract base type for all SWS requests-->
<xs:complexType name="SWSRequestType" abstract="true">
  <xs:sequence>
    <!--Server versions accepted-->
    <!--When omitted, server shall return latest supported version.-->
    <xs:element name="AcceptVersions" type="sws:SWSVersionListType" minOccurs="0"/>

    <!--Response format accepted-->
    <!--When omitted, server shall return service metadata document using-->
    <!--the MIME type "text/xml". If the specified format is not supported,-->
    <!--the server shall raise a NotSupported exception.-->
    <xs:element name="AcceptFormat" type="sws:ResponseFormatType" minOccurs="0"/>

    <!--Element set name, optional. Default is "full"-->
    <xs:element name="ElementSet" type="sws:ElementSetNameType"/>
  </xs:sequence>

  <!--Service type, default is SWS (Semantic Web Service)-->
  <xs:attribute name="service" type="sws:ServiceType" use="required"/>

  <!--Language used for lexical labels and documentation, optional.-->
  <!--If omitted, server shall retrieve lexical labels and documentation-->
  <!--in all available languages.-->
  <!--If specified language is not supported, server shall return empty-->
  <!--lexical labels and documentation.-->
  <xs:attribute name="responseLanguage"" type="sws:LanguageType" use="optional"/>
</xs:complexType>
```

The sws:AcceptVersionsType, sws:AcceptFormatsType, and sws:ServiceType, and sws:ElementSetNameType types are defined in the SWS requests XML schemas.

6.2 GetCapabilities Operation

The semantic web service SHALL have the ability to describe its capabilities by returning service metadata in response to a *GetCapabilities* request. The GetCapabilities operation is more completely specified in Table 6.3. Its parameters are specified in Table 6.4.

Table 6.4. Definition of the *GetCapabilities* Operation

Definition	Allows clients to retrieve service metadata
Receives	Optional capabilities document section names
Returns	Service capabilities document including the sections specified in the request
Exceptions	InvalidParameterValue, InternalError

6.2.1 *KVP Encoding*

The *GetCapabilities* request uses the *section* parameter that specifies the capabilities document sections requested by the client (c.f., Table 6.5).

Table 6.5. *GetCapabilities* Request Parameters

Parameter	Cardinality	Definition	Example
section	0..4	Capabilities document section requested ***Possible values:*** • "ServerIdentification" • "ServiceProvider" • "OperationsMetadata" • "SupportedConceptSchemeList"	section=ServiceProvider &section=OperationsMetadata

6.2.2 *XML Encoding*

The <sws:GetCapabilities> element is used to request a capabilities document from a semantic web service; it is defined by the following XML Schema fragment.

```
<!--GetCapabilities request-->
<xs:element name="GetCapabilities" type="sws:GetCapabilitiesType"/>

<!--GetCapabilitiesType-->
<xs:complexType name="GetCapabilitiesType">
  <xs:sequence>
    <!--Capabilities sections-->
    <!--When omitted or not supported by server, server shall return-->
    <!--complete service metadata (Capabilities) document.-->
    <xs:element name="Sections" type="sws:CapabilitiesSectionListType"
      minOccurs="0"/>
  </xs:sequence>
  <!--Service type, default is SWS (Semantic Web Service)-->
  <xs:attribute name="service" type="sws:ServiceType" use="required"/>
</xs:complexType>
```

The optional Sections element of type sws:CapabilitiesSectionListType specifies the capabilities sections requested by the client (c.f., next subsection).

6.2.3 *Response*

The root element of the response to a *GetCapabilities* request is the <sws:GetCapabilitiesResponse> element, which is partially defined in the following XML Schema fragment.

```
<!--GetCapabilitiesResponse-->
<xs:element name="GetCapabilitiesResponse" type="sws:GetCapabilitiesResponseType"/>

<!--GetCapabilitiesResponseType-->
<xs:complexType name="GetCapabilitiesResponseType">
  <xs:sequence>

    <!--Server identification information section.-->
    <!--Contains service type and service type version.-->
    <xs:element ref="sws:ServiceIdentification" minOccurs="0"/>

    <!--Service provider section-->
    <!--Conatins provider name, provider site, and contact information-->
    <xs:element ref="sws:ServiceProvider" minOccurs="0"/>

    <!--Supported operations section-->
    <!--Contains information about the operations supported by the SWS.-->
    <xs:element ref="sws:OperationsMetadata" minOccurs="0"/>

    <!--Concept schemes section-->
    <!--List of concept schemes delivered by the SWS-->
    <xs:element name="SupportedConceptSchemes" type="sws:ConceptSchemeListType"
      minOccurs="0"/>

    <!--Service version ,required-->
    <xs:attribute name="version" type="sws:SWSVersionType" use="required"/>
  </xs:sequence>
</xs:complexType>
```


The capabilities response document contains the following sections:

1. **Server Identification Section –** Provides information about the SWS itself;
2. **Service Provider Section –** Provides metadata about the organisation operating the semantic web service;
3. **Operations Metadata Section –** Provides the list of SWS operations implemented by the SWS instance;
4. **Supported Concept Schemes Section –** Provides the list of concept schemes delivered by the SWS, including their SKOS lexical labels and annotations.

6.3 GetConceptSchemes Operation

The mandatory GetConceptSchemes operation is used to retrieve information about the available concept schemes. This operation is more completely specified in Table 6.6.

Table 6.6. Definition of the *GetConceptSchemes* Operation

Definition	Allows clients to describe the available concept schemes delivered by the SWS
Receives	Optionally, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language
Returns	List of available concept schemes with their annotation properties, encoded in RDF/XML.
Exceptions	InvalidParameterValue, InternalError

6.3.1 *KVP Encoding*

The *GetConceptSchemes* operation does not require more parameters than the ones defined in Table 6.3. An example of a *GetConceptSchemes* request encoded in KVP is shown blow.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetConceptSchemes
&elementSet=full
&responseLanguage=en
```

6.3.2 *XML Encoding*

The following XML Schema fragment defines the XML encoding of the *GetConceptSchemes* operation.

```
<!--GetConceptSchemes-->
<xs:element name="GetConceptSchemes" type="sws:GetConceptSchemesType"/>

<!--GetConceptSchemesType-->
<xs:complexType name="GetConceptSchemesType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType"/>
  </xs:complexContent>
</xs:complexType>
```

6.3.3 *Response*

The *GetConceptSchemes* response consists is an RDF document containing the list of concepts schemes delivered by the SWS.

The XML below shows a fragment of the *GetConceptSchemes* response associated with the request example above. For sake of readability, namespace declarations and concept schemes URI namaespaces have been omitted.

```
<rdf:RDF>
  <skos:ConceptScheme rdf:about="Themes">
    <skos:prefLabel xml:lang="en">Themes</skos:prefLabel>
    <skos:definition xml:lang="en">
      A concept scheme for defining theme keywords
    </skos:definition>
  </skos:ConceptScheme>
  <skos:ConceptScheme rdf:about="Parameters">
    <skos:prefLabel xml:lang="en">Parameters</skos:prefLabel>
    <skos:definition xml:lang="en">
      A concept scheme for defining parameter keywords
    </skos:definition>
  </skos:ConceptScheme>
  <!--Other concept schemes-->
</rdf:RDF>
```

6.4 GetConceptScheme Operation

The *GetConceptScheme* operation, which is fully defined in Table 6.7, allows a client to retrieve information about a specified concept scheme identified by its URI.

Table 6.7. Definition of the *GetConceptScheme* Operation

Definition	Allows clients to describe a concept scheme specified by its URI
Receives	The URI of a concept scheme of interest, and optionally, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language
Returns	Definition of the concept scheme specified in the request, encoded in RDF/XML.
Exceptions	InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError

6.4.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetConceptScheme* request uses the *conceptScheme* parameter that specifies the URI of the concept scheme requested by the client (c.f., Table 6.8).

Table 6.8. *GetConceptScheme* Request Parameters

Parameter	Cardinality	Definition	Example
conceptScheme	1	URI of the concept scheme requested ***Type:*** URI	conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes

An example of a *GetConceptScheme* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetConceptScheme
&elementSet=full
&responseLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes
```


6.4.2 *XML Encoding*

The <sws:GetConceptScheme> element is used to call the *GetConceptScheme* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetConceptScheme request-->
<xs:element name="GetConceptScheme" type="sws:GetConceptSchemeType"/>

<!--GetConceptSchemeType-->
<xs:complexType name="GetConceptSchemeType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept scheme, mandatory-->
        <xs:element name="ConceptScheme" type="xs:anyURI"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

6.4.3 *Response*

The *GetConceptScheme* response is an RDF document describing the requested concept scheme. The XML below shows a fragment of the *GetConceptScheme* response associated with the request example above.

```
<rdf:RDF>
  <skos:ConceptScheme rdf:about="Themes">
    <skos:prefLabel xml:lang="en">Themes</skos:prefLabel>
    <skos:definition xml:lang="en">
      A concept scheme for defining theme keywords
    </skos:definition>
  </skos:ConceptScheme>
</rdf:RDF>
```

6.5 SearchConceptScheme Operation

The optional *SearchConceptScheme* operation, which is fully defined in Table 6.9, allows a client to retrieve information about one or more concept scheme(s) matching a free-text keyword.

Table 6.9. Definition of the *SearchConceptScheme* Operation

Definition	Allows clients to search one or more concept schemes by keyword.
Receives	A free-text keyword, and optionally, the keyword's language, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language
Returns	Definition of the concept schemes matching the specified keyword in the specified language, encoded in RDF/XML. If the keyword language is not specified in the request, then all languages are considered.
Exceptions	InvalidParameterValue, MissingParameter, InternalError

6.5.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *SearchConceptScheme* request uses the *keyword* parameter and its language (*keywordLanguage*), c.f., Table 6.8.

Table 6.10. *SearchConceptScheme* Request Parameters

Parameter	Cardinality	Definition	Example
keyword	1	Search keyword ***Type:*** free text	keyword=Theme
keywordLanguage	0..1	Keyword language ***Type:*** ISO 639-1 two-letter language code	keywordLanguage=en

An example of a *SearchConceptScheme* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=SearchConceptScheme
&elementSet=full
&responseLanguage=en
&keyword=Theme
&keywordLanguage=en
```

6.5.2 *XML Encoding*

The <sws:SearchConceptScheme> element is used to call the *SearchConceptScheme* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--SearchConceptScheme request-->
<xs:element name="SearchConceptScheme" type="sws:SearchConceptSchemeType"/>

<!--SearchConceptSchemeType-->
<xs:complexType name="SearchConceptSchemeType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--Free-text keyword, mandatory-->
        <xs:element name="Keyword" type="sws:KeywordType"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The sws:KeywordType is a data type that contains a free-text keyword and a language attribute (xml:lang).

6.5.3 *Response*

The *SearchConceptScheme* response consists of an RDF document describing the concept scheme(s) matching the keyword specified in the request. The XML below shows a fragment of the *SearchConceptScheme* response associated with the request example above.

```
<rdf:RDF>
  <skos:ConceptScheme rdf:about="Themes">
    <skos:prefLabel xml:lang="en">Themes</skos:prefLabel>
    <skos:definition xml:lang="en">
      A concept scheme for defining theme keywords
    </skos:definition>
  </skos:ConceptScheme>
</rdf:RDF>
```

6.6 GetConceptSchemeContent Operation

The optional *GetConceptSchemeContent* operation, which is fully defined in Table 6.11, allows a client to retrieve the content of a specified concept scheme identified by its URI, including its collections and concepts.

Table 6.11. Definition of the *GetConcetpSchemeContent* Operation

Definition	Allows clients to retrieve the content (collections and concepts) of a specified concept scheme.
Receives	The URI of the concept scheme of interest and, optionally, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language
Returns	List of collections and concepts belonging to the specified concept scheme, with their annotation properties in the specified language, encoded in RDF/XML.
Exceptions	InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError

6.6.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetConceptSchemeContent* request uses the *conceptScheme* parameter that specifies the URI of the concept scheme requested by the client (c.f., Table 6.12).

Table 6.12. *GetConceptSchemeContent* Request Parameters

Parameter	Cardinality	Definition	Example
conceptScheme	1	URI of the concept scheme requested ***Type:*** URI	conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes

An example of a *GetConceptSchemeContent* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetConceptSchemeContent
&elementSet=brief
&responseLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments
```

6.6.2 *XML Encoding*

The <sws:GetConceptSchemeContent> element is used to call the *GetConceptSchemeContent* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetConceptSchemeContent request-->
<xs:element name="GetConceptSchemeContent" type="sws:GetConceptSchemeContentType"/>

<!--GetConceptSchemeContentType-->
<xs:complexType name="GetConceptSchemeContentType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept scheme, mandatory-->
        <xs:element name="ConceptScheme" type="xs:anyURI"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

6.6.3 *Response*

The *GetConceptSchemeContent* response is an RDF document listing and describing the concepts and collections belonging to the requested concept scheme. The XML below shows a fragment of the *GetConceptSchemeContent* response associated with the request example above.

```
<rdf:RDF>
  <skos:Collection rdf:about="RemoteSensingInstruments">
    <skos:prefLabel xml:lang="en">Remote Sensing Instruments</skos:prefLabel>
  </skos: Collection>
  <skos:Collection rdf:about="InSituLaboratoryInstruments">
    <skos:prefLabel xml:lang="en">In Situ/Laboratory
      Instruments</skos:prefLabel>
  </skos:Collection>
  <!--More collections-->
  <skos:Concept rdf:about="SidescanSonar">
    <skos:prefLabel xml:lang="en">Sidescan Sonar</skos:prefLabel>
  </skos:Concept>
  <!--More concepts-->
</rdf:RDF>
```

6.7 GetCollections Operation

The optional *GetCollections* operation, which is fully defined in Table 6.13, allows a client to retrieve the list of collections available, with the possibility of filtering by concept schemes.

Table 6.13. Definition of the *GetConcetpSchemeContent* Operation

Definition	Allows a client to retrieve the list of available collections, possibly within one or more specified concept schemes.
Receives	Optionally, the URIs of the concept schemes of interest and, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language
Returns	List of collections belonging to the specified concept schemes, with their annotation properties in the specified language, encoded in RDF/XML.
Exceptions	InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError

6.7.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetConceptSchemeContent* request uses the *conceptScheme* parameter that specifies the URI of the concept scheme of interest (c.f., Table 6.14).

Table 6.14. *GetCollections* Request Parameters

Parameter	Cardinality	Definition	Example
conceptScheme	*	URI of the concept scheme requested ***Type:*** URI	conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments

An example of a *GetCollections* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetCollections
&elementSet=brief
&responseLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments
```

6.7.2 *XML Encoding*

The <sws:GetCollections> element is used to call the *GetCollections* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetCollections request-->
<xs:element name="GetCollections" type="sws:GetCollectionsType"/>

<!--GetCollectionsType-->
<xs:complexType name="GetCollectionsType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept scheme-->
        <xs:element name="ConceptScheme" type="xs:anyURI" minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

6.7.3 *Response*

The *GetCollections* response is an RDF document describing the available collections belonging to the specified concept scheme. The XML below shows a fragment of the *GetCollections* response associated with the request example above.

```
<rdf:RDF>
  <skos:Collection rdf:about="RemoteSensingInstruments">
    <skos:prefLabel xml:lang="en">Remote Sensing Instruments</skos:prefLabel>
  </skos: Collection>
  <skos:Collection rdf:about="InSituLaboratoryInstruments">
    <skos:prefLabel xml:lang="en">In Situ/Laboratory
      Instruments</skos:prefLabel>
  </skos:Collection>
  <!--More collections-->
</rdf:RDF>
```

6.8 GetCollection Operation

The *GetCollection* operation, which is fully defined in Table 6.15, allows a client to retrieve information about a specified concept collection identified by its URI.

Table 6.15. Definition of the *GetCollection* Operation

Definition	Allows clients to describe a collection specified by its URI
Receives	The URI of a concept collection of interest, and optionally, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language
Returns	Definition of the concept collection specified in the request, encoded in RDF/XML.
Exceptions	InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError

6.8.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetCollection* request uses the *collection* parameter that specifies the URI of the concept collection requested by the client (c.f., Table 6.16).

Table 6.16. *GetCollection* Request Parameters

Parameter	Cardinality	Definition	Example
collection	1	URI of the concept collection requested ***Type:*** URI	collection=http://netmar.ucc.ie/ont/20120801/geoscience.owl#RemoteSensingInstruments

An example of a *GetCollection* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetCollection
&elementSet=full
&responseLanguage=en
&collection=http://netmar.ucc.ie/ont/20120801/geoscience.owl#RemoteSensingInstruments
```

6.8.2 *XML Encoding*

The <sws:GetCollection> element is used to call the *GetCollection* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetCollection request-->
<xs:element name="GetCollection" type="sws:GetCollectionType"/>

<!--GetCollectionType-->
<xs:complexType name="GetCollectionType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept scheme, mandatory-->
        <xs:element name="Collection" type="xs:anyURI"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

6.8.3 *Response*

The *GetCollection* response is an RDF document describing the requested collection. The XML below shows a fragment of the *GetCollection* response associated with the request example above.

```
<rdf:RDF>
  <skos:Collection rdf:about="RemoteSensingInstruments">
    <skos:prefLabel xml:lang="en">Remote Sensing Instruments</skos:prefLabel>
    <skos:definition xml:lang="en">
      Instruments that are deployed on or from a platform and that allow the
      collection of data about the earth's land or water areas
    </skos:definition>
  </skos:Collection>
</rdf:RDF>
```

6.9 SearchCollection Operation

The optional *SearchCollection* operation, which is fully defined in Table 6.17, allows a client to retrieve information about one or more collection(s) matching a free-text keyword.

Table 6.17. Definition of the *SearchCollection* Operation

Definition	Allows a client to search one or more concept collections by keyword.
Receives	A free-text keyword, and optionally, the keyword's language, the URIs of the concept schemes of interest, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language.
Returns	Definition of the collections matching the specified keyword in the specified language, encoded in RDF/XML. If the keyword language is not specified in the request, then all languages are considered.
Exceptions	InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError.

6.9.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *SearchCollection* request uses the *keyword* parameter, its language (*keywordLanguage*), and the conceptScheme parametr which specifies the URI of a concept scheme of interest, c.f., Table 6.18.

Table 6.18. *SearchCollection* Request Parameters

Parameter	Cardinality	Definition	Example
keyword	1	Search keyword ***Type:*** free text	keyword=remote sensing
keywordLanguage	0..1	Keyword language ***Type:*** ISO 639-1 two-letter language code	keywordLanguage=en
conceptScheme	*	URI of a concept scheme of interest ***Type:*** URI	conceptScheme= http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments

An example of a *SearchCollection* request is provided below.

<Service URL>?
service=SWS
&version=2.0
&request=SearchCollection
&elementSet=full
&responseLanguage=en
&keyword=remote sensing
&keywordLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments

6.9.2 *XML Encoding*

The <sws:SearchCollection> element is used to call the *SearchCollection* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--SearchCollection request-->
<xs:element name="SearchCollection" type="sws:SearchCollectionType"/>

<!--SearchCollectionType-->
<xs:complexType name="SearchCollectionType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--Free-text keyword, mandatory-->
        <xs:element name="Keyword" type="sws:KeywordType"/>

        <!--URI of the requested concept schemes-->
        <xs:element name="ConceptScheme" type="xs:anyURI" minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The sws:KeywordType is a data type that contains a free-text keyword and a language attribute (xml:lang).

6.9.3 *Response*

The *SearchCollection* response is an RDF document describing the collection(s) matching the keyword specified in the request and belonging to the concept schemes of interest. The XML below shows a fragment of the *SearchCollection* response associated with the request example above.


```
<rdf:RDF>
  <skos:Collection rdf:about="RemoteSensingInstruments">
    <skos:prefLabel xml:lang="en">Remote Sensing Instruments</skos:prefLabel>
    <skos:definition xml:lang="en">
      Instruments that are deployed on or from a platform and that allow the
      collection of data about the earth's land or water areas
    </skos:definition>
  </skos:Collection>
</rdf:RDF>
```

6.10 GetCollectionContent Operation

The optional *GetCollectionContent* operation, which is fully defined in Table 6.19, allows a client to retrieve the content of a specified collection identified by its URI, including its member collections and concepts. The content may be filtered by concept schemes.

Table 6.19. Definition of the *GetCollectionContent* Operation

Definition	Allows a client to retrieve the content (member collections and concepts) of a specified collection.
Receives	The URI of the collection of interest and, optionally, the URIs of the concept schemes of interest, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language
Returns	List of collections and concepts belonging to the specified collection and concept schemes, with their annotation properties in the specified language, encoded in RDF/XML.
Exceptions	InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError

6.10.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetCollectionContent* request uses the collection parameter that specifies the URI of the requested collection, and the *conceptScheme* parameter that specifies the URI of a concept scheme to which the returned resources must belong (c.f., Table 6.20).

Table 6.20. *GetCollectionContent* Request Parameters

Parameter	Cardinality	Definition	Example
collection	1	URI of the collection requested ***Type:*** URI	collection=http://netmar.ucc.ie/ont/20120801/geoscience.owl#RemoteSensingInstruments
conceptScheme	*	URI of the concept scheme requested ***Type:*** URI	conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments

An example of a *GetCollectionContent* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetCollectionContent
&elementSet=brief
```

```
&responseLanguage=en
&collection=http://netmar.ucc.ie/ont/20120801/geoscience.owl#RemoteSensingInstruments
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments
```

6.10.2 *XML Encoding*

The <sws:GetCollectionContent> element is used to call the *GetCollectionContent* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetCollectionContent request-->
<xs:element name="GetCollectionContent" type="sws:GetCollectionContentType"/>

<!--GetCollectionContentType-->
<xs:complexType name="GetCollectionContentType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested collection, mandatory-->
        <xs:element name="Collection" type="xs:anyURI"/>

        <!--URIs of the requested concept schemes-->
        <xs:element name="ConceptScheme" type="xs:anyURI" minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

6.10.3 *Response*

The *GetCollectionContent* response is an RDF document listing and describing the concepts and collections belonging to the requested concept scheme. The XML below shows a fragment of the *GetCollectionContent* response associated with the request example above.

```
<rdf:RDF>
  <skos:Collection rdf:about="ActiveRemoteSensingInstruments">
    <skos:prefLabel xml:lang="en">
      Active Remote Sensing Instruments
    </skos:prefLabel>
  </skos: Collection>
  <skos:Collection rdf:about="PassiveRemoteSensingInstruments">
    <skos:prefLabel xml:lang="en">
      Passive Remote Sensing Instruments
    </skos:prefLabel>
  </skos:Collection>
  <!--More collections-->
</rdf:RDF>
```

6.11 GetConcepts Operation

The mandatory *GetConcepts* operation, which is fully defined in Table 6.21, retrieves all the concepts of one concept scheme with possible filtering by collections.

Table 6.21. Definition of the *GetCollectionContent* Operation

Definition	Allows a client to retrieve the concepts of a specified concept scheme with possible filtering by collections.
Receives	The URI of the concept scheme of interest, and optionally, the URIs of the collections of

	interest, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language.
Returns	List of concepts belonging to the specified concept scheme and collections, with their annotation properties in the specified language, encoded in RDF/XML.
Exceptions	InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError

6.11.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetConcepts* request uses the *conceptScheme* parameter that specifies the URI of the concept scheme of interest, the collection parameter that specifies the URI of the collections to which the returned concepts must belong (c.f., Table 6.20).

Table 6.22. *GetConcepts* Request Parameters

Parameter	Cardinality	Definition	Example
conceptScheme	1	URI of the concept scheme requested ***Type:*** URI	conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments
collection	*	URI of a collection requested ***Type:*** URI	collection=http://netmar.ucc.ie/ont/20120801/geoscience.owl#RemoteSensingInstruments

An example of a *GetConcepts* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetConcepts
&elementSet=brief
&responseLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Instruments
&collection=http://netmar.ucc.ie/ont/20120801/geoscience.owl#ActiveRemoteSensingInstruments
```

6.11.2 *XML Encoding*

The <sws:GetConcepts> element is used to call the *GetConcepts* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetConcepts request-->
<xs:element name="GetConcepts" type="sws:GetConceptsType"/>

<!--GetConceptsType-->
<xs:complexType name="GetConceptsType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept scheme, mandatory-->
        <xs:element name="ConceptScheme" type="xs:anyURI"/>

        <!--URIs of the collections of interest-->
        <xs:element name="Collection" type="xs:anyURI" minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

6.11.3 *Response*

The *GetConcepts* response is an RDF document listing and describing the concepts belonging to the requested concept scheme and collections (if specified). The XML below shows a fragment of the *GetConcepts* response associated with the request example above.

```
<rdf:RDF>
  <skos:Concept rdf:about="SidescanSonar">
    <skos:prefLabel xml:lang="en">Sidescan Sonar</skos:prefLabel>
  </skos:Concept>
  <skos:Concept rdf:about="LiDAR">
    <skos:prefLabel xml:lang="en">LiDAR</skos:prefLabel>
  </skos:Concept>
  <!--More concepts-->
</rdf:RDF>
```

6.12 GetConcept Operation

The *GetConcept* operation, which is fully defined in Table 6.23, allows a client to retrieve information about a specified concept collection identified by its URI.

Table 6.23. Definition of the *GetConcept* Operation

Definition	Allows clients to describe a Concept specified by its URI
Receives	The URI of a concept of interest, and optionally, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language
Returns	Definition of the concept specified in the request, encoded in RDF/XML.
Exceptions	InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError

6.12.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetConcept* request uses the *concept* parameter that specifies the URI of the concept requested by the client (c.f., Table 6.24).

Table 6.24. *GetConcept* Request Parameters

Parameter	Cardinality	Definition	Example
concept	1	URI of the concept requested ***Type:*** URI	concept=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Geology

An example of a *GetConcept* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetCollection
&elementSet=full
&responseLanguage=en
&concept=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Geology
```

6.12.2 *XML Encoding*

The <sws:GetConcept> element is used to call the *GetConcept* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetConcept request-->
<xs:element name="GetConcept" type="sws:GetConceptType"/>

<!--GetConceptType-->
<xs:complexType name="GetConceptType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept, mandatory-->
        <xs:element name="Concept" type="xs:anyURI"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

6.12.3 *Response*

The *GetConcept* response is an RDF document describing the requested collection. The XML below shows a fragment of the *GetConcept* response associated with the request example above.

```
<rdf:RDF>
  <skos:Concept rdf:about="Geology">
    <skos:prefLabel xml:lang="en">Geology</skos:prefLabel>
    <skos:inScheme rdf:about="Themes"/>
    <skos:inScheme rdf:about="Disciplines"/>
    <skos:inScheme rdf:about="INSPIREThemes_Annex2"/>
    <skos:definition xml:lang="en">
      Geology characterised according to composition and structure.
      Includes bedrock, aquifers and geomorphology.
    </skos:definition>
  </skos:Concept>
</rdf:RDF>
```

6.13 SearchConcept Operation

The *SearchConcept* operation, which is fully defined in Table 6.25, allows a client to retrieve information about one or more concept(s) matching a free-text keyword.

Table 6.25. Definition of the *SearchConcept* Operation

Definition	Allows a client to search one or more concept collections by keyword.
Receives	A free-text keyword, and optionally, the keyword's language, the URI of a concept scheme of interest, the URIs of the collections of interest, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language.
Returns	Definition of the concepts matching the specified keyword in the specified language, and belonging to the specified concept scheme and/or collections, encoded in RDF/XML. If the keyword language is not specified in the request, then all languages are considered.
Exceptions	InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError.

6.13.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *SearchConcept* request uses the *keyword* parameter, and optionally its language (*keywordLanguage*), the conceptScheme parameter that specifies the URI of the concept scheme of interest, and the collection parameter that specifies the URI of a collection of interest, c.f., Table 6.26.

Table 6.26. *SearchConcept* Request Parameters

Parameter	Cardinality	Definition	Example
keyword	1	Search keyword ***Type:*** free text	keyword=geology
keywordLanguage	0..1	keyword language ***Type:*** ISO 639-1 two-letter language code	keywordLanguage=en
conceptScheme	0..1	URI of a concept scheme of interest ***Type:*** URI	conceptScheme= http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes
collection	*	URI of a collection of interest ***Type:*** URI	collection= http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes

An example of a *SearchConcept* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=SearchConcept
&elementSet=full
&responseLanguage=en
&keyword=geo
&keywordLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Parameters
```

6.13.2 *XML Encoding*

The <sws:SearchConcept> element is used to call the *SearchConcept* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--SearchConcept request-->
<xs:element name="SearchConcept" type="sws:SearchConceptType"/>

<!--SearchConceptType-->
<xs:complexType name="SearchConceptType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--Free-text keyword, mandatory-->
        <xs:element name="Keyword" type="sws:KeywordType"/>

        <!--URI of the requested concept scheme-->
        <xs:element name="ConceptScheme" type="xs:anyURI" minOccurs="0"/>

        <!--URI of the requested concept schemes-->
        <xs:element name="Collection" type="xs:anyURI" minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The sws:KeywordType is a data type that contains a free-text keyword and a language attribute (xml:lang).

6.13.3 *Response*

The *SearchConcept* response is an RDF document describing the concept(s) matching the keyword specified in the request and belonging to the concept scheme and/or collections of interest. The XML below shows a fragment of the *SearchConcept* response associated with the request example above.

```
<rdf:RDF>
  <skos:Concept rdf:about="GeotechnicalPropertyOfSeabedSamples">
    <skos:prefLabel xml:lang="en">
      Geotechnical Property of Seabed Samples
    </skos:prefLabel>
    <skos:inScheme rdf:about="Parameters"/>
    <skos:definition xml:lang="en">
      Parameter pertaining to the geotechnical analysis of seabed samples
    </skos:definition>
  </skos:Concept>
</rdf:RDF>
```

6.14 GetRelatedConcepts Operation

The mandatory *GetRelatedConcepts* operation, which is fully defined in Table 6.27, retrieves all the concepts related to one or more specified concepts, identified by their URIs, by one or more SKOS relationships. Concepts may be filtered by concept scheme and/or collections.

Table 6.27. Definition of the *GetRelatedConcepts* Operation

Definition	Allows a client to retrieve the concepts related to one or more concept schemes.
Receives	A list of concepts, and optionally, a list of SKOS relationship URIs, the URIs of the concepts schemes and collections of interest, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language.
Returns	Definitions of the concepts related to the specified ones using the specified relationships and belonging to the specified concept scheme and collections, in the specified language, encoded in RDF/XML.
Exceptions	InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError.

6.14.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetRelatedConcepts* request uses the parameters defined in Table 6.28.

Table 6.28 *GetRelatedConcepts* Request Parameters

Parameter	Cardinality	Definition	Example
concept	1..*	URI of the concept of interest	concept=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Geology
relationship	*	URI or local name of a SKOS relationship	relationship=http://www.w3.org/2004/02/skos/core#narrowerTransitive
conceptScheme	*	URI of a concept scheme ***Type:*** URI	conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Disciplines
collection	*	URI of a collection ***Type:*** URI	collection=http://netmar.ucc.ie/ont/20120801/geoscience.owl#RemoteSensingInstruments

An example of a *GetRelatedConcepts* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetRelatedConcepts
&elementSet=full
&responseLanguage=en
&concept=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Geology
&relationship=http://www.w3.org/2004/02/skos/core#narrowerTransitive
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes
```

6.14.2 *XML Encoding*

The <sws:GetRelatedConcepts> element is used to call the *GetRelatedConcepts* operation of an SWS. It is defined by the following XML Schema fragment.


```
<!--GetRelatedConcepts-->
<xs:element name="GetRelatedConcepts" type="sws:GetRelatedConceptsType"/>

<!--GetRelatedConceptsType-->
<xs:complexType name="GetRelatedConceptsType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept, mandatory-->
        <xs:element name="Concept" type="xs:anyURI" maxOccurs="unbounded"/>

        <!--SKOS Relationship, optional, if omitted than the SKOS relationship-->
        <!--skos:semanticRelation SHALL be considered.-->
        <xs:element ref="sws:SKOSRelationship" minOccurs="0"
          maxOccurs="unbounded"/>

        <!--URIs of target concept schemes; optional. If omitted then the-->
        <!--related concepts search is performed in all concept schemes.-->
        <xs:element name="ConceptScheme" type="xs:anyURI" minOccurs="0"
          maxOccurs="unbounded"/>

        <!--URIs of target collections; optional. If omitted then the-->
        <!--related concepts search is performed in all collections.-->
        <xs:element name="Collection" type="xs:anyURI" minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

6.14.3 *Response*

The *GetRelatedConcepts* response is an RDF document listing the concepts related to the requested concepts using the specified relationships and belonging to the specified concept schemes and or collections. The XML below shows a fragment of the *GetRelatedConcepts* response associated with the request example above.

```
<rdf:RDF>
  <skos:Concept rdf:about="AcousticSeabedClassification">
    <skos:prefLabel xml:lang="en">
      Acoustic Seabed Classification
    </skos:prefLabel>
    <skos:inScheme rdf:about="Themes"/>
    <skos:definition xml:lang="en">
      The classification of seabed based on acoustic properties. This can
      be divided into two main categories: seabed surface classification and
      seabed sub-surface classification (sub bottom profiling).
    </skos:definition>
  </skos:Concept>
  <skos:Concept rdf:about="FaciesInterpretation">
    <skos:prefLabel xml:lang="en">Facies interpretation</skos:prefLabel>
    <skos:inScheme rdf:about="Themes"/>
    <skos:definition xml:lang="en">
      The characterisation of a rock or series of rocks reflecting their
      appearance, composition, and conditions of formation
    </skos:definition>
  </skos:Concept>
  <!--Other related concepts-->
</rdf:RDF>
```


6.15 GetExplicitTopConcepts Operation

The optional *GetExplicitTopConcepts* operation, which is fully defined in Table 6.29, retrieves all the concepts asserted as being the top concepts of a specified concept scheme identified by its URI.

Table 6.29. Definition of the *GetExplicitTopConcepts* Operation

Definition	Allows a client to retrieve the top concepts of a specified concept scheme.
Receives	The URI of the concept scheme of interest and, optionally, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language
Returns	List of concepts asserted as top concepts of the specified concept scheme, with their annotation properties in the specified language, encoded in RDF/XML.
Exceptions	InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError

6.15.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetExplicitTopConcepts* request uses the *conceptScheme* parameter that specifies the URI of the concept scheme requested by the client (c.f., Table 6.30).

Table 6.30. *GetExplicitTopConcepts* Request Parameters

Parameter	Cardinality	Definition	Example
conceptScheme	1	URI of the concept scheme requested ***Type:*** URI	conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes

An example of a *GetExplicitTopConcepts* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetExplicitTopConcepts
&elementSet=brief
&responseLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes
```

6.15.2 *XML Encoding*

The <sws:GetExplicitTopConcepts> element is used to call the *GetExplicitTopConcepts* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetExplicitTopConcepts request-->
<xs:element name="GetExplicitTopConcepts" type="sws:GetExplicitTopConceptsType"/>

<!--GetExplicitTopConceptsType-->
<xs:complexType name="GetExplicitTopConceptsType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept scheme, mandatory-->
        <xs:element name="ConceptScheme" type="xs:anyURI"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

6.15.3 *Response*

The *GetExplicitTopConcepts* response is an RDF document listing and describing the explicitly asserted top concepts of the specified concept scheme. The XML below shows a fragment of the *GetExplicitTopConcepts* response associated with the request example above.

```
<rdf:RDF>
  <skos:Concept rdf:about="Geology">
    <skos:prefLabel xml:lang="en">Geology</skos:prefLabel>
  </skos: Concept>
  <skos:Concept rdf:about="Elevation">
    <skos:prefLabel xml:lang="en">Elevation</skos:prefLabel>
  </skos:Concept>
  <!--More concepts-->
</rdf:RDF>
```

6.16 GetImplicitTopConcepts Operation

The *GetImplicitTopConcepts* operation, which is fully defined in Table 6.31, retrieves all the concepts belonging to a concept scheme and having no broader concepts within that same concept scheme. This includes the explicit top concepts.

Table 6.31. Definition of the *GetImplicitTopConcepts* Operation

Definition	Allows a client to retrieve the top concepts of a specified concept scheme.
Receives	The URI of the concept scheme of interest and, optionally, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language
Returns	List of concepts belonging to the specified concept scheme and having no broader concepts within that concept scheme, with their annotation properties in the specified language, encoded in RDF/XML.
Exceptions	InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError

6.16.1 *KVP Encoding*

Like the GetExplicitTopConcepts operation, the *GetImplicitTopConcepts* request uses the *conceptScheme* parameter that specifies the URI of the concept scheme requested by the client (c.f., Table 3.32).

Table 6.32. *GetImplicitTopConcepts* Request Parameters

Parameter	Cardinality	Definition	Example
conceptScheme	1	URI of the concept scheme requested ***Type:*** URI	conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes

An example of a *GetImplicitTopConcepts* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetImplicitTopConcepts
&elementSet=brief
&responseLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes
```

6.16.2 *XML Encoding*

The <sws:GetImplicitTopConcepts> element is used to call the *GetImplicitTopConcepts* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetImplicitTopConcepts request-->
<xs:element name="GetImplicitTopConcepts" type="sws:GetImplicitTopConceptsType"/>

<!--GetImplicitTopConceptsType-->
<xs:complexType name="GetImplicitTopConceptsType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept scheme, mandatory-->
        <xs:element name="ConceptScheme" type="xs:anyURI"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

6.16.3 *Response*

The *GetImplicitTopConcepts* response is an RDF document listing and describing the implicit top concepts of the specified concept scheme. The XML below shows a fragment of the *GetImplicitTopConcepts* response associated with the request example above.

```
<rdf:RDF>
  <skos:Concept rdf:about="Geology">
    <skos:prefLabel xml:lang="en">Geology</skos:prefLabel>
  </skos: Concept>
  <skos:Concept rdf:about="Elevation">
    <skos:prefLabel xml:lang="en">Elevation</skos:prefLabel>
  </skos:Concept>
  <!--More concepts-->
</rdf:RDF>
```


6.17 GetConceptHierarchy Operation

The *GetConceptHierarchy* operation, which is fully defined in Table 6.33, retrieves all the concepts of a specified concept scheme, identified by its URI, organised in a hierarchy (nested structure) according to the SKOS broader and narrower relationships. This operation is suitable for small concept schemes (hundreds of concepts) and is useful for ontology browsers.

Table 6.33. Definition of the *GetConceptHierarchy* Operation

Definition	Allows a client to retrieve the hierarchy of the concepts of a specified concept scheme.
Receives	The URI of the concept scheme of interest and, optionally, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language
Returns	The concepts of the specified concept scheme, organised in a hierarchy (nested structure) according to the SKOS broader and narrower relationships, with their annotation properties in the specified language, encoded in RDF/XML.
Exceptions	InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError

6.17.1 *KVP Encoding*

In addition to the common SWS semantic request parameters (defined in Table 6.3), the *GetConceptHierarchy* request uses the *conceptScheme* parameter that specifies the URI of the concept scheme requested by the client (c.f., Table 6.34).

Table 6.34. *GetConceptHierarchy* Request Parameters

Parameter	Cardinality	Definition	Example
conceptScheme	1	URI of the concept scheme requested ***Type:*** URI	conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes

An example of a *GetConceptHierarchy* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=GetConceptHierarchy
&elementSet=brief
&responseLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Themes
```

6.17.2 *XML Encoding*

The <sws:GetConceptHierarchy> element is used to call the *GetConceptHierarchy* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--GetConceptHierarchy request-->
<xs:element name="GetConceptHierarchy" type="sws:GetConceptHierarchyType"/>

<!--GetConceptHierarchyType-->
<xs:complexType name="GetConceptHierarchyType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--URI of the requested concept scheme, mandatory-->
        <xs:element name="ConceptScheme" type="xs:anyURI"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

6.17.3 *Response*

The *GetConceptHierarchy* response is an RDF document containing the hierarchy of the concepts belonging to the requested concept scheme. The XML below shows a fragment of the *GetConceptHierarchy* response associated with the request example above.

```
<rdf:RDF>
  <skos:Concept rdf:about="NaturalRiskZones">
    <skos:prefLabel xml:lang="en">Natural risk zones</skos:prefLabel>
  </skos:Concept>
  <skos:Concept rdf:about="Geology">
    <skos:prefLabel xml:lang="en">Geology</skos:prefLabel>
    <skos:narrower>
      <skos:Concept rdf:about="MarineGeology">
        <skos:prefLabel xml:lang="en">Marine geology</skos:prefLabel>
        <skos:narrower>
          <skos:Concept rdf:about="MarineGeophysics">
            <skos:prefLabel xml:lang="en">Marine Geophysics</skos:prefLabel>
            <skos:narrower>
              <skos:Concept rdf:about="MarineGravityField">
                <skos:prefLabel xml:lang="en">
                  Marine Gravity Field
                </skos:prefLabel>
              </skos:Concept>
            </skos:narrower>
            <!--Other concepts narrower than "MarineGeophysics"-->
          </skos:Concept>
        </skos:narrower>
        <!--Other concepts narrower than "MarineGeology"-->
      </skos:Concept>
    </skos:narrower>
    <!--Other concepts narrower than "Geology"-->
  </skos:Concept>
  <!--Other top concepts-->
</rdf:RDF>
```

6.18 *InterpretKeyword Operation*

The *InterpretKeyword* operation, which is fully defined in Table 6.35, allows a client to retrieve the concept(s) that semantically match to a free-text keyword.

Table 6.35. Definition of the *InterpretKeyword* Operation

Definition	Allows a client to retrieve the concepts that semantically match a free-text keyword in a given language or a concept URI.
Receives	A free-text keyword (which may be a concept URI), and optionally, the keyword's

	language, the URI of a concept scheme of interest, the URIs of the collections of interest, the element set name, which specifies the level of detail of the resource descriptions in the response, and the response language.
Returns	Definition of the concepts matching the specified keyword in the specified language, and belonging to the specified concept scheme and/or collections, encoded in RDF/XML. If the keyword language is not specified in the request, then all languages are considered.
Exceptions	InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError.

6.18.1 *KVP Encoding*

In addition to the common SWS request parameters (defined in Table 6.3), the *InterpretKeyword* request uses the parameters defined in Table 6.36.

Table 6.36. *InterpretKeyword* Request Parameters

Parameter	Cardinality	Definition	Example
keyword	1	Search keyword ***Type:*** free text	keyword=marine geophysics
keywordLanguage	0..1	keyword language ***Type:*** ISO 639-1 two-letter language code	keywordLanguage=en
conceptScheme	0..1	URI of a concept scheme of interest ***Type:*** URI	conceptScheme= http://netmar.ucc.ie/ont/20120801/geoscience.owl#Parameters

An example of an *InterpretKeywod* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=InterpretKeyword
&elementSet=brief
&responseLanguage=en
&keyword=marine geophysics
&keywordLanguage=en
&conceptScheme=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Parameters
```

6.18.2 *XML Encoding*

The <sws:InterpretKeyword> element is used to call the *InterpretKeyword* operation of an SWS. It is defined by the following XML Schema fragment.

```
<!--InterpretKeyword request-->
<xs:element name="InterpretKeyword" type="sws:InterpretKeywordType"/>

<!--InterpretKeywordType-->
<xs:complexType name="InterpretKeywordType">
  <xs:complexContent>
    <xs:extension base="sws:SWSRequestType">
      <xs:sequence>
        <!--Free-text keyword, or concept URI, mandatory-->
        <xs:element name="Keyword" type="sws:KeywordType"/>

        <!--URI of the requested concept scheme-->
        <xs:element name="ConceptScheme" type="xs:anyURI" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The sws:KeywordType is a data type that contains a free-text keyword and a language attribute (xml:lang).

6.18.3 *Response*

The *InterpretKeyword* response is an RDF document listing the concepts related to the specified keyword and belonging to the specified concept scheme. The XML below shows a fragment of the *InterpretKeyword* response associated with the request example above.

```
<rdf:RDF>
  <skos:Concept rdf:about="MarineGravityField">
    <skos:prefLabel xml:lang="en">Marine Gravity Field</skos:prefLabel>
  </skos:Concept>
  <skos:Concept
   rdf:about="MarineMagnetics">
    <skos:prefLabel xml:lang="en">Marine Magnetics</skos:prefLabel>
  </skos:Concept>
  <!--Other concepts-->
</rdf:RDF>
```

6.19 CheckRelation Operation

This optional *CheckRelation* operation, which is fully defined in Table 6.37, checks whether two specified concepts are related via a specified SKOS relationship.

Table 6.37. Definition of the *CheckRelation* Operation

Definition	Checks whether two specified concepts are related via a specified SKOS relationship.
Receives	The URI of an origin concept scheme (subject), that of a target concept (object), and the URI or short name of a SKOS relationship (predicate).
Returns	A SPARQL response encapsulating a Boolean value: true if the concepts are related, false else.
Exceptions	InvalidParameterValue, MissingParameter, NullResourceValue, ResourceNotFound, ResourceTypeMismatch, InternalError

6.19.1 *KVP Encoding*

In addition to the common SWS request parameters, the *GetConceptHierarchy* request uses the *subject* parameter that specifies the URI of the origin concept, the *predicate* parameter that specifies the URI or local name of the SKOS relationship, and the *object* parameter that specifies the URI of the target concept (c.f., Table 6.38).

Table 6.38. *CheckRelation* Request Parameters

Parameter	Cardinality	Definition	Example
subject	1	URI of the origin concept (subject) ***Type:*** URI	subject=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Geology
predicate	1	URI or local name of a SKOS relationship (predicate)	predicate=http://www.w3.org/2004/02/skos/core#narrowerTransitive
object	1	URI of the target concept (object) ***Type:*** URI	object=http://netmar.ucc.ie/ont/20120801/geoscience.owl#MarineGeophysics

An example of a *CheckRelation* request is provided below.

```
<Service URL>?
service=SWS
&version=2.0
&request=CheckRelation
&subject=http://netmar.ucc.ie/ont/20120801/geoscience.owl#Geology
&predicate=http://www.w3.org/2004/02/skos/core#narrowerTransitive
&object=http://netmar.ucc.ie/ont/20120801/geoscience.owl#MarineGeophysics
```

6.19.2 *XML Encoding*

The <sws:CheckRelation> element is used to call the *CheckRelation* operation of an SWS. It is defined by the following XML Schema fragment.


```
<!--CheckRelation-->
<xs:element name="CheckRelation" type="sws:CheckRelationType"/>

<!--CheckRelationType-->
<xs:complexType name="CheckRelationType">
  <xs:sequence>
    <!--Server versions accepted-->
    <!--When omitted, server shall return latest supported version.-->
    <xs:element name="AcceptVersions" type="sws:SWSVersionListType" minOccurs="0"/>

    <!--Response format accepted-->
    <!--If omitted, server shall return service metadata document using-->
    <!--the MIME type "text/xml". If the specified format is not supported,-->
    <!--the server shall raise a NotSupported exception.-->
    <xs:element name="AcceptFormat" type="sws:ResponseFormatType" minOccurs="0"/>


    <!--URI of the origin concept (subject), mandatory-->
    <xs:element name="Subject" type="xs:anyURI"/>

     <!--SKOS Relationship (predicate), mandatory-->
    <xs:element name="Predicate"  type="sws:SKOSRelationshipType"/>

    <!--URI of the target concept (object), mandatory-->
    <xs:element name="Subject" type="xs:anyURI"/>
  </xs:sequence>

  <!--Service type, default is SWS (Semantic Web Service)-->
  <xs:attribute name="service" type="sws:ServiceType" use="required"/>
</xs:complexType>
```

6.19.3 *Response*

The *CheckRelation* response is a SPARQL response document encapsulating a Boolean value stating whether the specified concepts are related via the specified SKOS relationship. The XML below shows a fragment of the *CheckRelation* response associated with the request example above.

```
<sparql:sparql>
  <sparql:results>
    <sparql:boolean>true</sparql:boolean>
  </sparql:results>
</sparql:sparql>
```


7 References

[AAC09] Antonioletti, A., Aranda, C.B., Corcho, O., Esteban-Gutiérrez, M., Goméz-Pérez, A., Kojima, I., Lynden, S., and Pahlevi, S.M.: WS-DAI RDF(S) Realization: Introduction, Motivational Use Cases and Terminologies. 30 December 2004. Available online at: http://www.ogf.org/documents/GFD.163.pdf.

[BG04] Brickley, D., and Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation. 10 February 2004. Available online at: http://www.w3.org/TR/rdf-schema/.

[BM04] Beckett, D., and McBride, B.: RDF/XML Syntax Specification (Revised). W3C Recommendation. 10 February 2004. Available online at http://www.w3.org/TR/REC-rdf-syntax/.

[Co11] Cox, S.: Observations and Measurements – XML Implementation, Version 2.0. OGC Implementation. 22 March 2011. URL: http://www.opengeospatial.org/standards/om.

[CW11] Cornford, D., and Williams, M.: UncertML Best Practice Proposal. The Uncertainty Enabled Model Web, UncertWeb. March 2011.

[Do11] Domenico, B.: OGC Network Common Data Form (NetCDF) Core Encoding Standrad version 1.0. Candidate OpenGIS Encoding Standard. 05 April 2011.

[DS04] Dean, M., and Schreiber, G.: OWL Web Ontology Language: Reference. W3C Recommendation. 10 February 2004. Available online at: http://www.w3.org/TR/owl-ref/.

[HLW11] T. Hamre, Y. Lassoued, P. Walker, J. de Jesus, and M. Treguer. NETMAR Deliverable D6.2 – First Version of EUMIS Subsystems. May 2011.

[ISO03] ISO/TC211: ISO 19115:2003 – Geographic Information – Metadata. International Organization for Standardization, 2003.

[ISO05] ISO, 2005. ISO 19119:2005 Geographic information – Services.

[ISO06] ISO/TC211: ISO/PRF TS 19139 – Geographic information – Metadata – XML schema implementation. International Organization for Standardization, 2006.

[JBCW10] Jones, R., Bastin, L., Cornford, D., and Williams, M.: Handling and communicating uncertainty in chained geospatial Web Services. Spatial Accuracy, Leicester, UK July 2010.

[LBTR04] Lake, R., Burggraf, D. S., Trninic, M., and Rae, L.: Geography Mark-Up Language (GML) – Foundation for the Geo-Web. John Wiley. 2004.

[LLW10] Lassoued, Y., Leadbetter, A., and Walker, P.: Semantic Framework Strategy. November 2010.

[MM04] Manola, F., and Miller, E.: RDF Primer. W3C Recommendation. 10 February 2004. Available online at: http://www.w3.org/TR/rdf-primer/.

[MB09] Miles, A., and Bechhofer, S.: SKOS Simple Knowledge Organization System: Reference. W3C Recommendation. 18 August 2009. Available online at: http://www.w3.org/TR/2009/REC-skos-reference-20090818/.

[MH04] McGuinness, D., and Harmelen, F.: OWL Web Ontology Language: Overview. W3C Recommendation. 10 February 2004. URL: http://www.w3.org/TR/owl-features/.

[MGH09] Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., and Lutz, C.: OWL 2 Web Ontology Language Profiles. W3C Recommendation. 27 April 2009. Available online at: http://www.w3.org/TR/owl2-profiles/.

[NW05] Nebert, D., and Whiteside, A.: OGC Catalogue Services Specification. Open Geospatial Consortium Inc. 2005.

[Pe02] Percivall, George, ed.: OpenGIS Document 02-112, The OpenGIS Abstract Specification, Topic 12: OpenGIS Service Architecture, Version 4.3, 2002.

[PS08] Prud'hommeaux, E., and Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation. 15 January 2008. URL: http://www.w3.org/TR/rdf-sparql-query/.

[RDF04] RDF Working Group: Resources Description Framework, World Wide Web Consortium (W3C), 10 February 2004. Available at: http://www.w3.org/RDF/.

[Sc07] Schut, P.: OpenGIS Web Processing Service, Version 1.0.0. OpenGIS Standard. 08 June 2007. URL: http://www.opengeospatial.org/standards/wps.

[SGPP10] Stasch, C., Gerharz, L., Pross, B., and Pebesma, E.: Report on Integration Requierements in UncertWeb. The Uncertainty Enabled Model Web, UncertWeb. November 2010.

[Ta82] Taylor, J. R.: An Introduction to Error Analysis. University Science Books, Mill Valley CA, 1982.

[WE08] Whiteside, A. and Evans, J.D., Web Coverage Service (WCS) Implementation Standard (Version 1.1). Open Geospatial Consortium Inc., March 2008.

8 Acronyms

API	Application Programming Interface
BODC	British Oceanographic Data Centre
CF	Climate and Forecast
CMRC	Coastal and Marine Resources Centre
CSW	Catalogue Service for the Web
EO	Earth Observation
EUMIS	European Marine Information System
GML	Geography Markup Language
INTAMAP	Interoperability and Automated Mapping
ISO	International Organization for Standardization
MODEIS	Moderate Resolution Imaging Spectroradiometer
NERC	Natural Environmental Research Council
NetCDF	Network Common Data Form
NETMAR	Open service Network for Marine Environmental Data
NVS	NERC Vocabulary Server
O&M	Observations and Measurements
OGC	Open Geospatial Consortium
OWL	Web Ontology Language
PSGP	Projected Sequential Gaussian Processes
RDBMS	Relational Database Management System
RDF	Resource Description Framework
RDFS	(Also RDF-S) RDF Schema
ReST	(Also REST) Representational State Transfer
SKOS	Simple Knowledge Organisation System
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language *(Recursive acronym)*
SWS	Semantic Web Service
URI	Uniform Resource Identifier
URN	Uniform Resource Name
WCS	Web Coverage Service
WPS	Web Processing Service
XML	eXtensible Markup Language

