



2 ebXML RegRep Profile for Web Ontology 3 Language (OWL)

4 Version 2.0 Draft 5

5 **November 23, 2009**

6 **Specification URIs:**

7 **Release Notes:**

8 [http://wxforge.wx.ll.mit.edu:8080/jira/secure/ReleaseNote.jspa?](http://wxforge.wx.ll.mit.edu:8080/jira/secure/ReleaseNote.jspa?projectId=10023&styleName=Html&version=10076)
9 [projectId=10023&styleName=Html&version=10076](http://wxforge.wx.ll.mit.edu:8080/jira/secure/ReleaseNote.jspa?projectId=10023&styleName=Html&version=10076)

10 **This Version:**

11 <http://docs.oasis-open.org/regrep/4.0-cd3-draft1/specs/owl-profile/regrep-owl-profile.html>
12 <http://docs.oasis-open.org/regrep/4.0-cd3-draft1/specs/owl-profile/regrep-owl-profile.odt>
13 <http://docs.oasis-open.org/regrep/4.0-cd3-draft1/specs/owl-profile/regrep-owl-profile.pdf>

14 **Previous Version:**

15 <http://docs.oasis-open.org/regrep/v3.0/profiles/owl/regrep-owl-profile-v1.5.html>
16 <http://docs.oasis-open.org/regrep/v3.0/profiles/owl//regrep-owl-profile-v1.5.odt>
17 <http://docs.oasis-open.org/regrep/v3.0/profiles/owl//regrep-owl-profile-v1.5.pdf>

18 **Latest Approved Version:**

19 <http://docs.oasis-open.org/regrep/v3.0/profiles/owl/regrep-owl-profile-v1.5.html>
20 <http://docs.oasis-open.org/regrep/v3.0/profiles/owl//regrep-owl-profile-v1.5.odt>
21 <http://docs.oasis-open.org/regrep/v3.0/profiles/owl//regrep-owl-profile-v1.5.pdf>

22 **Technical Committee:**

23 OASIS ebXML RegRep TC

24 **Chair(s):**

25 Kathryn Breining, Boeing

26 **Editor(s):**

27 Farrukh Najmi, Wellfleet Software

28 **Contributors:**

29 Kathryn Breining, Boeing
30 Kajal Claypool, MIT Lincoln Labs

31 Carl Mattocks, MetLife
32 Farrukh Najmi, Wellfleet Software
33 Oliver Newell, MIT Lincoln Labs
34 Nikola Stojanovic, Individual
35 David Webber, Individual

36 **Related Work:**

37 This specification replaces or supersedes:

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

40 This specification is related to:

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

43 **Declared XML Namespace(s):**

44

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

46

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

47

Abstract:

48
49
50

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

Status:

51
52
53
54
55
56
57
58
59
60
61

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>).

62
63

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>

64 Notices

65 Copyright © OASIS® 2008. All Rights Reserved.

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

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

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

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

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

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

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

104 The names "OASIS", [insert specific trademarked names, abbreviations, etc. here] are trademarks of
105 [OASIS](#), the owner and developer of this specification, and should be used only to refer to the organization
106 and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications,

107 while reserving the right to enforce its marks against misleading uses. Please see <http://www.oasis->
108 [open.org/who/trademark.php](http://www.oasis-open.org/who/trademark.php) for above guidance.

Table of Contents

110	1 Introduction.....	10
111	1.1 Scope.....	10
112	1.1 Use Cases for OWL Support in ebXML RegRep.....	10
113	1.1 Document Organization.....	11
114	1.1 Terminology.....	12
115	1.2 Normative References.....	12
116	1.3 Informative References.....	12
117	2 OWL Overview (Informative).....	14
118	2.1 Semantic Web Languages upon which OWL is Layered.....	14
119	3 Publish Profile.....	15
120	3.1 Publishing OWL.....	15
121	3.1.1 Ontology Partitioning for Collaborative Development.....	15
122	3.1.2 Using RepositoryItem Canonical URL for xml:base Attribute.....	15
123	3.1.3 Import Processing.....	16
124	3.1 Validating OWL.....	16
125	3.1 Cataloging OWL.....	16
126	3.1 Updating OWL.....	17
127	3.2 Deleting OWL.....	17
128	3.3 Semantic Annotation of RegistryObjects.....	17
129	3.3.1 Types of OWL Construct References.....	17
130	3.3.1 Use Cases for Semantic Annotations.....	17
131	3.3.1 Referencing OWL Constructs in Slots.....	18
132	3.3.1 Referencing OWL Constructs in Reference Attributes.....	18
133	4 Ontology Versioning.....	20
134	4.1 Versioning Requirements.....	20
135	4.2 Types of Ontology Changes.....	20
136	4.3 Compatibility of Changes.....	20
137	4.4 Publishing Changes to an Ontology.....	21
138	4.5 Specifying Ontology Context for Queries.....	21
139	5 Discovery Profile.....	23
140	5.1 Specifying Ontology Context for Queries.....	23
141	5.2 Canonical Functions.....	24
142	5.2.1 Canonical Function: dataTypeProperties.....	25
143	5.2.1.1 Parameter Summary.....	25
144	5.2.1.2 Function Semantics.....	25
145	5.2.2 Canonical Function: equivalentClasses.....	26
146	5.2.2.1 Parameter Summary.....	26
147	5.2.2.2 Function Semantics.....	27
148	5.2.3 Canonical Function: equivalentProperties.....	27

149	5.2.3.1 Parameter Summary.....	27
150	5.2.3.2 Function Semantics.....	27
151	5.2.4 Canonical Function: instanceOf.....	27
152	5.2.4.1 Parameter Summary.....	27
153	5.2.4.2 Function Semantics.....	28
154	5.2.5 Canonical Function: instances.....	28
155	5.2.5.1 Parameter Summary.....	28
156	5.2.5.2 Function Semantics.....	28
157	5.2.6 Canonical Function: inverseProperties.....	28
158	5.2.6.1 Parameter Summary.....	29
159	5.2.6.2 Function Semantics.....	29
160	5.2.7 Canonical Function: objectProperties.....	29
161	5.2.7.1 Parameter Summary.....	29
162	5.2.7.2 Function Semantics.....	29
163	5.2.8 Canonical Function: sameAs.....	30
164	5.2.8.1 Parameter Summary.....	30
165	5.2.8.2 Function Semantics.....	30
166	5.2.9 Canonical Function: subClasses.....	30
167	5.2.9.1 Parameter Summary.....	30
168	5.2.9.2 Function Semantics.....	30
169	5.2.10 Canonical Function: subProperties.....	31
170	5.2.10.1 Parameter Summary.....	31
171	5.2.10.2 Function Semantics.....	31
172	5.2.11 Canonical Function: superClasses.....	31
173	5.2.12 Parameter Summary.....	31
174	5.2.12.1 Function Semantics.....	31
175	5.2.13 Canonical Function: superProperties.....	32
176	5.2.13.1 Parameter Summary.....	32
177	5.2.13.2 Function Semantics.....	32
178	5.2.14 Canonical Function: similarTerms.....	33
179	5.2.14.1 Parameter Summary.....	33
180	5.2.14.2 Function Semantics.....	33
181	5.3 Invoking SPARQL Queries.....	34
182	5.3.1 QueryRequest Requirements.....	34
183	5.3.2 QueryResponse Requirements.....	35
184	6 Governance Profile.....	36
185	6.1 Creation of an Ontology Register.....	36
186	6.2 Designation of Organization Roles.....	36
187	6.3 Designation of Person Roles.....	36
188	6.4 Publishing Draft Ontology Files.....	37
189	6.5 Submitting Ontologies for Review.....	37
190	6.6 Receiving Ontology Change Proposals.....	37
191	6.7 Reviewing Ontology Change Proposal.....	38
192	6.8 Creating New Version of Ontologies.....	38
193		

Illustration Index

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

194

Index of Tables

Table 1: Namespaces Used.....	3
-------------------------------	---

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

195

1 Introduction

196

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

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

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

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

1.1 Scope

207

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

- 210 ● How OWL ontologies MAY be published to an ebXML RegRep server ([Publish Profile](#))
- 211 ● How ebXML RegRep metadata (RegistryObjects) within and ebXML RegRep server MAY
212 reference published OWL ontology constructs ([Semantic Annotation](#))
- 213 ● How ebXML RegRep queries may discover semantically annotated RegistryObjects using
214 published OWL ontologies ([Discovery Profile](#))
- 215 ● How ebXML RegRep queries may be used to perform SPARQL queries on the OWL repository
216 content ([Invoking SPARQL Queries](#))
- 217 ● How published OWL ontologies may be governed using ebXML RegRep policies and registration
218 procedures ([Governance Profile](#))

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

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

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

1.1 Use Cases for OWL Support in ebXML RegRep

226

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

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

- 231 ○ Support the use of ontological concepts as attribute value for ebRIM objects, specifically as
232 value of slots, association type and other RegistryObject attributes
- 233 ● Semantic discovery
- 234 ○ Allows discovery of objects such as datasets, services etc. based on a semantic match on a
235 attribute value rather than a precise literal match
- 236 ○ Makes use of domain and mapping ontologies to perform semantic harmonization and
237 determine similar terms
- 238 ○ Makes use of ontologies to match parent (broader) and child (narrower) terms
- 239 ○ May make use of semantically annotated ebRIM metadata
- 240 ● Collaborative ontology publishing and management
- 241 ○ Supports ontology elements to be published by multiple organizations and individuals
242 collaboratively
- 243 ○ Allows browsing and discovering ontology elements within regrep
- 244 ○ Provide governance process for managing changes to an ontologies
- 245 ● Semantic mediation
- 246 ○ Allows specialized clients to bi-directionally transforms data from one format to another. A
247 specific example is mediation between OGC WFS and JMBL. For example, a client may
248 make a WFS request to a server server that supports JMBL.
- 249 ○ Allows data registered using different data standards (e.g. 19139, FGDC) to be discovered
250 uniformly using the same query
- 251 ○ Mediation is an application of semantically enhanced RegRep rather than a feature of it
- 252 In addition to above use cases, this specification aims to address the use cases identified in [ORUC].

253 1.1 Document Organization

254 The document is organized as follows:

- 255 ● [Chapter 1: Introduction](#) - provides an introduction to the rest of this document
- 256 ● [Chapter 2: OWL Overview](#) - provides an overview of the Web Ontology Language
- 257 ● [Chapter 3: Publish Profile](#) - specifies how OWL Ontologies are published to the server
- 258 ● [Chapter 4: Ontology Versioning](#) – specifies how multiple versions of ontologies are supported
- 259 ● [Chapter 5: Discovery Profile](#) - specifies how discovery queries make use of OWL Ontologies
260 available in the server
- 261 ● [Chapter 6: Governance Profile](#) - specifies how OWL ontologies are governed with the server

262 1.1 Terminology

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

266 1.2 Normative References

- 267 **[ebRIM]** ebXML RegRep Information Model Version 4.0
268 [http://www.oasis-open.org/committees/regrep/documents/4.0/specs/regrep-
rim-4.0-cs.pdf](http://www.oasis-open.org/committees/regrep/documents/4.0/specs/regrep-
269 rim-4.0-cs.pdf)
- 270 **[ebRS]** ebXML RegRep Services and Protocols 4.0
271 [http://www.oasis-open.org/committees/regrep/documents/4.0/specs/regrep-
rs-4.0-cs.pdf](http://www.oasis-open.org/committees/regrep/documents/4.0/specs/regrep-
272 rs-4.0-cs.pdf)
- 273 **[OWL]** Web Ontology Language Overview, W3C Recommendation 10 February 2004
274 <http://www.w3.org/TR/2004/REC-owl-features-20040210/>
- 275 **[OWL/REF]** OWL Web Ontology Language Reference, W3C Recommendation 10 February
276 2004
277 <http://www.w3.org/TR/2004/REC-owl-ref-20040210/>
- 278 **[RDF/XML]** RDF/XML Syntax Specification (Revised) W3C Recommendation 10 February
279 2004
280 <http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/>
- 281 **[RDFS]** RDF Vocabulary Description Language 1.0: RDF Schema, W3C
282 Recommendation 10 February 2004
283 <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>
- 284 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF
285 RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>
- 286 **[SPARQL]** SPARQL Query Language for RDF, W3C Recommendation 15 January 2008
287 <http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/>
- 288 **[SPARQLX]** SPARQL Query Results XML Format, W3C Recommendation 15 January 2008
289 <http://www.w3.org/TR/2008/REC-rdf-sparql-XMLres-20080115/>

290 1.3 Informative References

- 291 **[LeeHendler]** Berners-Lee, T., Hendler, J., Lassila, O., "The Semantic Web", Scientific
292 American, May 2001.
293 <http://www.scientificamerican.com/article.cfm?id=the-semantic-web>
- 294 **[MMI]** Marine Metadata Interoperability Project
295 <http://marinemetadata.org>
- 296 **[OWLG]** OWL Web Ontology Language Guide, W3C Recommendation 10 February 2004
297 <http://www.w3.org/TR/2004/REC-owl-guide-20040210/>
- 298 **[ORUC]** MMI Ontology Repository Use Cases
299 [http://marinemetadata.org/community/teams/ont/ontwebservices/mmirepository/o
ntrepositoryuc](http://marinemetadata.org/community/teams/ont/ontwebservices/mmirepository/o
300 ntrepositoryuc)

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

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

307 **[ALIGN]** A format for ontology alignment
308 <http://alignapi.gforge.inria.fr/format.html>

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

311 **[RDFP]** RDF Primer
312 <http://www.w3.org/TR/rdf-primer/>

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

315

316 2 OWL Overview (Informative)

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

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

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

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

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

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

335 2.1 Semantic Web Languages upon which OWL is Layered

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

345 **3 Publish Profile**

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

347 **3.1 Publishing OWL**

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

350 The following additional requirements are defined for publishing OWL:

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

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

367 **3.1.1 Ontology Partitioning for Collaborative Development**

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

376 **3.1.2 Using RepositoryItem Canonical URL for xml:base Attribute**

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

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

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

384 3.1.3 Import Processing

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

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

391 3.1 Validating OWL

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

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

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

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

404 3.1 Cataloging OWL

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

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

- 415 ● [//owl:Ontology/rdfs:comment](#) element MUST be cataloged as the value of the
416 Description/LocalizedString/@value attribute of the on the ExtrinsicObject for the OWL
417 repository item.
- 418 ○ If the rdfs:comment has an xml:lang attribute specified then the
419 Description/LocalizedString/@locale attribute MUST have the value of that attribute.
- 420 ○ If the rdfs:comment does not have an xml:lang attribute specified then
421 Description/LocalizedString/@locale attribute MUST NOT be specified

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

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

426 3.1 Updating OWL

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

429 3.2 Deleting OWL

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

431 3.3 Semantic Annotation of RegistryObjects

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

435 3.3.1 Types of OWL Construct References

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

- 439 ● owl:Class
- 440 ● owl:Individual
- 441 ● owl:ObjectProperty
- 442 ● owl:DatatypeProperty

443 3.3.1 Use Cases for Semantic Annotations

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

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

- 447 ● Referencing OWL constructs in EmailAddress as value of type attribute
- 448 ● Referencing OWL constructs in PostalAddress as value of type attribute
- 449 ● Referencing OWL constructs in TelephoneNumber as value of type attribute
- 450 ● Referencing OWL constructs in external Classifications as value of nodeRepresentation attribute

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

454 3.3.1 Referencing OWL Constructs in Slots

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

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

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

```

464 <Person id="urn:acme:person:Danyal" ...>
465 ...
466 <rim:Slot name="foodPreference"
467   dataType="urn:oasis:names:tc:ebxml-
468   regrep:profile:webontology:2.0:resourceReference">
469   <rim:ValueList>
470     <rim:ValueListItem xsi:type="rim:StringValueType">
471       <rim:Value>http://www.co-
472       ode.org/ontologies/pizza/pizza.owl#VegetarianPizza</rim:Value>
473     </rim:ValueListItem>
474   </rim:ValueList>
475 </rim:Slot>
476 </Person>
  
```

478 3.3.1 Referencing OWL Constructs in Reference Attributes

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

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

481 The following example shows an Association between two Person objects where the type attribute
 482 references a "hasBrother" ObjectProperty.

```

483 <Association ...
484   sourceObject="urn:acme:person:Danyal"
485   targetObject="urn:acme:person:Omar"
486   type="http://www.mindswap.org/ontologies/family.owl#hasBrother"/>
  
```

487 Issue: No way to distinguish a normal regrep reference from reference to an OWL resource?? Should we
488 use a prefix hack like:
489
490 type="owlref:<http://www.mindswap.org/ontologies/family.owl#hasBrother>"

491 4 Ontology Versioning

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

495 4.1 Versioning Requirements

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

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

504 4.2 Types of Ontology Changes

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

- 507 ● Create - Creation of new classes, properties and individuals
- 508 ● Update - Update of existing classes, properties and individuals
- 509 ● Delete - Deletion of existing classes, properties and individuals

510 4.3 Compatibility of Changes

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

- 513 ● Backward compatible – Individuals that are created for old version of ontology are compatible with
514 new version of ontology. Client applications compatible with old version of ontology are
515 compatible with new version of ontology
- 516 ● Forward compatible – Individuals that are created for new version of ontology are compatible with
517 old version of ontology. Client applications compatible with new version of ontology are
518 compatible with old version of ontology
- 519 ● Incompatible – The versions of the ontology are incompatible with each other

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

523 “Update” and “Delete” changes are usually incompatible. They typically require publishing a modified
524 version of the original OWL file to the server with specific constructs updated or deleted. Such a modified

525 version of the original OWL file should have its own unique version-specific [rdf/RDF/@xml:base](#) attribute
526 value.

527 **4.4 Publishing Changes to an Ontology**

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

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

- 537 ● When the mode attribute value is “CreateOrUpdate” then submitting a modified version of an
538 existing OWL file updates (replaces) the existing version
- 539 ● When the mode attribute value is “CreateOrVersion” then submitting a modified version of an
540 existing OWL file creates a new version in [Validating OWL](#).

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

544 A deployment may further control the review and approval of changes to an ontology as described in
545 [Governance Profile](#).

546 **4.5 Specifying Ontology Context for Queries**

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

554 The details of how the ontology context is defined and how it impacts the Query protocol is defined in
555 [Discovery Profile](#).

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

560

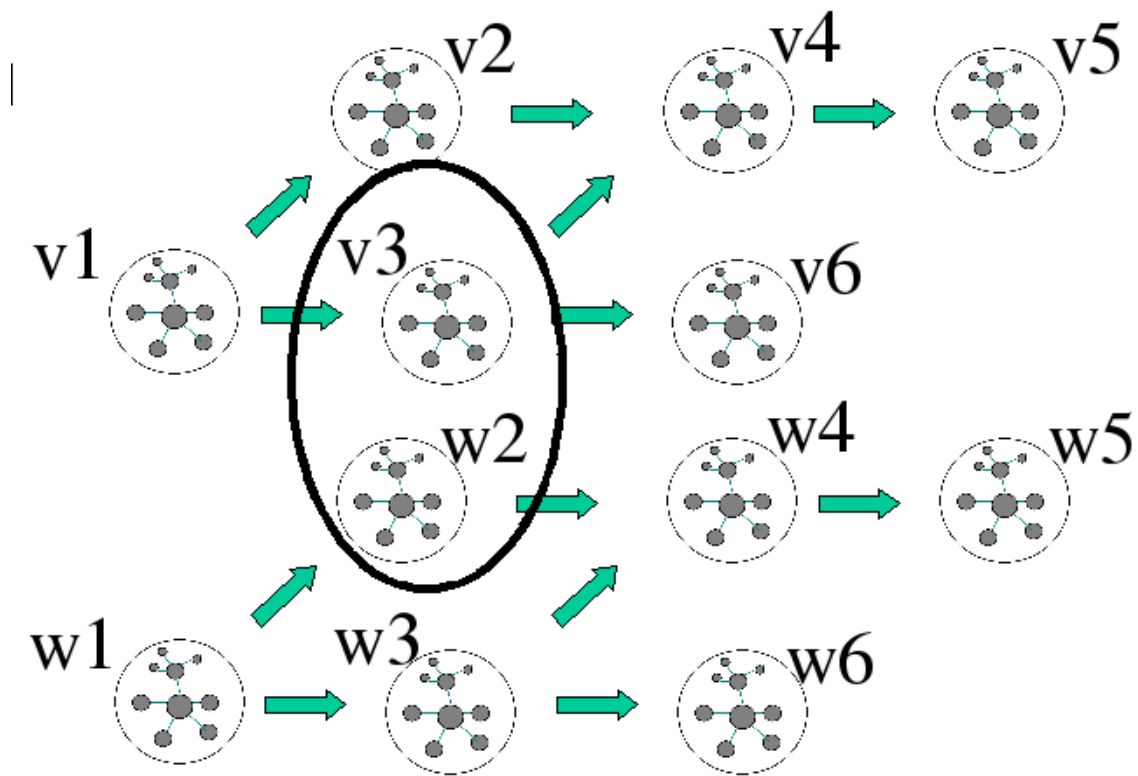


Illustration 1: Ontology Context - a Visual Example

5 Discovery Profile

562

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

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

5.1 Specifying Ontology Context for Queries

571

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

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

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

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

586

```
587 <query:QueryRequest maxResults="-1" startIndex="0" ...>  
588   <rim:Slot  
589     name="urn:oasis:names:tc:ebxml-  
590     regrep:profile:webontology:2.0:QueryRequest:ontologyContext "  
591     dataType="urn:oasis:names:tc:ebxml-regrep:DataType:String"  
592     collectionType="urn:oasis:names:tc:ebxml-regrep:CollectionType:SortedSet "  
593   >  
594  
595   <rim:ValueList>  
596     <rim:ValueListItem xsi:type="StringValueType">  
597       <rim:Value>urn:example:ontology:V:3.0</rim:Value>  
598       <rim:Value>urn:example:ontology:W:2.0</rim:Value>  
599     </rim:ValueListItem>  
600   </rim:ValueList>  
601 </rim:Slot>  
602   ...
```

```
603 <query:Query queryDefinition="urn:example:query:SomeExample">
604   ...
605 </query:Query>
606 </query:QueryRequest>
```

607

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

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

614 A server MUST perform all evaluation of SPARQL queries and [canonical owl functions](#) defined by this
615 profile as if the only ontologies defined are the specific versions of specific ontologies defined by the
616 ontology context for the QueryRequest. Other versions and other ontologies MUST NOT have any impact
617 on the evaluation of SPARQL queries and [canonical owl functions](#).

618 5.2 Canonical Functions

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

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

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

630 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

631 *Table 2: Canonical Functions Defined By This Profile*

632 5.2.1 Canonical Function: dataTypeProperties

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

635 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

636 5.2.1.2 Function Semantics

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

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

```

644 <query:QueryRequest ...>
645   <rim:Slot
646     name="urn:oasis:names:tc:ebxml-
647     regrep:profile:webontology:2.0:QueryRequest:ontologyContext "
648     dataType="urn:oasis:names:tc:ebxml-regrep:DataType:String"
649     collectionType="urn:oasis:names:tc:ebxml-regrep:CollectionType:SortedSet "
650   >
651
652   <rim:ValueList>
653     <rim:ValueListItem xsi:type="StringValueType">
654       <rim:Value>urn:example:ontology:camera.owl:1.0</rim:Value>
655     </rim:ValueListItem>
656   </rim:ValueList>
657 </rim:Slot>
658 ...
659 <query:Query queryDefinition="urn:oasis:names:tc:ebxml-
660 regrep:query:AdhocQuery">
661
662   <rim:Slot name="queryLanguage">
663     <rim:ValueList>
664       <rim:ValueListItem xsi:type="StringValueType"
665 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
666       <rim:Value>urn:oasis:names:tc:ebxml-
667 regrep:QueryLanguage:EJBQL</rim:Value>
668     </rim:ValueListItem>
669   </rim:ValueList>
670 </rim:Slot>
671
672   <rim:Slot name="queryExpression">
673     <rim:ValueList>
674       <rim:ValueListItem xsi:type="StringValueType"
675 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
676       <rim:Value>#@SELECT Object(eo) FROM ExtrinsicObjectType eo LEFT
677 OUTER JOIN eo.slot semref_slot WHERE (semref_slot.dataType IN (@#
678 owl:datatypeProperties(\ "http://www.xfront.com/owl/ontologies/camera/#Range\"
679 , true, \",\") #@) )@#</rim:Value>
680     </rim:ValueListItem>
681   </rim:ValueList>
682 </rim:Slot>
683
684 </query:Query>
685 </query:QueryRequest>
  
```

686 5.2.2 Canonical Function: equivalentClasses

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

689 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	
--	---	--

690 5.2.2.2 Function Semantics

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

696

697 5.2.3 Canonical Function: `equivalentProperties`

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

700 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

701 5.2.3.2 Function Semantics

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

707

708 5.2.4 Canonical Function: `instanceOf`

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

711 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

712 5.2.4.2 Function Semantics

- 713 ● The server **MUST** return a string if the query is processed without any exceptions
- 714 ● The string **MAY** be empty if no Individual is found with specified id or if no classes are found that
715 are types for the Individual matching specified id
- 716 ● The string **MUST** consist of a set of class ids separated by the appropriate delimiter character
717 when any classes are found that are a type for the Individual matching specified id
- 718

719 5.2.5 Canonical Function: instances

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

722 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

723 5.2.5.2 Function Semantics

- 724 ● The server **MUST** return a string if the query is processed without any exceptions
- 725 ● The string **MAY** be empty if no class is found with specified id or if no individuals are found that
726 are an instance of the class matching specified id
- 727 ● The string **MUST** consist of a set of individuals' ids separated by the appropriate delimiter
728 character when any individuals are found that are an instance of the class matching specified id
- 729

730 5.2.6 Canonical Function: inverseProperties

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

733 **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

734 **5.2.6.2 Function Semantics**

- 735 ● The server MUST return a string if the query is processed without any exceptions
 - 736 ● The string MAY be empty if no property is found with specified id or if no inverse properties are
737 found for property matching specified id
 - 738 ● The string MUST consist of a set of inverse property ids separated by the appropriate delimiter
739 character when any inverse property is found for property matching specified id
- 740

741 **5.2.7 Canonical Function: objectProperties**

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

744 **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

745 **5.2.7.2 Function Semantics**

- 746 ● The server MUST return a string if the query is processed without any exceptions
 - 747 ● The string MAY be empty if no class is found with specified id or if no object properties are found
748 for class matching specified id
 - 749 ● The string MUST consist of a set of object property ids separated by the appropriate delimiter
750 character when any object properties are found for class matching specified id
- 751

752 5.2.8 Canonical Function: sameAs

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

755 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

756 5.2.8.2 Function Semantics

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

762

763 5.2.9 Canonical Function: subclasses

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

766 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

767 5.2.9.2 Function Semantics

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

773

774 **5.2.10 Canonical Function: subProperties**

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

777 **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

778 **5.2.10.2 Function Semantics**

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

784

785 **5.2.11 Canonical Function: superClasses**

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

788 **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

789 **5.2.12.1 Function Semantics**

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

791 ● The string MAY be empty if no class is found with specified id or if no ancestor classes are found
792 for class matching specified id

793 ● The string MUST consist of a set of ancestor class ids separated by the appropriate delimiter
794 character when any ancestor classes are found for class matching specified id

795

796

797 5.2.13 Canonical Function: **superProperties**

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

800 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

801 5.2.13.2 Function Semantics

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

803 ● The string MAY be empty if no property is found with specified id or if no ancestor properties are
804 found for property matching specified id

805 ● The string MUST consist of a set of ancestor property ids separated by the appropriate delimiter
806 character when any ancestor classes are found for class matching specified id

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

```
812 <query:QueryRequest ...>  
813   ...  
814   <query:Query queryDefinition="urn:oasis:names:tc:ebxml-  
815   regrep:query:FindAssociations">  
816     <rim:Slot name="sourceObjectId">  
817       <rim:ValueList>  
818         <rim:ValueListItem xsi:type="StringValueList">  
819           <rim:Value>urn:acme:Person:Danyal</rim:Value>  
820         </rim:ValueListItem>  
821       </rim:ValueList>  
822     </rim:Slot>  
823     <rim:Slot name="associationType">  
824       <rim:ValueList>
```



```

825     <rim:ValueListItem xsi:type="StringValueTypes">
826         <rim:Value>owlp:superProperties ("http://www.mindswap.org/ontologies/
827 family.owl#hasBrother", true, ",")#@#@#</rim:Value>
828     </rim:ValueListItem>
829 </rim:ValueList>
830 </rim:Slot>
831 </query:Query>
832 </query:QueryRequest>

```

833 5.2.14 Canonical Function: similarTerms

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

837 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

838 5.2.14.2 Function Semantics

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

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

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

```

850 <query:QueryRequest ...>
851   ...
852   <query:Query queryDefinition="urn:example:DatasetDiscoveryQuery">
853     <rim:Slot name="field">
854       <rim:ValueList>
855         <rim:ValueListItem xsi:type="StringValueTypes">
856           <rim:Value>owlp:similarTerms (\ "temperature\ ", 0.8)#@#@#</rim:Value>

```

```
857     </rim:ValueListItem>
858   </rim:ValueList>
859 </rim:Slot>
860 </query:Query>
861 </query:QueryRequest>
```

862

863 5.3 Invoking SPARQL Queries

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

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

868 5.3.1 QueryRequest Requirements

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

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

877

878 The following example shows SPARQL query expression within a QueryRequest:

```
879 <query:QueryRequest ...>
880   ...
881   <query:Query queryDefinition="urn:oasis:names:tc:ebxml-
882   regrep:query:AdhocQuery">
883     <rim:Slot name="queryLanguage">
884       <rim:ValueList>
885         <rim:ValueListItem xsi:type="StringValue">
886           <rim:Value>urn:oasis:names:tc:ebxml-
887   regrep:QueryLanguage:SPARQL</rim:Value>
888         </rim:ValueListItem>
889       </rim:ValueList>
890     </rim:Slot>
891     <rim:Slot name="queryExpression">
892       <rim:ValueList>
893         <rim:ValueListItem xsi:type="StringValue">
894           <rim:Value>
895   SELECT ?givenName
896   WHERE
897     { ?y <http://www.w3.org/2001/vcard-rdf/3.0#Family> "Smith" .
898       ?y <http://www.w3.org/2001/vcard-rdf/3.0#Given> ?givenName .
899     }
900           </rim:Value>
901         </rim:ValueListItem>
```

```
902     </rim:ValueList>
903   </rim:Slot>
904 </query:Query>
905 </query:QueryRequest>
```

906

907 5.3.2 QueryResponse Requirements

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

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

920 The following example shows SPARQL response within a QueryResponse:

```
921 <query:QueryResponse ...>
922   ...
923   <rim:Slot
924     name="urn:oasis:names:tc:ebxml-
925     regrep:profile:webontology:2.0:sparqlResponse"
926     dataType="urn:oasis:names:tc:ebxml-regrep:profile:webontology:2.0:sparql">
927     <rim:ValueList>
928       <rim:ValueListItem xsi:type="rim:AnyValueType">
929         <sparqlx:sparql>
930           ...
931         <sparqlx:sparql>
932       </rim:ValueListItem>
933     </rim:ValueList>
934   </rim:Slot>
935 </query:QueryResponse>
```

937

938

6 Governance Profile

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

944

6.1 Creation of an Ontology Register

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

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

952

6.2 Designation of Organization Roles

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

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

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

964

6.3 Designation of Person Roles

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

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

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

976 **6.4 Publishing Draft Ontology Files**

977 Once an ontology Register has been created and organization and person roles have been assigned
978 authorized ontology developers MAY begin publishing ontology files to the server as described in [Publish](#)
979 [Profile](#). Once an ontology file has been published to the server the submitter MAY add it as a members of
980 an ontology Register that has a status of "Draft". This includes a newly created Register or a new version
981 of an existing Register.

982 Throughout the ontology development phase, the ontology Register version stays in a "Draft" status.
983 During this time, its ChangeProposalSubmitter MAY freely make changes to the ontology files that are its
984 members as described in [Publish Profile](#).

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

989 **6.5 Submitting Ontologies for Review**

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

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

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

998 **6.6 Receiving Ontology Change Proposals**

999 When a ChangeProposalReceiver is notified of the submission of a ontology changes in an ontology
1000 Register it MAY take on the activity of performing acceptance review for the proposed changes by setting
1001 the status of the Register to "UnderAcceptanceReview".

1002 The default Register notification policy defined by [ebRS] requires that when a Register status is changed
1003 to "UnderAcceptanceReview", the server MUST deliver a notification to all ChangeProposalReceiver's.
1004 This lets other ChangeProposalReceiver's know that a peer has begun working on the activity and they
1005 need not work on it.

1006 The ChangeProposalReceiver MAY then accept (set status to "Accepted") or reject (set status to
1007 "Rejected) the change proposal in it entirety as described in Registration Procedures feature set of
1008 [ebRS].

1009 The ChangeProposalReceiver MAY NOT be a person and instead MAY be an automated service that
1010 implements a NotificationListener interface.

1011 The default Register notification policy defined by [ebRS] requires that when a Register status is changed
1012 to “Accepted”, the server MUST deliver a notification to all ChangeProposalReviewers. This initiates the
1013 change proposal review activity.

1014 **6.7 Reviewing Ontology Change Proposal**

1015 When a ChangeProposalReviewer is notified that ontology changes in an ontology Register have been
1016 accepted for review, it MAY take on the activity of performing detailed review of the proposed changes by
1017 setting the status of the Register to “UnderReview”.

1018 The default Register notification policy defined by [ebRS] requires that when a Register status is changed
1019 to “UnderReview”, the server MUST deliver a notification to all ChangeProposalReviewer's. This lets other
1020 ChangeProposalReviewer's know that a peer has begun working on the activity and they need not work
1021 on it.

1022 The ChangeProposalReviewer MUST then perform a detailed review of the proposed changes as
1023 described by [ebRS]. The following OWL specific requirements are defined for the review process:

1024 **Issue: Need OWL specific review requirements (if any) here??**

1025 During the review the ChangeProposalReviewer MAY then approve (set status to “Approved”) or reject
1026 (set status to “Rejected”) the change proposal in it entirety or at the granularity of individual changes as
1027 described in Registration Procedures feature set of [ebRS].

1028 The default Register notification policy defined by [ebRS] requires that when a Register status is changed
1029 to “Approved”, the server MUST deliver a notification to all ChangeProposalSubmitters.

1030 **6.8 Creating New Version of Ontologies**

1031 Once a specific version of an ontology Register has been reviewed and approved, its ontologies may be
1032 used by a broader community. This is considered to be the deployment phase for the ontologies in the
1033 Register. While an older version of an ontology Register is in deployment phase, a new “Draft” version
1034 may be created at any time to begin a new development phase for making the next round of changes to
1035 the ontologies that are members of that Register.

1036 Details on ontology versioning are described in [Ontology Versioning](#).

1037 **Appendix A. Acknowledgments**

1038 The following individuals have contributed significantly towards the creation of this specification and are
1039 gratefully acknowledged.

1040 **Contributors to Current Version:**

- 1041 ● Luis Bermudez, SURA
- 1042 ● Kristin Stock, University of Nottingham, Allworlds Geothinking
- 1043 ● Kendall Clark, Clark & Parsia LLC

1044 **Contributors to Version 1.5:**

- 1045 ● Asuman Dogac, Middle East Technical University, Ankara Turkey
- 1046 ● Yildiray Kabak, Middle East Technical University, Ankara Turkey
- 1047 ● Gokce Banu Laleci, Middle East Technical University, Ankara Turkey