ebXML RegRep Profile for Web Ontology Language (OWL)

Version 2.0 Draft 5

November 23, 2009

Specification URIs:

Release Notes:

http://wxforge.wx.ll.mit.edu:8080/jira/secure/ReleaseNote.jspa?projectId=10023&styleName=Html&version=10076

This Version:

http://docs.oasis-open.org/regrep/4.0-cd3-draft1/specs/owl-profile/regrep-owl-profile.html

http://docs.oasis-open.org/regrep/4.0-cd3-draft1/specs/owl-profile/regrep-owl-profile.odt

http://docs.oasis-open.org/regrep/4.0-cd3-draft1/specs/owl-profile/regrep-owl-profile.pdf

Previous Version:

http://docs.oasis-open.org/regrep/v3.0/profiles/owl/regrep-owl-profile-v1.5.html

http://docs.oasis-open.org/regrep/v3.0/profiles/owl//regrep-owl-profile-v1.5.odt

http://docs.oasis-open.org/regrep/v3.0/profiles/owl//regrep-owl-profile-v1.5.pdf

Latest Approved Version:

http://docs.oasis-open.org/regrep/v3.0/profiles/owl/regrep-owl-profile-v1.5.html

http://docs.oasis-open.org/regrep/v3.0/profiles/owl//regrep-owl-profile-v1.5.odt

http://docs.oasis-open.org/regrep/v3.0/profiles/owl//regrep-owl-profile-v1.5.pdf

Technical Committee:

OASIS ebXML RegRep TC

Chair(s):

Kathryn Breininger, Boeing

Editor(s):

Farrukh Najmi, Wellfleet Software

Contributors:

Kathryn Breininger, Boeing

Kajal Claypool, MIT Lincoln Labs

Carl Mattocks, MetLife
Farrukh Najmi, Wellfleet Software
Oliver Newell, MIT Lincoln Labs
Nikola Stojanovic, Individual
David Webber, Individual

Related Work:

This specification replaces or supersedes:

- [specifications replaced by this standard - OASIS as well as other standards organizations]
- [specifications replaced by this standard - OASIS as well as other standards organizations]

This specification is related to:

- [specifications related to this standard - OASIS as well as other standards organizations]
- [specifications related to this standard - OASIS as well as other standards organizations]

Declared XML Namespace(s):

This following table lists the namespace prefixes defined and / or referenced by this specification.

Namespace Prefix	Namespace URI	Defining Specification
lcm	urn:oasis:names:tc:ebxml-regrep:xsd:lcm:4.0	ebXML RegRep Services and Protocols 4.0 (ebRS)
mime	http://schemas.xmlsoap.org/wsdl/mime/	WSDL namespace for WSDL MIME binding.
owl	http://www.w3.org/2002/07/owl#	The OWL namespace
owlp	urn:oasis:names:tc:ebxml-regrep:profile:webontology:2.0	The base namespace for this profile
query	urn:oasis:names:tc:ebxml-regrep:xsd:query:4.0	ebXML RegRep Services and Protocols 4.0 (ebRS)
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	The RDF namespace
rdfs	http://www.w3.org/2000/01/rdf-schema#	The RDF Schema namespace
rim	urn:oasis:names:tc:ebxml-regrep:xsd:rim:4.0	ebXML RegRep Registry Information Model 4.0 (ebRIM)
rs	urn:oasis:names:tc:ebxml-regrep:xsd:rs:4.0	ebXML RegRep Services and Protocols 4.0 (ebRS)
sparqlx	http://www.w3.org/2005/sparql-results#	The SPARQL Query Results XML Format schema as defined by [SPARQLX]
xs	http://www.w3.org/2001/XMLSchema	XML Schema [XML Schema Part 1], [XML Schema Part 2] specification
xsi	"http://www.w3.org/2001/XMLSchema-instance	W3C XML Schema specification [XML Schema Part 1], [XML Schema Part 2].

Table 1: Namespaces Used

Abstract:

This document defines the ebXML RegRep profile for publishing, management, discovery and reuse of OWL DL Ontologies.

Status:

This document is a draft specification for review, revision and approval by the OASIS ebXML RegRep TC.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at http://www.oasis-open.org/committees/regrep/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/regrep/ipr.php.

The non-normative errata page for this specification is located at http://docs.oasis-open.org/regrep/4.0-draft-1/specs/core/errata.pdf

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The names "OASIS", [insert specific trademarked names, abbreviations, etc. here] are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.

Table of Contents

1 Introduction....10

1.1 Scope....10

1.1 Use Cases for OWL Support in ebXML RegRep....10

1.1 Document Organization....11

1.1 Terminology....12

1.2 Normative References....12

1.3 Informative References....12

2 OWL Overview (Informative)....14

2.1 Semantic Web Languages upon which OWL is Layered....14

3 Publish Profile....15

3.1 Publishing OWL....15

3.1.1 Ontology Partitioning for Collaborative Development....15

3.1.2 Using RepositoryItem Canonical URL for xml:base Attribute....15

3.1.3 Import Processing....16

3.1 Validating OWL....16

3.1 Cataloging OWL....16

3.1 Updating OWL....17

3.2 Deleting OWL....17

3.3 Semantic Annotation of RegistryObjects....17

3.3.1 Types of OWL Construct References....17

3.3.1 Use Cases for Semantic Annotations....17

3.3.1 Referencing OWL Constructs in Slots....18

3.3.1 Referencing OWL Constructs in Reference Attributes....18

4 Ontology Versioning....20

4.1 Versioning Requirements....20

4.2 Types of Ontology Changes....20

4.3 Compatibility of Changes....20

4.4 Publishing Changes to an Ontology....21

4.5 Specifying Ontology Context for Queries....21

5 Discovery Profile....23

5.1 Specifying Ontology Context for Queries....23

5.2 Canonical Functions....24

5.2.1 Canonical Function: dataTypeProperties....25

5.2.1.1 Parameter Summary....25

5.2.1.2 Function Semantics....25

5.2.2 Canonical Function: equivalentClasses....26

5.2.2.1 Parameter Summary....26

5.2.2.2 Function Semantics....27

5.2.3 Canonical Function: equivalentProperties....27

5.2.3.1 Parameter Summary....27
5.2.3.2 Function Semantics....27
5.2.4 Canonical Function: instanceOf....27
5.2.4.1 Parameter Summary....27
5.2.4.2 Function Semantics....28
5.2.5 Canonical Function: instances....28
5.2.5.1 Parameter Summary....28
5.2.5.2 Function Semantics....28
5.2.6 Canonical Function: inverseProperties....28
5.2.6.1 Parameter Summary....29
5.2.6.2 Function Semantics....29
5.2.7 Canonical Function: objectProperties....29
5.2.7.1 Parameter Summary....29
5.2.7.2 Function Semantics....29
5.2.8 Canonical Function: sameAs....30
5.2.8.1 Parameter Summary....30
5.2.8.2 Function Semantics....30
5.2.9 Canonical Function: subClasses....30
5.2.9.1 Parameter Summary....30
5.2.9.2 Function Semantics....30
5.2.10 Canonical Function: subProperties....31
5.2.10.1 Parameter Summary....31
5.2.10.2 Function Semantics....31
5.2.11 Canonical Function: superClasses....31
5.2.12 Parameter Summary....31
5.2.12.1 Function Semantics....31
5.2.13 Canonical Function: superProperties....32
5.2.13.1 Parameter Summary....32
5.2.13.2 Function Semantics....32
5.2.14 Canonical Function: similarTerms....33
5.2.14.1 Parameter Summary....33
5.2.14.2 Function Semantics....33
5.3 Invoking SPARQL Queries....34
5.3.1 QueryRequest Requirements....34
5.3.2 QueryResponse Requirements....35
6 Governance Profile....36
6.1 Creation of an Ontology Register....36
6.2 Designation of Organization Roles....36
6.3 Designation of Person Roles....36
6.4 Publishing Draft Ontology Files....37
6.5 Submitting Ontologies for Review....37
6.6 Receiving Ontology Change Proposals....37
6.7 Reviewing Ontology Change Proposal....38
6.8 Creating New Version of Ontologies....38

Illustration Index

Illustration 1: Ontology Context - a Visual Example.....22

Index of Tables

Table 1: Namespaces Used.....3

Table 2: Canonical Functions Defined By This Profile.....25

1 Introduction

This chapter provides an introduction to the rest of this document.

The ebXML RegRep's repository contains electronic documents while its registry contains metadata that describes the documents in the repository. The metadata is defined using the [ebRIM] model which is quite flexible in its ability to describe the documents in the repository.

However, many applications domains require considerably richer ability to describe information content. Ontologies are emerging as a means to provide a richer more semantically expressive means to model information content. One of the driving forces for ontologies is the Semantic Web initiative [LeeHendler]. As a part of this initiative, W3C's Web Ontology Working Group defined Web Ontology Language [OWL].

Naturally, there is lot to be gained from using a standard ontology definition language, like OWL, to express richer information modeling semantics in ebXML RegRep.

1.1 Scope

This specification normatively defines the ebXML RegRep profile for Web Ontology Language (OWL) DL. More specifically, this specification normatively specifies the following:

- How OWL ontologies MAY be published to an ebXML RegRep server (Publish Profile)
- How ebXML RegRep metadata (RegistryObjects) within and ebXML RegRep server MAY reference published OWL ontology constructs (Semantic Annotation)
- How ebXML RegRep queries may discover semantically annotated RegistryObjects using published OWL ontologies (Discovery Profile)
- How ebXML RegRep queries may be used to perform SPARQL queries on the OWL repository content (Invoking SPARQL Queries)
- How published OWL ontologies may be governed using ebXML RegRep policies and registration procedures (Governance Profile)

Issue-LB: Will also be good to reference how to deal with SKOS, because OGC, Geonetwork, and MMI have ontologies already in this format. I think now SKOS is OWL-DL??

The first three items above utilize OWL ontology capabilities to improve the capabilities of ebXML RegRep while the fourth item utilizes capabilities of ebXML RegRep to provide much needed collaborative ontology authoring and change management procedures to OWL ontology developers.

This specification specifically does not define mappings from OWL constructs to the [ebRIM] model. This is a significant departure from previous versions of this specification.

1.1 Use Cases for OWL Support in ebXML RegRep

This section describes some use cases that have been considered as main motivations for adding OWL features to ebXML RegRep within this profile.

- Semantic annotation
    - ○ Allow repository content to be described by semantically annotated ebRIM metadata

    - Support the use of ontological concepts as attribute value for ebRIM objects, specifically as value of slots, association type and other RegistryObject attributes
- Semantic discovery
    - Allows discovery of objects such as datasets, services etc. based on a semantic match on a attribute value rather than a precise literal match
    - Makes use of domain and mapping ontologies to perform semantic harmonization and determine similar terms
    - Makes use of ontologies to match parent (broader) and child (narrower) terms
    - May make use of semantically annotated ebRIM metadata
- Collaborative ontology publishing and management
    - Supports ontology elements to be published by multiple organizations and individuals collaboratively
    - Allows browsing and discovering ontology elements within regrep
    - Provide governance process for managing changes to an ontologies
- Semantic mediation
    - Allows specialized clients to bi-directionally transforms data from one format to another. A specific example is mediation between OGC WFS and JMBL. For example, a client may make a WFS request to a server server that supports JMBL.
    - Allows data registered using different data standards (e.g. 19139, FGDC) to be discovered uniformly using the same query
    - Mediation is an application of semantically enhanced RegRep rather than a feature of it

In addition to above use cases, this specification aims to address the use cases identified in [ORUC].

1.1 Document Organization

The document is organized as follows:

- Chapter 1: Introduction - provides an introduction to the rest of this document
- Chapter 2: OWL Overview - provides an overview of the Web Ontology Language
- Chapter 3: Publish Profile - specifies how OWL Ontologies are published to the server
- Chapter 4: Ontology Versioning – specifies how multiple versions of ontologies are supported
- Chapter 5: Discovery Profile - specifies how discovery queries make use of OWL Ontologies available in the server
- Chapter 6: Governance Profile - specifies how OWL ontologies are governed with the server

1.1 Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in IETF RFC 2119 .

1.2 Normative References

[ebRIM] ebXML RegRep Information Model Version 4.0 http://www.oasis-open.org/committees/regrep/documents/4.0/specs/regrep-rim-4.0-cs.pdf

[ebRS] ebXML RegRep Services and Protocols 4.0 http://www.oasis-open.org/committees/regrep/documents/4.0/specs/regrep-rs-4.0-cs.pdf

[OWL] Web Ontology Language Overview, W3C Recommendation 10 February 2004 http://www.w3.org/TR/2004/REC-owl-features-20040210/

[OWL/REF] OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004 http://www.w3.org/TR/2004/REC-owl-ref-20040210/

[RDF/XML] RDF/XML Syntax Specification (Revised) W3C Recommendation 10 February 2004 http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/

[RDFS] RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation 10 February 2004 http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

[RFC 2119] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt

[SPARQL] SPARQL Query Language for RDF, W3C Recommendation 15 January 2008 http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/

[SPARQLX] SPARQL Query Results XML Format, W3C Recommendation 15 January 2008 http://www.w3.org/TR/2008/REC-rdf-sparql-XMLres-20080115/

1.3 Informative References

[LeeHendler] Berners-Lee, T., Hendler, J., Lassila, O., "The Semantic Web", Scientific American, May 2001. http://www.scientificamerican.com/article.cfm?id=the-semantic-web

[MMI] Marine Metadata Interoperability Project http://marinemetadata.org

[OWLG] OWL Web Ontology Language Guide, W3C Recommendation 10 February 2004 http://www.w3.org/TR/2004/REC-owl-guide-20040210/

[ORUC] MMI Ontology Repository Use Cases http://marinemetadata.org/community/teams/ont/ontwebservices/mmirepository/ontrepositoryuc

[OVCS] Managing Change: An Ontology Version Control System, Timothy Redmond, Michael Smith, Nick Drummond, and Tania Tudorache
http://clarkparsia.com/files/pdf/change-owled2008-eu.pdf

[OVSW] Ontology Versioning on the Semantic Web, Michel Klein, Dieter Fensel SWWS Stanford, July 30, 2001
http://www.cs.vu.nl/~mcaklein/presentations/2001-07-31-SWWS-Stanford.pdf

[ALIGN] A format for ontology alignment
http://alignapi.gforge.inria.fr/format.html

[RDFC] Resource Description Framework (RDF): Concepts and Abstract Syntax
http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[RDFP] RDF Primer
http://www.w3.org/TR/rdf-primer/

[StaabStuder] Staab, S., Studer, R., Handbook on Ontologies, Springer, 2004.
http://www.amazon.com/gp/product/3540408347

2 OWL Overview (Informative)

This chapter provides an very brief overview of the Web Ontology Language (OWL). For a more complete overview please refer to [OWL].

OWL is a semantic markup language for publishing and sharing ontologies on the World Wide Web. OWL is derived from the DAML+OIL Web Ontology Language [DAML+OIL] and builds upon the Resource Description Framework [RDF].

OWL provides three decreasingly expressive sub-languages [McGuinness, Harmelen]:

- **OWL Full** is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. It is unlikely that any reasoning software will be able to support complete reasoning for OWL Full.
- **OWL DL** supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL is so named due to its correspondence with description logics which form the formal foundation of OWL.
- **OWL Lite** supports those users primarily needing a classification hierarchy and simple constraints.

Within the scope of this document, only OWL DL constructs are considered and in the rest of the document, “OWL” is used to mean “OWL DL” unless otherwise stated.

OWL describes the structure of a domain in terms of classes and properties.

2.1 Semantic Web Languages upon which OWL is Layered

OWL is one of a set of languages defined for the Semantic Web. It occupies the Ontology layer of an architecture sometimes referred to as the Semantic Web Layer Cake. This moniker alludes to the fact that each language in the architecture sits on top of another while exposing some of the layer below is often seen of a wedding cake. OWL is situated in this architecture directly above the RDF Vocabulary Description Language: RDF Schema (RDFS) [RDFS]. RDFS is a language for defining vocabularies or models with which to describe or categorize resources in the semantic web. RDFS, in turn, sits atop the Resource Description Framework (RDF) [RDF]. RDF provides a basic data model, XML based transfer syntax, and other basic tools. The whole Semantic Web stack itself then sits atop XML technologies which are used for identification and syntax definition.

3 Publish Profile

This chapter specifies how OWL Ontologies are published to the server.

3.1 Publishing OWL

A client publishes OWL Ontologies to the server using the standard SubmitObjects protocol as defined by [ebRS].

The following additional requirements are defined for publishing OWL:

- A client MUST publish OWL constructs as a repository item associated with an ExtrinsicObject
- The repository item MUST contain an OWL document in the RDF/XML format as defined by [RDF/XML]
- The OWL repository item MUST be syntactically valid or else it MUST return a InvalidRequestException.
- The OWL repository item MAY result in a semantic inconsistency during publish. Semantic inconsistency will be dealt with in the validation and governance steps and not the publish step
- The ExtrinsicObject MUST have a mimeType attribute with value "application/rdf+xml"
- The ExtrinsicObject MUST have an objectType attribute that references the canonical ObjectType OWL as defined by this specification. The value of this attribute MUST be "urn:oasis:names:tc:ebxml-regrep:profile:webontology:ObjectType:RegistryObject:ExtrinsicObject:OWL"

This specification does not define the granularity of the submitted OWL repository item content. A single OWL repository item MAY be an entire ontology at one extreme or may be a single OWL axiom at the other extreme. More commonly a single OWL repository item SHOULD be multiple OWL axioms that share commonalities such as a single owl:Class and all the properties defined for the class.

3.1.1 Ontology Partitioning for Collaborative Development

Ontologies are typically developed by a group of individuals that are collaboratively authoring or editing different parts of the ontology. Ontologies are stores as OWL file in RDF/XML format according to this specification. In order to allow maximum concurrent development on an ontology it is suggested that an ontology be partitioned across multiple OWL files in RDF/XML format. It is suggested that such partitioning should be along the lines of a topic (subject) within the ontology. This allows different individuals to author different topic areas within the ontology with less impact to each other. One special topic within an ontology may be a "core" topic which defines some common axioms that are used in multiple topics.

3.1.2 Using RepositoryItem Canonical URL for xml:base Attribute

An OWL file in RDF/XML representation has an rdf:RDF/xml:base attribute. This value of this attribute is a URI that is typically a URL that serves the OWL file via HTTP GET protocol.

[ebRS] defines a REST interface which specifies a canonical URL that may be used to fetch a repository item via HTTP GET.

A client MAY specify the value of the rdf:RDF/xml:base attribute to be “$repositoryItemURL”. A server supporting this profile MUST replace the value of “$repositoryItemURL” for the rdf:RDF/xml:base attribute with the canonical URL for the repository item for that file.

3.1.3 Import Processing

A server MUST process import statements while persisting an OWL Ontology to the OWL repository during publish. The effect of this processing MUST be that the knowledge graph stored in the OWL repository MUST include the OWL content from the publish OWL file as well as those from OWL files that are directly or indirectly imported by the published OWL file.

Issue: Should we generate an ExtrinsicObject RepositoryItem pair for each imported document or not. ?? WSDL profile experience suggests this may not be desirable.

3.1 Validating OWL

When an Ontology is in “Approved” status it MUST be semantically valid. However, during design stage when the Ontology is in “Draft” status an Ontology MAY be semantically invalid or inconsistent.

A server supporting this profile MUST support semantic validation of OWL content at certain points in the lifecycle of OWL content. A server SHOULD NOT perform semantic validation during publishing of OWL content. Semantic validation SHOULD be deferred to the Change Review Process defined by Registration Procedure feature set of [ebRS].

The following requirements are defined for a server when validating the OWL content during the Change Review process:

- A server MUST provide a Validation Service as defined by [ebRS] for semantically validating OWL content when it is invoked. Additional details of validating OWL content will be presented in the Governance Profile chapter.

3.1 Cataloging OWL

A server MUST provide a cataloging service for OWL content as defined by [ebRS]. The cataloging service MUST minimally catalog the following OWL content as described below:

- If the rdf:RDF/xml:base attribute value is “$repositoryItemURL” then the server MUST replace that value with the canonical URL for the repository item for that OWL content
- //owl:Ontology/owl:versionInfo elements content value MUST be cataloged as a value for the /rim:VersionInfo/@userVersionName attribute value on the ExtrinsicObject for the OWL repository item
- //owl:Ontology/owl:imports/@rdf:resource attribute values MUST be cataloged as a value for a multi-valued canonical slot with name “urn:oasis:names:tc:ebxml-regrep:profile:webontology:2.0:imports” on the ExtrinsicObject for the OWL repository item

- //owl:Ontology/rdfs:comment element MUST be cataloged as the value of the Description/LocalizedString/@value attribute of the on the ExtrinsicObject for the OWL repository item.
    - If the rdfs:comment has an xml:lang attribute specified then the Description/LocalizedString/@locale attribute MUST have the value of that attribute.
    - If the rdfs:comment does not have an xml:lang attribute specified then Description/LocalizedString/@locale attribute MUST NOT be specified

Issue: how to determine which OWL constructs were published in repository item for which ExtrinsicObject?? Is this a requirement?? Once OWL content is published to OWL repo how do we keep track of units of submission for access control and governance purposes??

Issue: How do ExtrisicObjects for OWL get linked to reflected imports??

3.1 Updating OWL

A client updates OWL Ontologies using either the standard SubmitObjects or UpdateObjects protocol as defined by [ebRS].

3.2 Deleting OWL

A client deletes OWL Ontologies using the standard RemoveObjects protocol as defined by [ebRS].

3.3 Semantic Annotation of RegistryObjects

This section specifies how attribute values within a RegistryObject may reference an OWL construct. An [ebRIM] RegistryObject may reference various types of OWL constructs by using the fully-qualified id of the OWL construct as the value of an attribute within a class defined by [ebRIM].

3.3.1 Types of OWL Construct References

In general, any RDF resource MAY be referenced via semantic annotations in [ebRIM] RegistryObjects. Typically however, the following types of OWL constructs are referenced via semantic annotations in [ebRIM] RegistryObjects:

- owl:Class
- owl:Individual
- owl:ObjectProperty
- owl:DatatypeProperty

3.3.1 Use Cases for Semantic Annotations

Some use cases for semantic annotation of RegistryObjects are as follows:

- Referencing OWL constructs in Slots
- Referencing OWL constructs in Associations as value of type attribute

- Referencing OWL constructs in EmailAddress as value of type attribute
- Referencing OWL constructs in PostalAddress as value of type attribute
- Referencing OWL constructs in TelephoneNumber as value of type attribute
- Referencing OWL constructs in external Classifications as value of nodeRepresentation attribute

The [ebRIM] specification provides a detailed description for the classes and attributes mentioned above. Among the use cases above all but the first use case can be generalized to the use case of referencing of OWL constructs in reference attributes of an [ebRIM] class.

3.3.1 Referencing OWL Constructs in Slots

An [ebRIM] Slot MAY reference a supported OWL construct as follows:

- The rim:Slot/rim:ValueList/rim:ValueListItem/@xsi:type attribute value MUST be rim:StringValueType
- The content of the rim:Slot/rim:ValueList/rim:ValueListItem/rim:Value element of the Slot MUST be the fully-qualified id of the OWL construct
- The rim:Slot/@dataType attribute value of the Slot MUST be defined to have a value “urn:oasis:names:tc:ebxml-regrep:profile:webontology:2.0:resourceReference”

The following example shows a Person object with a Slot named “foodPreference” that references the owl:Class for VegetarianPizza.

```
<Person id="urn:acme:person:Danyal" ...>
  …
  <rim:Slot name="foodPreference"
    dataType="urn:oasis:names:tc:ebxml-
regrep:profile:webontology:2.0:resourceReference">
    <rim:ValueList>
      <rim:ValueListItem xsi:type="rim:StringValueType">
        <rim:Value>http://www.co-
ode.org/ontologies/pizza/pizza.owl#VegetarianPizza</rim:Value>
      </rim:ValueListItem>
    </rim:ValueList>
  </rim:Slot>
</Person>
```

3.3.1 Referencing OWL Constructs in Reference Attributes

A RegistryObject attribute MAY reference a supported OWL construct as follows:

- The value of the attribute MUST be the fully-qualified id of the OWL construct

The following example shows an Association between two Person objects where the type attribute references a “hasBrother” ObjectProperty.

```
<Association …
  sourceObject=”urn:acme:person:Danyal”
  targetObject=”urn:acme:person:Omar”
  type=”http://www.mindswap.org/ontologies/family.owl#hasBrother”/>
```

Issue: No way to distinguish a normal regrep reference from reference to an OWL resource?? Should we use a prefix hack like:

type=”owlref:**http://www.mindswap.org/ontologies/family.owl#hasBrother**”

4 Ontology Versioning

This chapter defines how ontologies evolve over time within a server and how clients may use different versions of ontologies as a basis for semantic searches and SPARQL queries. The ontology versioning features of this specification rely largely upon the versioning features specified in [ebRS].

4.1 Versioning Requirements

The ontology versioning features of this specification are designed to support the following core requirements:

- Different versions of an ontology may be deployed within a RegRep server at the same time
- Different client applications may be using different versions of the same ontology at the same time
- An application must be able to perform SPARQL queries and queries that use owl canonical functions within the context of a specific versions of a specific set of ontologies. Other version of the ontologies and other ontologies MUST NOT have any effect on such a scope constrained semantic search

4.2 Types of Ontology Changes

An ontology evolves over time as multiple authors collaboratively update its constructs. Evolution of ontologies consists of the following types of change:

- Create - Creation of new classes, properties and individuals
- Update - Update of existing classes, properties and individuals
- Delete - Deletion of existing classes, properties and individuals

4.3 Compatibility of Changes

A change to an ontology may further be categorized by how it impacts instance data (individuals) as follows:

- Backward compatible – Individuals that are created for old version of ontology are compatible with new version of ontology. Client applications compatible with old version of ontology are compatible with new version of ontology
- Forward compatible – Individuals that are created for new version of ontology are compatible with old version of ontology. Client applications compatible with new version of ontology are compatible with old version of ontology
- Incompatible – The versions of the ontology are incompatible with each other

“Create” changes can often be done in a backward compatible manner by defining the additional constructs in a new OWL file and publishing it to the server with its own unique version-specific rdf/RDF/@xml:base attribute value.

“Update” and “Delete” changes are usually incompatible. They typically require publishing a modified version of the original OWL file to the server with specific constructs updated or deleted. Such a modified version of the original OWL file should have its own unique version-specific rdf/RDF/@xml:base attribute value.

4.4 Publishing Changes to an Ontology

To change an ontology, a client typically publishes a modified OWL file as described in the Publish Profile. When a modified OWL file is published it MAY update or replace the current version without creating a new version or it MAY create a new version while leaving the current version unchanged. In the latter case, when a new version of an OWL file is published, the new version is associated with the version it supersedes via a "Supersedes" Association as required by [ebRS] versioning feature. Thus, in general versions of an OWL file may be represented by a tree structure where there may be many parallel version streams.

A client controls the behavior of whether to update (replace) the existing version or create new version via the mode attribute of SubmitObjectsRequest as described by [ebRS].

- When the mode attribute value is "CreateOrUpdate" then submitting a modified version of an existing OWL file updates (replaces) the existing version

- When the mode attribute value is "CreateOrVersion" then submitting a modified version of an existing OWL file creates a new version in Validating OWL.

In both cases the id and lid of the ExtrinsicObject MUST match that for an existing OWL file in the server. In both cases, a server MUST report a ValidationException during the validation of the OWL file if any inconsistency is introduced by the change as describe

A deployment may further control the review and approval of changes to an ontology as described in Governance Profile.

4.5 Specifying Ontology Context for Queries

A client MAY specify a specific ontology context when invoking ebRS query that uses SPARQL query syntax or that uses a canonical owl function defined by this profile. The ontology context consists of a specific set of OWL files. The OWL files are specified using the id of the ExtrinsicObject associated with them. A server evaluates a SPARQL query or a canonical owl function in an ontology context that has only the specified versions of the specified OWL files loaded. Other versions and other ontologies have no impact on the evaluation of the SPARQL query or a canonical owl function within the processing of a QueryRequest.

The details of how the ontology context is defined and how it impacts the Query protocol is defined in Discovery Profile.

Illustration 1 Shows a visual example of the version graphs of two ontologies V and W and an ontology context consisting of version v3 of ontology V and version w2 of ontology W. All OWL functions and SPARQL queries for this context will be evaluated within the context of the axioms defined for ontology versions v3 and w2 only.

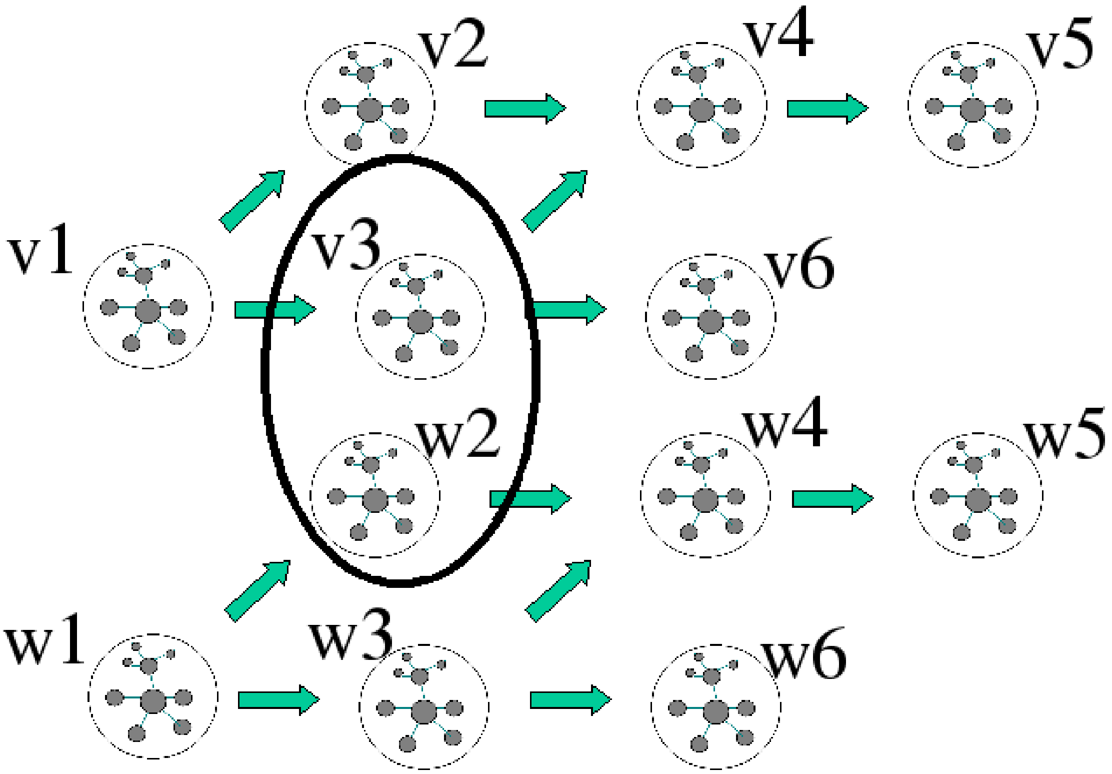

Illustration 1: Ontology Context - a Visual Example

5 Discovery Profile

This chapter specifies how discovery queries make use of OWL Ontologies available in the server. This specification provides the following new feature for supporting semantic discovery using OWL ontologies:

- Specifies how a client may specify an ontology context when invoking the Query protocol as defined by [ebRS]
- Specifies a set of OWL-specific Canonical Function that may be used in Query protocol as defined by [ebRS]
- Specifies how SPARQL queries may be invoked using the AdhocQuery canonical query defined by [ebRS]

5.1 Specifying Ontology Context for Queries

A client MAY specify a specific ontology context when invoking ebRS query that uses SPARQL query syntax or that uses a canonical owl function defined by this profile. The ontology context consists of a specific set of OWL files. The OWL files are specified using the id of the ExtrinsicObject associated with them.

A client MAY specify the ontology context for a query using a canonical slot as a child of the QueryRequest element as follows:

- The Slot/@name MUST be urn:oasis:names:tc:ebxml-regrep:profile:webontology:2.0:QueryRequest:ontologyContext
- The Slot/@dataType MUST be urn:oasis:names:tc:ebxml-regrep:DataType:String
- The Slot/@collectionType MUST be urn:oasis:names:tc:ebxml-regrep:CollectionType:SortedSet
- The Slot/ValueList MUST contain ValueListItems whose xsi:type is StringValueType
- Each Slot/ValueList/ValueListItem/Value element MUST specify the id of an ExtrinsicObject whose repository item is an OWL file as described in Publish Profile

The following example shows how a client MAY specify an ontology context for a QueryRequest:

```
<query:QueryRequest maxResults="-1" startIndex="0" ...>
  <rim:Slot
    name="urn:oasis:names:tc:ebxml-
regrep:profile:webontology:2.0:QueryRequest:ontologyContext"
    dataType="urn:oasis:names:tc:ebxml-regrep:DataType:String"
    collectionType="urn:oasis:names:tc:ebxml-regrep:CollectionType:SortedSet"
>

    <rim:ValueList>
      <rim:ValueListItem xsi:type="StringValueType">
        <rim:Value>urn:example:ontology:V:3.0</rim:Value>
        <rim:Value>urn:example:ontology:W:2.0</rim:Value>
      </rim:ValueListItem>
    </rim:ValueList>
  </rim:Slot>
  ...
  <query:Query queryDefinition="urn:example:query:SomeExample">
    ...
  </query:Query>
</query:QueryRequest>
```

A client SHOULD make sure that the specified ontology context results in a knowledge base that is semantically consistent and is supported by the reasoner used by the server implementation.

When a client does not specify an ontology context, a server MUST define a default ontology context such that it results in a knowledge base that is semantically consistent and is supported by the reasoner used by the server implementation. This specification does not define the manner in which the default ontology context is defined.

A server MUST perform all evaluation of SPARQL querys and canonical owl functions defined by this profile as if the only ontologies defined are the specific versions of specific ontologies defined by the ontology context for the QueryRequest. Other versions and other ontologies MUST NOT have any impact on the evaluation of SPARQL querys and canonical owl functions.

5.2 Canonical Functions

The [ebRS] specification defines a set of canonical functions, their parameters, their semantics and how they may be used within queries and query parameters. [ebRS] Will define the Function concept in CD4 per issue 119.

This profile defines several additional canonical functions that MUST be supported by a server implementing this profile. Table 2 summarizes the functions defined by this profile. Subsequent sections specify them in detail.

The canonical functions defined in this section, along with semantic annotation of RegistryObjects described earlier, together enable semantic search capability in standard [ebRS] queries. A client MAY use these functions within the static part of a query expression or within the value of a parameter to a parameterized query. A server MUST process the functions according to their behavior as specified in this section. The function processing model is specified by [ebRS].

A client MUST use the “owlp:” namespace prefix when using a canonical function defined by this profile.

Function Name	Semantics
owlp:dataTypeProperties	Returns the datatype properties for specified class
owlp:equivalentClasses	Returns all equivalent classes for the specified class
owlp:equivalentProperties	Returns all equivalent classes for the specified property.
owlp:instanceOf	Returns all individuals that are instances of specified class
owlp:instances	Returns all Individuals that are instances of specified class
owlp:inverseProperty	Returns the inverse property for the specified property
owlp:objectProperties	Returns the object properties for specified class
owlp:sameAs	Returns all individuals that are same as the specified individual
owlp:subClasses	Returns all sub-classes of the specified class
owlp:subProperties	Returns all sub-properties of the specified property
owlp:superClasses	Returns all super classes of the specified class
owlp:superProperties	Returns all super properties of the specified property
owlp:similarTerms	Returns all terms deemed to be semantically similar to specified term

Table 2: Canonical Functions Defined By This Profile

5.2.1 Canonical Function: dataTypeProperties

This canonical function takes an OWL class's id as parameter and returns all dataType properties of the specified class.

5.2.1.1 Parameter Summary

Parameter	Description	Data Type
classId	The value of this parameter SHOULD be the id of an OWL class	string
direct	If true, restrict the properties returned to those directly associated with this class	boolean
delimiter	The value of this parameter specifies the delimiter string to be used as separator between the tokens representing the ids matched by the function	string

5.2.1.2 Function Semantics

- The server MUST return a string if the query is processed without any exceptions
- The string MAY be empty if no class is found with specified id or if no data properties are found for class matching specified id
- The string MUST consist of a set of data property ids separated by the appropriate delimiter character when any data properties are found for class matching specified id

The following example shows an EJBQL query that matches RegistryObjects that have a slot whose dataType attribute matches any one of the dataType properties of the specified OWL class:

```
<query:QueryRequest ...>
  <rim:Slot
    name="urn:oasis:names:tc:ebxml-
regrep:profile:webontology:2.0:QueryRequest:ontologyContext"
    dataType="urn:oasis:names:tc:ebxml-regrep:DataType:String"
    collectionType="urn:oasis:names:tc:ebxml-regrep:CollectionType:SortedSet"
>

    <rim:ValueList>
      <rim:ValueListItem xsi:type="StringValueType">
        <rim:Value>urn:example:ontology:camera.owl:1.0</rim:Value>
      </rim:ValueListItem>
    </rim:ValueList>
  </rim:Slot>
  ...
  <query:Query queryDefinition="urn:oasis:names:tc:ebxml-
regrep:query:AdhocQuery">

    <rim:Slot name="queryLanguage">
      <rim:ValueList>
        <rim:ValueListItem xsi:type="StringValueType"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
          <rim:Value>urn:oasis:names:tc:ebxml-
regrep:QueryLanguage:EJBQL</rim:Value>
        </rim:ValueListItem>
      </rim:ValueList>
    </rim:Slot>

    <rim:Slot name="queryExpression">
      <rim:ValueList>
        <rim:ValueListItem xsi:type="StringValueType"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
          <rim:Value>#@SELECT Object(eo) FROM ExtrinsicObjectType eo LEFT
OUTER JOIN eo.slot semref_slot WHERE (semref_slot.dataType IN (@#
owlp:dataTypeProperties(\"http://www.xfront.com/owl/ontologies/camera/#Range\"
, true, \",\") #@) )@#</rim:Value>
        </rim:ValueListItem>
      </rim:ValueList>
    </rim:Slot>

  </query:Query>
</query:QueryRequest>
```

5.2.2 Canonical Function: equivalentClasses

This canonical function takes an OWL class's id as parameter and returns all classes that are equivalent to the specified class.

5.2.2.1 Parameter Summary

Parameter	Description	Data Type
classId	The value of this parameter SHOULD be the id of an OWL class	string
delimiter	The value of this parameter specifies the delimiter string to be used as	string

	separator between the tokens representing the ids matched by the function	

5.2.2.2 Function Semantics

- The server MUST return a string if the query is processed without any exceptions
- The string MAY be empty if no class is found with specified id or if no equivalent classes are found for class matching specified id
- The string MUST consist of a set of equivalent class ids separated by the appropriate delimiter character when any equivalent classes are found for class matching specified id

5.2.3 Canonical Function: equivalentProperties

This canonical function takes an OWL property's id as parameter and returns all equivalent properties for the specified property.

5.2.3.1 Parameter Summary

Parameter	Description	Data Type
propertyId	The value of this parameter SHOULD be the id of an OWL property	string
delimiter	The value of this parameter specifies the delimiter string to be used as separator between the tokens representing the ids matched by the function	string

5.2.3.2 Function Semantics

- The server MUST return a string if the query is processed without any exceptions
- The string MAY be empty if no property is found with specified id or if no equivalent properties are found for property matching specified id
- The string MUST consist of a set of equivalent property ids separated by the appropriate delimiter character when any equivalent properties are found for property matching specified id

5.2.4 Canonical Function: instanceOf

This canonical function takes an OWL Individual's id as parameter and returns all classes of which the specified Individual is an instance of.

5.2.4.1 Parameter Summary

Parameter	Description	Data Type
individualId	The value of this parameter SHOULD be the id of an OWL Individual	string

direct	If true, only consider the direct types of this individual, ignoring the super-classes of the stated types.	boolean
delimiter	The value of this parameter specifies the delimiter string to be used as separator between the tokens representing the ids matched by the function	string

5.2.4.2 Function Semantics

- The server MUST return a string if the query is processed without any exceptions
- The string MAY be empty if no Individual is found with specified id or if no classes are found that are types for the Individual matching specified id
- The string MUST consist of a set of class ids separated by the appropriate delimiter character when any classes are found that are a type for the Individual matching specified id

5.2.5 Canonical Function: instances

This canonical function takes an OWL class's id as parameter and returns all individuals that are instances of the specified class.

5.2.5.1 Parameter Summary

Parameter	Description	Data Type
classId	The value of this parameter SHOULD be the id of an OWL class	string
direct	If true, only direct instances are matched (i.e. instances of sub-classes of this class are not matched)	boolean
delimiter	The value of this parameter specifies the delimiter string to be used as separator between the tokens representing the ids matched by the function	string

5.2.5.2 Function Semantics

- The server MUST return a string if the query is processed without any exceptions
- The string MAY be empty if no class is found with specified id or if no individuals are found that are an instance of the class matching specified id
- The string MUST consist of a set of individuals' ids separated by the appropriate delimiter character when any individuals are found that are an instance of the class matching specified id

5.2.6 Canonical Function: inverseProperties

This canonical function takes an OWL property's id as parameter and returns the inverse properties (if any) for the specified property.

5.2.6.1 Parameter Summary

Parameter	Description	Data Type
propertyId	The value of this parameter SHOULD be the id of an OWL property	string
delimiter	The value of this parameter specifies the delimiter string to be used as separator between the tokens representing the ids matched by the function	string

5.2.6.2 Function Semantics

- The server MUST return a string if the query is processed without any exceptions
- The string MAY be empty if no property is found with specified id or if no inverse properties are found for property matching specified id
- The string MUST consist of a set of inverse property ids separated by the appropriate delimiter character when any inverse property is found for property matching specified id

5.2.7 Canonical Function: objectProperties

This canonical function takes an OWL class's id as parameter and returns all objects properties of the specified class.

5.2.7.1 Parameter Summary

Parameter	Description	Data Type
classId	The value of this parameter SHOULD be the id of an OWL class	string
direct	If true, restrict the properties returned to those directly associated with this class.	boolean
delimiter	The value of this parameter specifies the delimiter string to be used as separator between the tokens representing the ids matched by the function	string

5.2.7.2 Function Semantics

- The server MUST return a string if the query is processed without any exceptions
- The string MAY be empty if no class is found with specified id or if no object properties are found for class matching specified id
- The string MUST consist of a set of object property ids separated by the appropriate delimiter character when any object properties are found for class matching specified id

5.2.8 Canonical Function: sameAs

This canonical function takes an RDF resource id as parameter and returns all resources that are same as the specified resource.

5.2.8.1 Parameter Summary

Parameter	Description	Data Type
resourceId	The value of this parameter SHOULD be the id of an RDF resource	string
delimiter	The value of this parameter specifies the delimiter string to be used as separator between the tokens representing the ids matched by the function	string

5.2.8.2 Function Semantics

- The server MUST return a string if the query is processed without any exceptions
- The string MAY be empty if no RDF resource is found with specified id or if no RDF resources are found to be same as the specified resource
- The string MUST consist of a set of resources ids separated by the appropriate delimiter character when any resources are found to be same as the resource matching specified id

5.2.9 Canonical Function: subClasses

This canonical function takes an OWL class's id as parameter and returns all sub-classes (children, grandchildren, etc.) of the specified class.

5.2.9.1 Parameter Summary

Parameter	Description	Data Type
classId	The value of this parameter SHOULD be the id of an OWL class	string
direct	If true, only match the direct sub-classes of the specified class	boolean
delimiter	The value of this parameter specifies the delimiter string to be used as separator between the tokens representing the ids matched by the function	string

5.2.9.2 Function Semantics

- The server MUST return a string if the query is processed without any exceptions
- The string MAY be empty if no class is found with specified id or if no sub-classes are found for class matching specified id
- The string MUST consist of a set of sub-class ids separated by the appropriate delimiter character when any sub-classes are found for class matching specified id

5.2.10 Canonical Function: subProperties

This canonical function takes an OWL property's id as parameter and returns all sub-properties (children , grandchildren etc.) of the specified property.

5.2.10.1 Parameter Summary

Parameter	Description	Data Type
propertyId	The value of this parameter SHOULD be the id of an OWL property	string
direct	If true, only match the immediate sub-properties in the property hierarchy	boolean
delimiter	The value of this parameter specifies the delimiter string to be used as separator between the tokens representing the ids matched by the function	string

5.2.10.2 Function Semantics

- The server MUST return a string if the query is processed without any exceptions
- The string MAY be empty if no property is found with specified id or if no sub-properties are found for property matching specified id
- The string MUST consist of a set of sub-property ids separated by the appropriate delimiter character when any sub-properties are found for property matching specified id

5.2.11 Canonical Function: superClasses

This canonical function takes an OWL class's id as parameter and returns all super classes of the specified class.

5.2.12 Parameter Summary

Parameter	Description	Data Type
classId	The value of this parameter SHOULD be the id of an OWL class	string
direct	If true, only match the direct sub-classes of the specified class	boolean
delimiter	The value of this parameter specifies the delimiter string to be used as separator between the tokens representing the ids matched by the function	string

5.2.12.1 Function Semantics

- The server MUST return a string if the query is processed without any exceptions

- The string MAY be empty if no class is found with specified id or if no ancestor classes are found for class matching specified id
- The string MUST consist of a set of ancestor class ids separated by the appropriate delimiter character when any ancestor classes are found for class matching specified id

5.2.13 Canonical Function: superProperties

This canonical function takes an OWL property's id as parameter and returns all ancestor properties (parent, grandparent, etc.) of the specified property.

5.2.13.1 Parameter Summary

Parameter	Description	Data Type
propertyId	The value of this parameter SHOULD be the id of an OWL property	string
direct	If true, only match the immediate super-properties in the property hierarchy	boolean
delimiter	The value of this parameter specifies the delimiter string to be used as separator between the tokens representing the ids matched by the function	string

5.2.13.2 Function Semantics

- The server MUST return a string if the query is processed without any exceptions
- The string MAY be empty if no property is found with specified id or if no ancestor properties are found for property matching specified id
- The string MUST consist of a set of ancestor property ids separated by the appropriate delimiter character when any ancestor classes are found for class matching specified id

The following example shows the use of the function in the canonical FindAssociations query defined by [ebRS]. The query MUST match all Associations where the sourceObject is “urn:acme:Person:Danyal” and the associationType value references the id of an ancestor property for the “http://www.mindswap.org/ontologies/family.owl#hasBrother” property. For example, this query would match Associations where the type attribute value is “http://www.mindswap.org/ontologies/family.owl#hasSibling”

```
<query:QueryRequest ...>
  ...
  <query:Query queryDefinition="urn:oasis:names:tc:ebxml-
regrep:query:FindAssociations">
    <rim:Slot name="sourceObjectId">
      <rim:ValueList>
        <rim:ValueListItem xsi:type="StringValueType">
          <rim:Value>urn:acme:Person:Danyal</rim:Value>
        </rim:ValueListItem>
      </rim:ValueList>
    </rim:Slot>
    <rim:Slot name="associationType">
      <rim:ValueList>
```

```
        <rim:ValueListItem xsi:type="StringValueType">
          <rim:Value>owlp:superProperties(“http://www.mindswap.org/ontologies/
family.owl#hasBrother”, true, “,”)#@@#</rim:Value>
        </rim:ValueListItem>
      </rim:ValueList>
    </rim:Slot>
  </query:Query>
</query:QueryRequest>
```

5.2.14 Canonical Function: similarTerms

This canonical function takes a term specified as a string parameter and a threshold specified as a float parameter and returns all terms that are semantically similar to the specified term where the level of similarity is greater than or equal to the specified threshold of similarity.

5.2.14.1 Parameter Summary

Parameter	Description	Data Type
term	The value of this parameter may be an arbitrary term or concept (e.g. temperature)	string
threshold	A value in the range of 0.0 and 1.0 that signifies the minimum level of similarity for matching terms. A value of 0.0 indicates no similarity while a value of 1.0 indicates an exact match or perfect similarity	float
delimiter	The value of this parameter specifies the delimiter string to be used as separator between the tokens representing the ids matched by the function	string

5.2.14.2 Function Semantics

- The server MUST return a string if the query is processed without any exceptions
- The string MAY be empty if no similar terms are found for specified term
- The string MUST consist of a set of similar terms separated by the appropriate delimiter character when any term is found to be similar to specified term with a similarity level greater than or equal to the value specified by the threshold parameter

This specification does not define how terms are defined to be similar. A server is free to choose any means to support the semantics of this function. For example, a server MAY use a mapping ontology as defined in [ALIGN].

The following example shows how a client MAY use the similarTerms function when invoking a stored query to discover datasets to match all datasets that have the specified dataset field or any other dataset field that is similar to the specified field name:

```
<query:QueryRequest ...>
  ...
  <query:Query queryDefinition="urn:example:DatasetDiscoveryQuery">
    <rim:Slot name="field">
      <rim:ValueList>
        <rim:ValueListItem xsi:type="StringValueType">
          <rim:Value>owlp:similarTerms(\”temperature\”, 0.8)#@@#</rim:Value>
        </rim:ValueListItem>
      </rim:ValueList>
    </rim:Slot>
  </query:Query>
</query:QueryRequest>
```

5.3 Invoking SPARQL Queries

A server supporting this profile MUST support the invocation of SPARQL queries as defined by [SPARQL] against the OWL repository content as described in this section.

A client MAY submit an ad hoc SPARQL query to a server supporting this profile using the standard Query protocol as defined by [ebRS].

5.3.1 QueryRequest Requirements

A client MAY invoke a SPARQL query to the server using a QueryRequest. The following additional requirements are defined for a client to invoke a QueryRequest for a SPARQL query:

- The canonical AdhocQuery MUST be used within the query:Query
  - This implies that the query:Query/@queryDefinition MUST be "urn:oasis:names:tc:ebxml-regrep:query:AdhocQuery"
- The queryLanguage query parameter MUST have a value of "urn:oasis:names:tc:ebxml-regrep:QueryLanguage:SPARQL"
- The queryExpression query parameter MUST be a valid SPARQL query

The following example shows SPARQL query expression within a QueryRequest:

```
<query:QueryRequest ...>
  ...
  <query:Query queryDefinition="urn:oasis:names:tc:ebxml-
regrep:query:AdhocQuery">
    <rim:Slot name="queryLanguage">
      <rim:ValueList>
        <rim:ValueListItem xsi:type="StringValueType">
          <rim:Value>urn:oasis:names:tc:ebxml-
regrep:QueryLanguage:SPARQL</rim:Value>
        </rim:ValueListItem>
      </rim:ValueList>
    </rim:Slot>
    <rim:Slot name="queryExpression">
      <rim:ValueList>
        <rim:ValueListItem xsi:type="StringValueType">
          <rim:Value>
SELECT ?givenName
WHERE
  { ?y  <http://www.w3.org/2001/vcard-rdf/3.0#Family>  "Smith" .
    ?y  <http://www.w3.org/2001/vcard-rdf/3.0#Given>  ?givenName .
  }
          </rim:Value>
        </rim:ValueListItem>
      </rim:ValueList>
    </rim:Slot>
  </query:Query>
</query:QueryRequest>
```

5.3.2 QueryResponse Requirements

A server MUST process a SPARQL query and return its response within a QueryResponse. The following additional requirements are defined for a server to return a QueryResponse for a SPARQL query:

- A server MUST process the SPARQL query according to [SPARQL] within the context of the OWL content published within its repository. Specifically a server SHOULD NOT need to query its Registry metadata to process the SPARQL query
- A server MUST return the SPARQL response as a sparqlx:sparql element in the SPARQL Query Results XML Format as specified by [SPARQLX]
- The sparqlx:sparql MUST be the child element of a rim:Slot/rim:ValueList/rim:ValueListItem of type rim:AnyValueType within a Slot with name “urn:oasis:names:tc:ebxml-regrep:profile:webontology:2.0:sparqlResponse” and a dataType of “urn:oasis:names:tc:ebxml-regrep:profile:webontology:2.0:sparql” within the query:QueryResponse element
- The QueryResponse element SHOULD NOT have a query:RegistryObjectsList child element

The following example shows SPARQL response within a QueryResponse:

```
<query:QueryResponse ...>
  ...
  <rim:Slot
    name="urn:oasis:names:tc:ebxml-
regrep:profile:webontology:2.0:sparqlResponse"
    dataType="urn:oasis:names:tc:ebxml-regrep:profile:webontology:2.0:sparql">
    <rim:ValueList>
      <rim:ValueListItem xsi:type="rim:AnyValueType">
        <sparqlx:sparql>
          ...
        <sparqlx:sparql>
      </rim:ValueListItem>
    </rim:ValueList>
  </rim:Slot>

</query:QueryResponse>
```

6 Governance Profile

This chapter specifies how OWL Ontologies are governed within ebXML RegRep. The governance of ontologies within ebXML RegRep are based upon the using the standard Registration Procedures feature set defined by [ebRS]. A brief description of key governance concepts and activities are described here. The reader should consult the Registration Procedures feature set defined by [ebRS] for a detailed understanding and specification.

6.1 Creation of an Ontology Register

An ontology Register represents an ontology context as described by [ORUC]. Its members consists of ontology files that share a common context with respect to content, usage and governance policies. An ontology register MUST be created within a server by an organization representing a community collaborating on development of ontologies . The organization that creates the Register has the role of RegisterOwner for the Register.

The ontology Register upon creation MUST have a status of “Draft”. Changes to a Register's status trigger different work flow within the governance process for that Register.

6.2 Designation of Organization Roles

Once a register has been created, the RegisterOwner organization MUST assign various governance roles to other organizations within the community as defined by Registration Procedures feature set defined by [ebRS]. Here is a summary of these organization roles:

- SubmittingOrganization – MUST be assigned to all Organizations who are authorized to submit and change ontologies within the register
- RegisterManager – MUST be assigned to one Organizations that is authorized to receive ontology change proposals from SubmittingOrganizations and perform acceptance checks
- ControlBody – MUST be assigned to one Organizations that is authorized to perform detailed review of ontology change proposals from SubmittingOrganizations and to accept or reject each proposed change

The RegisterOwner organization MAY choose to assign one or more of above roles to itself.

6.3 Designation of Person Roles

Once an organization has been assigned a governance role for a Register it MUST assign various governance roles to persons affiliated with that organization as defined by Registration Procedures feature set defined by [ebRS]. Here is a summary of these person roles:

- ChangeProposalSubmitter - The SubmittingOrganization MUST assign this role to all persons within the organization who are authorized to submit and update ontologies within the Register
- ChangeProposalReceiver - The RegisterManager MUST assign this role to all persons or services within the organization that are authorized to receive changes to ontologies within the register and perform basic acceptance checks on the submitted changes

- ChangeProposalReviewer - The ControlBody MUST assign this role to all persons within the organization who are authorized to review and approve / reject changes to ontologies within the register

6.4 Publishing Draft Ontology Files

Once an ontology Register has been created and organization and person roles have been assigned authorized ontology developers MAY begin publishing ontology files to the server as described in Publish Profile. Once an ontology file has been published to the server the submitter MAY add it as a members of an ontology Register that has a status of “Draft”. This includes a newly created Register or a new version of an existing Register.

Throughout the ontology development phase, the ontology Register version stays in a “Draft” status. During this time, its ChangeProposalSubmitter MAY freely make changes to the ontology files that are its members as described in Publish Profile.

The default Register notification policy defined by [ebRS] requires that when members of a “Draft” Register are changed the server MUST deliver a notification to all subjects that have the role of ChangeProposalSubmitter for that Register. This is analogous to a commit email that is typical in software development project teams.

6.5 Submitting Ontologies for Review

When the ontology development cycle is complete the ChangeProposalSubmitter MAY submit the changes within the “Draft” Register for review and approval by changing the status to “Proposed” as described by the Registration Procedures feature in [ebRS].

The default Register notification policy defined by [ebRS] requires that when a Register status is changed to “Proposed”, the server MUST deliver a notification to all persons or services that have the role of **ChangeProposalReceiver** or ChangeProposalSubmitter for that Register.

Setting a Register status is changed to “Proposed” and subsequent notification initiates the change proposal receiving and acceptance / rejection activity.

6.6 Receiving Ontology Change Proposals

When a ChangeProposalReceiver is notified of the submission of a ontology changes in an ontology Register it MAY take on the activity of performing acceptance review for the proposed changes by setting the status of the Register to “UnderAcceptanceReview”.

The default Register notification policy defined by [ebRS] requires that when a Register status is changed to “UnderAcceptanceReview”, the server MUST deliver a notification to all ChangeProposalReceiver's. This lets other ChangeProposalReceiver's know that a peer has begun working on the activity and they need not work on it.

The ChangeProposalReceiver MAY then accept (set status to “Accepted”) or reject (set status to “Rejected) the change proposal in it entirety as described in Registration Procedures feature set of [ebRS].

The ChangeProposalReceiver MAY NOT be a person and instead MAY be an automated service that implements a NotificationListener interface.

The default Register notification policy defined by [ebRS] requires that when a Register status is changed to “Accepted”, the server MUST deliver a notification to all ChangeProposalReviewers. This initiates the change proposal review activity.

6.7 Reviewing Ontology Change Proposal

When a ChangeProposalReviewer is notified that ontology changes in an ontology Register have been accepted for review, it MAY take on the activity of performing detailed review of the proposed changes by setting the status of the Register to “UnderReview”.

The default Register notification policy defined by [ebRS] requires that when a Register status is changed to “UnderReview”, the server MUST deliver a notification to all ChangeProposalReviewer's. This lets other ChangeProposalReviewer's know that a peer has begun working on the activity and they need not work on it.

The ChangeProposalReviewer MUST then perform a detailed review of the proposed changes as described by [ebRS]. The following OWL specific requirements are defined for the review process:

Issue: Need OWL specific review requirements (if any) here??

During the review the ChangeProposalReviewer MAY then approve (set status to “Approved”) or reject (set status to “Rejected) the change proposal in it entirety or at the granularitry of individual changes as described in Registration Procedures feature set of [ebRS].

The default Register notification policy defined by [ebRS] requires that when a Register status is changed to “Approved”, the server MUST deliver a notification to all ChangeProposalSubmitters.

6.8 Creating New Version of Ontologies

Once a specific version of an ontology Register has been reviewed and approved, its ontologies may be used by a broader community. This is considered to be the deployment phase for the ontologies in the Register. While an older version of an ontology Register is in deployment phase, a new “Draft” version may be created at any time to begin a new development phase for making the next round of changes to the ontologies that are members of that Register.

Details on ontology versioning are described in Ontology Versioning.

Appendix A. Acknowledgments

The following individuals have contributed significantly towards the creation of this specification and are gratefully acknowledged.

Contributors to Current Version:

- Luis Bermudez, SURA
- Kristin Stock, University of Nottingham, Allworlds Geothinking
- Kendall Clark, Clark & Parsia LLC

Contributors to Version 1.5:

- Asuman Dogac, Middle East Technical University, Ankara Turkey
- Yildiray Kabak, Middle East Technical University, Ankara Turkey
- Gokce Banu Laleci, Middle East Technical University, Ankara Turkey