



ebXML Registry Profile for Web Ontology

Language (OWL)

Version 1.5

Committee Draft 01, September 25, 2006

Document identifier:

regrep-owl-profile-v1.5-cd01

Specification URIs:

This Version:

docs.oasis-open.org/regrep/v3.0/profiles/owl/regrep-owl-profile-v1.5-cd01.html

docs.oasis-open.org/regrep/v3.0/profiles/owl/regrep-owl-profile-v1.5-cd01.pdf

docs.oasis-open.org/regrep/v3.0/profiles/owl/regrep-owl-profile-v1.5-cd01.odt

Previous Version: [N/A]

Latest Version:

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

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

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

Technical Committee: OASIS ebXML Registry Technical Committee

Editors:

Name
Asuman Dogac

Abstract:

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

25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75

Status:

This document was last revised or approved by the ebXML Registry TC on the above date. The level of approval is also listed above. Check the current location noted above for possible later revisions of this document. This document is updated periodically on no particular schedule.

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 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 www.oasis-open.org/committees/regrep/ipr.php.

The non-normative errata page for this specification is located at www.oasis-open.org/committees/regrep.

Notices:

Copyright © OASIS® 1993–2007. All Rights Reserved. OASIS trademark, IPR and other policies apply.

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

76 specification. OASIS may include such claims on its website, but disclaims any obligation to do
77 so.

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

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

95

1 Table of Contents

96		
97	1 Table of Contents.....	3
98	1 Introduction.....	10
99	1.1 Terminology.....	11
100	1.2 Conventions.....	11
101	1.3 Recommended Enhancements.....	11
102	2 OWL Overview.....	12
103	2.1 Semantic Web Languages upon which OWL is Layered.....	12
104	2.2 OWL Lite Constructs.....	13
105	2.2.1 RDF Schema Features.....	13
106	2.2.2 (In)Equality.....	13
107	2.2.3 Property Characteristics	13
108	2.2.4 Property Restrictions.....	13
109	2.2.5 Restricted Cardinality.....	13
110	2.2.6 Class Intersection.....	13
111	2.2.7 Versioning.....	14
112	2.2.8 Annotation Properties	14
113	2.2.9 Datatypes	14
114	2.3 OWL DL Constructs.....	14
115	2.3.1 Class Axioms.....	14
116	2.3.2 Boolean Combinations of Class Expressions	14
117	2.3.3 Arbitrary Cardinality	14
118	2.3.4 Filler Information.....	14
119	3 ebXML Registry Overview.....	15
120	3.1 Overview of [ebRIM].....	15
121	3.1.1 RegistryObject.....	16
122	3.1.2 Object Identification.....	16
123	3.1.3 Object Naming and Description.....	17
124	3.1.4 Object Attributes.....	17
125	3.1.4.1 Slot Attributes.....	17
126	3.1.5 Object Classification.....	18
127	3.1.6 Object Association.....	18
128	3.1.7 Object References To Web Content.....	19
129	3.1.8 Object Packaging.....	19
130	3.1.9 ExtrinsicObject	20
131	3.1.10 Service Description.....	20
132	3.2 Overview of [ebRS].....	20
133	4 Representing OWL Lite Constructs in ebRIM	21
134	4.1 Representing RDF Schema Features in ebRIM.....	21
135	4.1.1 owl:Class → rim:ClassificationNode.....	21
136	4.1.2 rdf:Property → rim:Association Type HasProperty.....	21
137	4.1.3 rdfs:subPropertyOf → rim:Association Type SubPropertyOf.....	22
138	4.1.4 rdfs:subClassOf → rim:Association Type SubClassOf.....	22
139	4.1.5 owl:Individual → rim:ExtrinsicObject.....	23
140	4.2 Representing OWL (In)Equality Constructs in ebXML RIM.....	24

141	4.2.1 owl:equivalentClass, owl:equivalentProperty → rim:Association Type EquivalentTo	24
142	4.2.2 owl:sameAs → rim:Association Type SameAs.....	24
143	4.2.3 owl:differentFrom → rim:Association Type DifferentFrom.....	24
144	4.2.4 owl:AllDifferent.....	25
145	4.3 Representing OWL Property Characteristics in ebRIM.....	26
146	4.3.1 owl:ObjectProperty → rim:Association Type objectProperty.....	26
147	4.3.2 owl:DatatypeProperty → rim:Association Type DatatypeProperty.....	26
148	4.3.3 owl:TransitiveProperty → rim:Association Type TransitiveProperty.....	26
149	4.3.4 owl:inverseOf → rim:Association Type InverseOf.....	27
150	4.3.5 owl:SymmetricProperty→ rim:Association Type SymmetricProperty.....	28
151	4.3.6 owl:FunctionalProperty→ rim:Association Type FunctionalProperty.....	28
152	4.3.7 owl:InverseFunctionalProperty→ rim:Association Type InverseFunctionalProperty.....	29
153	4.4 OWL Property Restrictions in ebXML RIM.....	29
154	4.5 Representing OWL Restricted Cardinality in ebXML RIM.....	30
155	4.5.1 owl:minCardinality (only 0 or 1).....	30
156	4.5.2 owl:maxCardinality (only 0 or 1).....	31
157	4.5.3 owl:cardinality (only 0 or 1).....	32
158	4.6 Representing OWL Class Intersection in ebXML RIM.....	32
159	4.7 Representing OWL Versioning in ebXML RIM.....	33
160	4.7.1 owl:versionInfo, owl:priorVersion.....	33
161	4.8 Representing OWL Annotation Properties in ebXML RIM.....	34
162	4.8.1 rdfs:label.....	34
163	4.8.2 rdfs:comment.....	34
164	4.8.3 rdfs:seeAlso.....	34
165	4.9 OWL Datatypes in ebXML RIM.....	35
166	5 Cataloging Service Profile.....	36
167	5.1 Invocation Control File.....	36
168	5.2 Input Metadata.....	36
169	5.3 Input Content.....	36
170	5.4 Output Metadata.....	37
171	5.4.1 owl:Class → rim:ClassificationNode.....	37
172	5.4.2 rdf:Property → rim:Association Type HasProperty.....	37
173	5.4.3 rdfs:subPropertyOf → rim:Association Type SubPropertyOf.....	37
174	5.4.4 rdfs:subClassOf → rim:Association Type subClassOf.....	37
175	5.4.5 owl:Individual → rim:ExtrinsicObject.....	37
176	5.4.6 owl:equivalentClass, owl:equivalentProperty → rim:Association Type EquivalentTo	37
177	5.4.7 owl:sameAs → rim:Association Type SameAs	37
178	5.4.8 owl:differentFrom → rim:Association Type DifferentFrom.....	37
179	5.4.9 owl:AllDifferent → rim:RegistryPackage.....	37
180	5.4.10 owl:ObjectProperty → rim:Association Type ObjectProperty.....	38
181	5.4.11 owl:DatatypeProperty → rim:Association Type DatatypeProperty.....	38
182	5.4.12 owl:TransitiveProperty → rim:Association Type TransitiveProperty.....	38
183	5.4.13 owl:inverseOf → rim:Association Type InverseOf.....	38
184	5.4.14 owl:SymmetricProperty→ rim:Association Type SymetricProperty.....	38
185	5.4.15 owl:FunctionalProperty→ rim:Association Type FunctionalProperty.....	38
186	5.4.16 owl:InverseFunctionalProperty→ rim:Association Type InverseFunctionalProperty.....	38

187	5.4.17 owl:minCardinality (only 0 or 1).....	38
188	5.4.18 owl:maxCardinality (only 0 or 1).....	39
189	5.4.19 owl:cardinality.....	39
190	5.4.20 owl:intersectionOf.....	39
191	5.4.21 rdfs:label.....	39
192	5.4.22 rdfs:comment.....	39
193	5.4.23 rdfs:seeAlso.....	39
194	6 Discovery Profile.....	40
195	6.1 All SuperProperties Discovery Query.....	40
196	6.1.1 Parameter \$propertyName.....	40
197	6.1.2 Example of All SuperProperties Discovery Query.....	40
198	6.2 Immediate SuperClass Discovery Query.....	41
199	6.2.1 Parameter \$className.....	41
200	6.2.2 Example of Immediate SuperClass Discovery Query.....	41
201	6.3 Immediate SubClass Discovery Query.....	42
202	6.3.1 Parameter \$className.....	42
203	6.3.2 Example of Immediate SubClasses Discovery Query.....	42
204	6.4 All SuperClasses Discovery Query.....	42
205	6.4.1 Parameter \$className.....	43
206	6.4.2 Example of All SuperClasses Discovery Query.....	43
207	6.5 All SubClasses Discovery Query.....	43
208	6.5.1 Parameter \$className.....	43
209	6.5.2 Example of All SubClasses Discovery Query.....	43
210	6.6 EquivalentClasses Discovery Query.....	44
211	6.6.1 Parameter \$className.....	44
212	6.6.2 Example of EquivalentClasses Discovery Query.....	44
213	6.7 EquivalentProperties Discovery Query.....	45
214	6.7.1 Parameter \$propertyName.....	45
215	6.7.2 Example of EquivalentProperties Discovery Query.....	45
216	6.8 SameExtrinsicObjects Discovery Query.....	46
217	6.8.1 Parameter \$extrinsicObjectName.....	46
218	6.8.2 Example of SameExtrinsicObjects Discovery Query.....	46
219	6.9 DifferentExtrinsicObjects Discovery Query.....	46
220	6.9.1 Parameter \$extrinsicObjectName.....	47
221	6.9.2 Example of DifferentExtrinsicObjects Discovery Query.....	47
222	6.10 AllDifferentRegistryObject Discovery Query.....	47
223	6.10.1 Parameter \$registryObjectName.....	47
224	6.10.2 Example of AllDifferentRegistryObjects Discovery Query.....	47
225	6.11 ObjectProperties Discovery Query.....	48
226	6.11.1 Parameter \$className.....	48
227	6.11.2 Example of ObjectProperties Discovery Query.....	48
228	6.12 ImmediateInheritedObjectProperties Discovery Query.....	49
229	6.12.1 Parameter \$className.....	49
230	6.12.2 Example of ImmediateInheritedObjectProperties Discovery Query.....	49
231	6.13 AllInheritedObjectProperties Discovery Query.....	50
232	6.13.1 Parameter \$className.....	50

233	6.13.2 Example of AllInheritedObjectProperties Discovery Query.....	50
234	6.14 DatatypeProperties Discovery Query.....	51
235	6.14.1 Parameter \$className.....	51
236	6.14.2 Example of DatatypeProperties Discovery Query.....	51
237	6.15 AllInheritedDatatypeProperties Discovery Query.....	51
238	6.15.1 Parameter \$className.....	52
239	6.15.2 Example of AllInheritedDatatypeProperties Discovery Query.....	52
240	6.16 TransitiveRelationships Discovery Query.....	52
241	6.16.1 Parameter \$className.....	53
242	6.16.2 Parameter \$propertyName.....	53
243	6.16.3 Example of TransitiveRelationships Discovery Query.....	53
244	6.17 TargetObjects Discovery Query.....	53
245	6.17.1 Parameter \$className.....	54
246	6.17.2 Parameter \$propertyName.....	54
247	6.17.3 Example of TargetObjects Discovery Query.....	54
248	6.18 TargetObjectsInverseOf Discovery Query.....	54
249	6.18.1 Parameter \$className.....	55
250	6.18.2 Parameter \$propertyName.....	55
251	6.18.3 Example of TargetObjectsInverseOf Discovery Query.....	55
252	6.19 InverseRanges Discovery Query.....	55
253	6.19.1 Parameter \$className.....	56
254	6.19.2 Parameter \$propertyName.....	56
255	6.19.3 Example of InverseRanges Discovery Query.....	56
256	6.20 SymmetricProperties Discovery Query.....	57
257	6.20.1 Parameter \$className.....	57
258	6.20.2 Example of SymmetricProperties Discovery Query.....	57
259	6.21 FunctionalProperties Discovery Query.....	57
260	6.21.1 Parameter \$className.....	58
261	6.21.2 Example of FunctionalProperties Discovery Query.....	58
262	6.22 InverseFunctionalProperties Discovery Query.....	58
263	6.22.1 Parameter \$className.....	58
264	6.22.2 Example of InverseFunctionalProperties Discovery Query.....	58
265	6.23 Instances Discovery Query.....	59
266	6.23.1 Parameter \$className.....	59
267	6.23.2 Example of Instances Discovery Query.....	59
268	7 Canonical Metadata Definitions.....	61
269	7.1 ObjectType Extensions.....	61
270	7.2 AssociationType Extensions.....	61
271	7.3 Canonical Queries.....	64
272	7.3.1 All SuperProperties Discovery Query.....	64
273	7.3.2 Immediate SuperClass Discovery Query.....	64
274	7.3.3 Immediate SubClass Discovery Query.....	64
275	7.3.4 All SuperClasses Discovery Query.....	65
276	7.3.5 All SubClasses Discovery Query.....	65
277	7.3.6 EquivalentClasses Discovery Query.....	65
278	7.3.7 EquivalentProperties Discovery Query.....	66

279	7.3.8 SameExtrinsicObjects Discovery Query.....	66
280	7.3.9 DifferentExtrinsicObjects Discovery Query.....	67
281	7.3.10 AllDifferentRegistryObject Discovery Query.....	67
282	7.3.11 ObjectProperties Discovery Query.....	68
283	7.3.12 ImmediateInheritedObjectProperties Discovery Query.....	68
284	7.3.13 AllInheritedObjectProperties Discovery Query.....	69
285	7.3.14 DatatypeProperties Discovery Query.....	69
286	7.3.15 AllInheritedDatatypeProperties Discovery Query.....	69
287	7.3.16 TransitiveRelationships Discovery Query.....	69
288	7.3.17 TargetObjects Discovery Query.....	70
289	7.3.18 TargetObjectsInverseOf Discovery Query.....	70
290	7.3.19 InverseRanges Discovery Query.....	71
291	7.3.20 SymmetricProperties Discovery Query.....	72
292	7.3.21 FunctionalProperties Discovery Query.....	72
293	7.3.22 InverseFunctionalProperties Discovery Query.....	72
294	7.3.23 Instances Discovery Query Discovery Query.....	73
295	8 OWL Profile References.....	75
296	8.1 Normative References.....	75
297	8.2 Informative References.....	76
298	Appendix A.....	76
299		

Illustration Index

Figure 1: ebXML Registry Information Model, High Level Public View.....	15
Figure 2: ebXML Registry Information Model, Inheritance View.....	16

300

Index of Tables

301

1 Introduction

302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349

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

The ebXML Registry holds the metadata for the RegistryObjects and the documents pointed at by the RegistryObjects reside in an ebXML repository. The basic semantic mechanisms of ebXML Registry are classification hierarchies (ClassificationScheme) consisting of ClassificationNodes and the Association Types among RegistryObjects. Furthermore, RegistryObjects can be assigned properties through a slot mechanism and RegistryObjects can be classified using instances of Classification, ClassificationScheme and ClassificationNodes. Given these constructs, considerable amount of semantics can be defined in the registry.

However, currently semantics is becoming a much broader issue than it used to be since several application domains are making use of ontologies to add knowledge to their data and applications [StaabStuder]. 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 semantics in ebXML registries.

This document normatively defines the ebXML Registry profile for Web Ontology Language (OWL) Lite. More specifically, this document normatively specifies how OWL Lite constructs SHOULD be represented by ebXML RIM constructs **without causing any changes in the core ebXML Registry specifications [ebRIM], [ebRS]**. Furthermore, this document normatively specifies the code to process some of the OWL semantics through parameterized stored procedures that SHOULD be made available from the ebXML Registry.

These predefined stored queries provide the necessary means to exploit the enhanced semantics stored in the Registry. Hence, an application program does not have to develop additional code to process this semantics. In this way, it becomes possible to retrieve not only explicit but also the implied knowledge through queries, the enhancements to the registry are generic and also the registry specification is kept intact. The capabilities provided, move the semantics support beyond what is currently available in ebXML registries and it does so by using a standard ontology language.

Finally it is worth noting that ontologies can play two major roles: One is to provide a source of shared and precisely defined terms which can be used in formalizing knowledge and relationship among objects in a domain of interest. The other is to reason about the ontologies. When an ontology language like OWL is mapped to a class hierarchy like the one in ebXML, the first role can directly be achieved. Furthermore some implicit information can be obtained by predefined parameterized queries. However, when we want full reasoning power, we need reasoners. Yet, OWL reasoners can not directly run on the ebXML registry because all the registry information is not stored in OWL syntax.

Although this Profile is related to ebXML Registry specifications and not to any particular implementation, in order to be able to give concrete examples, the freebXML Registry implementation is used.

The document is organized as follows:

- Chapter 1 provides an introduction to the rest of this document.
- Chapter 2 provides an overview of the Web Ontology Language.
- Chapter 3 provides an overview of the ebXML Registry standard.
- Chapter 4 specifies the mapping between Web Ontology Language constructs and ebXML Registry Information Model.
- Chapter 5 describes the cataloging service for cataloging OWL content.
- Chapter 6 provides the discovery queries for a registry implementing this profile.
- Chapter 7 specifies the canonical metadata (such as object type extensions, new association types and the stored queries) defined by this profile.
- Chapter 8 provides normative and informative references that are used within or relevant to this document.

350 1.1 Terminology

351 The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT,
352 RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in IETF RFC
353 2119 [RFC2111].

354 The term “*repository item*” is used to refer to content (e.g., an XML document or a DTD) that resides in a
355 repository for storage and safekeeping. Each repository item is described by a RegistryObject instance.
356 The RegistryObject catalogs the RepositoryItem with metadata.

357 1.2 Conventions

358 Throughout the document the following conventions are employed to define the data structures used. The
359 following text formatting conventions are used to aide readability:

- 360 • UML Diagrams

361 UML diagrams are used as a way to concisely describe information models in a standard way. They
362 are not intended to convey any specific Implementation or methodology requirements.

- 363 • Identifier Placeholders

364 Listings may contain values that reference ebXML Registry objects by their id attribute. These id
365 values uniquely identify the objects within the ebXML Registry. For convenience and better readability,
366 these key values are replaced by meaningful textual variables to represent such id values.
367 For example, the following placeholder refers to the unique id defined for the canonical
368 ClassificationNode that defines the Organization ObjectType defined in [ebRIM]:

369

370

```
<id="{CANONICAL_OBJECT_TYPE_ID_ORGANIZATION}" >
```

371 1.3 Recommended Enhancements

372 In the current ebXML Registry implementation, when a stored query is submitted to the ebXML Registry, it
373 is stored in the “AdhocQuery” relational table without validation:

374 AdhocQuery (id, lid, objectType, status, versionName, comment_, queryLanguage, query);

375 When a user tries to invoke this stored query through a AdhocQuery, ebRS parses the stored query and
376 converts this stored query to the syntax acceptable by the underlying database. Furthermore currently
377 ebRS supports the SQL 92 [SQL 92] standard which does not include the “recursion” mechanisms. Also,
378 there seems to be problems in parsing queries involving UNION. Since some of the queries involved in
379 this Profile requires recursion and UNION mechanisms of SQL, it may help if ebRS is extended to support
380 SQL 99 standard [SQL 99].

2 OWL Overview

381

382 This chapter provides an overview of the Web Ontology Language [OWL]. Web Ontology Language
383 [OWL] is a semantic markup language for publishing and sharing ontologies on the World Wide Web.
384 OWL is derived from the DAML+OIL Web Ontology Language [DAML+OIL] and builds upon the Resource
385 Description Framework [RDF].

386 OWL provides three decreasingly expressive sublanguages [McGuinness, Harmelen]:

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

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

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

399 The list of OWL language constructs is as follows [McGuinness, Harmelen]:

2.1 Semantic Web Languages upon which OWL is Layered

400

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

410 Namespace information for these languages is given in the Table 1.

411

412 Table 1: Semantic Web namespace table

413

Commonly used Prefix	Namespace URI Reference
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
owl	http://www.w3.org/2002/07/owl#

414

415

416 The following section discusses elements of OWL along with a few elements of RDF and RDFS which are
417 most important to users of OWL. In this section Terms from RDF and RDFS vocabularies are
418 distinguished from OWL terms by the inclusion of the appropriate prefix as given in the Table 1.

419

420

421 2.2 OWL Lite Constructs

422 2.2.1 RDF Schema Features

- 423 • Class (Thing, Nothing)
- 424 • rdfs:subClassOf
- 425 • rdf:Property
- 426 • rdfs:subPropertyOf
- 427 • rdfs:domain
- 428 • rdfs:range
- 429 • Individual

430 2.2.2 (In)Equality

- 431 • equivalentClass
- 432 • equivalentProperty
- 433 • sameAs
- 434 • differentFrom
- 435 • AllDifferent
- 436 • distinctMembers

437 2.2.3 Property Characteristics

- 438 • ObjectProperty
- 439 • DatatypeProperty
- 440 • inverseOf
- 441 • TransitiveProperty
- 442 • SymmetricProperty
- 443 • FunctionalProperty
- 444 • InverseFunctionalProperty

445 2.2.4 Property Restrictions

- 446 • Restriction
- 447 • onProperty
- 448 • allValuesFrom
- 449 • someValuesFrom

450 2.2.5 Restricted Cardinality

- 451 • minCardinality (only 0 or 1)
- 452 • maxCardinality (only 0 or 1)
- 453 • cardinality (only 0 or 1)

454 2.2.6 Class Intersection

- 455 • intersectionOf

456 2.2.7 Versioning

- 457 • versionInfo
- 458 • priorVersion
- 459 • backwardCompatibleWith
- 460 • incompatibleWith
- 461 • DeprecatedClass
- 462 • DeprecatedProperty

463 2.2.8 Annotation Properties

- 464 • rdfs:label
- 465 • rdfs:comment
- 466 • rdfs:seeAlso
- 467 • rdfs:isDefinedBy
- 468 • AnnotationProperty
- 469 • OntologyProperty

470 2.2.9 Datatypes

- 471 • xsd datatypes

472 2.3 OWL DL Constructs

473 2.3.1 Class Axioms

- 474 • oneOf, dataRange
- 475 • disjointWith
- 476 • equivalentClass (applied to class expressions)
- 477 • rdfs:subClassOf (applied to class expressions)

478 2.3.2 Boolean Combinations of Class Expressions

- 479 • unionOf
- 480 • complementOf
- 481 • intersectionOf

482 2.3.3 Arbitrary Cardinality

- 483 • minCardinality
- 484 • maxCardinality
- 485 • cardinality

486 2.3.4 Filler Information

- 487 • hasValue

488

489

3 ebXML Registry Overview

490 This chapter provides an overview of ebXML Registry Information Model [ebRIM] and an overview of the
491 specific domain and/or application.

492 The [ebRIM] is the target for the mapping patterns defined by this document.

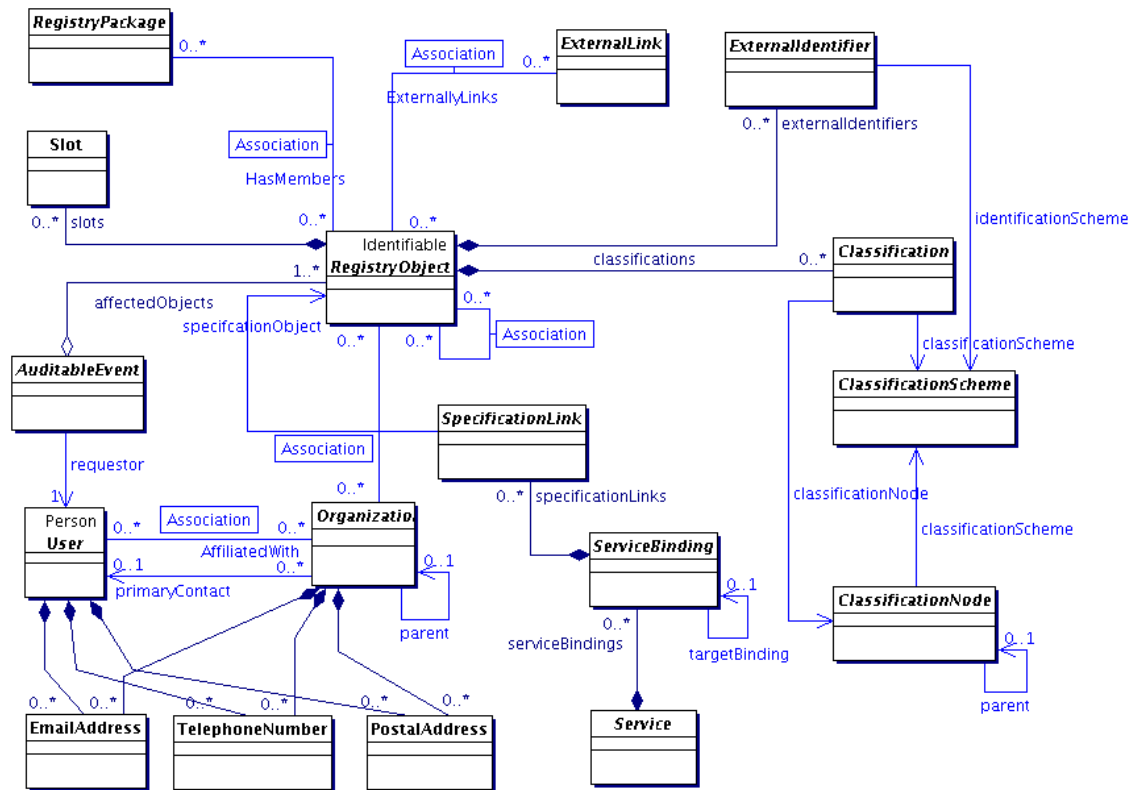
493 The information presented is informative and is not intended to replace the normative information defined
494 by ebXML Registry.

3.1 Overview of [ebRIM]

496 This section is provided in the « Deployment Profile Template for ebXML V3 specs » and can be removed
497 in a specific profile.

498 Normally only specifics topics needs to be developed here (but the profile editor can prefer to leave it)

499 This section summarizes the ebXML Registry Information Model [ebRIM]. This model is the target of the
500 mapping defined in this document. The reader SHOULD read [CMRR] for a more detailed overview of
501 ebXML Registry as a whole.



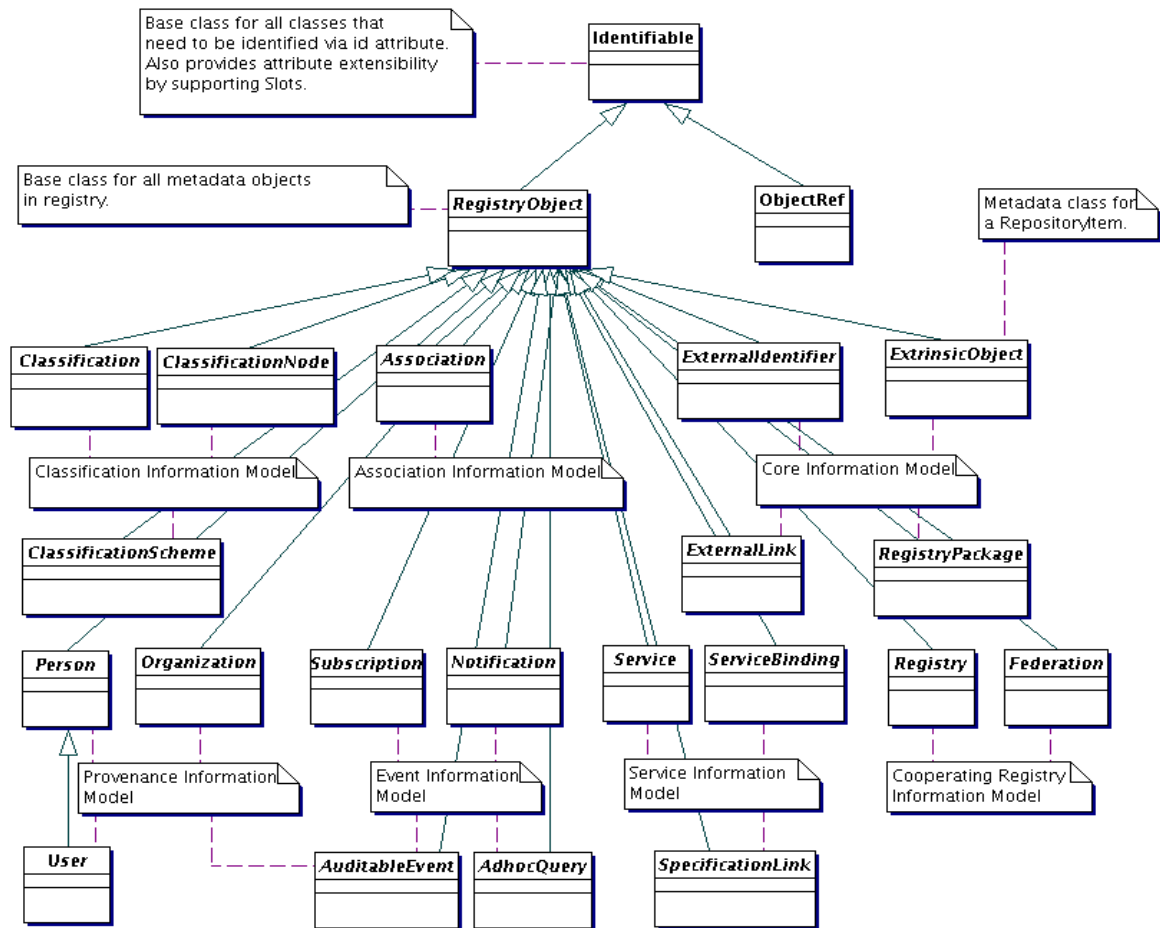
503

Figure 1: ebXML Registry Information Model, High Level Public View

504

505 The ebXML registry defines a Registry Information Model [ebRIM] that specifies the standard metadata
506 that may be submitted to the registry. Figure 1 presents the UML class diagram representing the Registry
507 Information Model. Figure 2, shows the inheritance relationships in among the classes of the ebXML
508 Registry Information Model.

509



511 **Figure 2: ebXML Registry Information Model, Inheritance View**

512 The next few sections describe the main features of the information model.

513 3.1.1 RegistryObject

514 This is an abstract base class used by most classes in the model. It provides minimal
 515 metadata for registry objects. The following sections use the Organization sub-class of RegistryObject as
 516 an example to illustrate features of the model.

517 3.1.2 Object Identification

518 A RegistryObject has a globally unique id which is a UUID based URN:

```
519
520 <rim:Organization id="urn:uuid:dafa4da3-1d92-4757-8fd8-ff2b8ce7a1bf" >
```

521 **Listing 1: Example of id attribute**

522 The id attribute value MAY potentially be human friendly but MUST be a unique ID value within the
 523 registry.

```
524
525 <rim:Organization id="uurn:oasis:Organization">
```

526 **Listing 2: Example of human friendly id attribute**

527 Since a RegistryObject MAY have several versions, a logical id (called lid) is also defined which is unique
 528 for different logical objects. However the lid attribute value MUST be the same for all versions of the same

529 logical object. The lid attribute value is a URN that, as well for id attribute, MAY potentially be human
530 friendly:

531

```
532 <rim:Organization id=${ACME_ORG_ID}  
533   lid="urn:acme:ACMEOrganization">
```

534

Listing 3: Example of lid Attribute

535 A RegistryObject MAY also have any number of ExternalIdentifiers which may be any string value within
536 an identified ClassificationScheme.

537

```
538 <rim:Organization id=${ACME_ORG_ID}  
539   lid="urn:acme:ACMEOrganization">  
540  
541   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}  
542     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}  
543     value="ACME"/>  
544   </rim:ExternalIdentifier>  
545  
546 </rim:Organization>
```

547

Listing 4: Example of ExternalIdentifier

548 3.1.3 Object Naming and Description

549 A RegistryObject MAY have a name and a description which consists of one or more strings in one or
550 more local languages. Name and description need not be unique across RegistryObjects.

551

```
552 <rim:Organization id=${ACME_ORG_ID}  
553   lid="urn:acme:ACMEOrganization">  
554  
555   <rim:Name>  
556     <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>  
557   </rim:Name>  
558   <rim:Description>  
559     <rim:LocalizedString value="ACME is a provider of Java software."  
560       xml:lang="en-US"/>  
561   </rim:Description>  
562  
563   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}  
564     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}  
565     value="ACME"/>  
566   </rim:ExternalIdentifier>  
567 </rim:Organization>
```

568

Listing 5: Example of Name and Description

569

570 3.1.4 Object Attributes

571 For each class in the model, [ebRIM] defines specific attributes. Examples of several of these attributes
572 such as id, lid, name and description have already been introduced.

573 3.1.4.1 Slot Attributes

574 In addition the model provides a way to add custom attributes to any RegistryObject instance using
575 instances of the Slot class. The Slot instance has a Slot name which holds the attribute name and MUST
576 be unique within the set of Slot names in that RegistryObject. The Slot instance also has a ValueList that
577 is a collection of one or more string values.

578 The following example shows how a custom attribute named "urn:acme:slot:NASDAQSymbol" and value
579 "ACME" MAY be added to a RegistryObject using a Slot instance.

580

```
581 <rim:Organization id=${ACME_ORG_ID}  
582   lid="urn:acme:ACMEOrganization">
```

```

583 <rim:Slot name="urn:acme:slot:NASDAQSymbol">
584   <rim:ValueList>
585     <rim:Value>ACME</rim:Value>
586   </rim:ValueList>
587 </rim:Slot>
588
589   <rim:Name>
590     <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
591   </rim:Name>
592   <rim:Description>
593     <rim:LocalizedString value="ACME makes Java. Provider of free Java
594 software."
595     xml:lang="en-US"/>
596   </rim:Description>
597   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
598     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
599     value="ACME"/>
600   </rim:ExternalIdentifier>
601 </rim:Organization>
602

```

Listing 6: Example of a Dynamic Attribute Using Slot

3.1.5 Object Classification

Any RegistryObject may be classified using any number of Classification instance. A Classification instance references an instance of a ClassificationNode as defined by [ebRIM]. The ClassificationNode represents a value within the ClassificationScheme. The ClassificationScheme represents the classification taxonomy.

```

610 <rim:Organization id=${ACME_ORG_ID}
611   lid="urn:acme:ACMEOrganization">
612   <rim:Slot name="urn:acme:slot:NASDAQSymbol">
613     <rim:ValueList>
614       <rim:Value>ACME</rim:Value>
615     </rim:ValueList>
616   </rim:Slot>
617   <rim:Name>
618     <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
619   </rim:Name>
620   <rim:Description>
621     <rim:LocalizedString value="ACME makes Java. Provider of free Java
622 software." xml:lang="en-US"/>
623   </rim:Description>
624   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
625     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
626     value="ACME"/>
627   </rim:ExternalIdentifier>
628
629   <!--Classify Organization as a Software Publisher using NAICS Taxonomy-->
630   <rim:Classification id=${CLASSIFICATION_ID}
631     classificationNode=${NAICS_SOFTWARE_PUBLISHER_NODE_ID}
632     classifiedObject=${ACME_ORG_ID}>
633
634 </rim:Organization>

```

Listing 7: Example of Object Classification

3.1.6 Object Association

Any RegistryObject MAY be associated with any other RegistryObject using an Association instance where one object is the sourceObject and the other is the targetObject of the Association instance. An Association instance MAY have an associationType which defines the nature of the association.

There are a number of predefined Association Types that a registry must support to be [ebRIM] compliant. These canonical association types are defined as a *ClassificationScheme* called AssociationType. The SubmitObjectsRequest document of the AssociationType Classification scheme is available at:

http://www.oasis-open.org/committees/regrep/documents/3.0/canonical/SubmitObjectsRequest_AssociationTypeScheme.x

645 ml

646 [ebRIM] allows this scheme to be extensible.

647 The following example shows an Association between the ACME Organization instance and a Service
648 instance with the associationType of "OffersService". This indicates that ACME Organization offers the
649 specified service (Service instance is not shown).

650

```
651 <rim:Association  
652     id=${ASSOCIATION_ID}  
653     associationType=${CANONICAL_ASSOCIATION_TYPE_OFFERS_SERVICE_ID}  
654     sourceObject=${ACME_ORG_ID}  
655     targetObject=${ACME_SERVICE1_ID}/>
```

656

Listing 8: Example of Object Association

657 3.1.7 Object References To Web Content

658 Any RegistryObject MAY reference web content that are maintained outside the registry using association
659 to an ExternalLink instance that contains the URL to the external web content. The following example
660 shows the ACME Organization with an Association to an ExternalLink instance which contains the URL to
661 ACME's web site. The associationType of the Association MUST be of type "ExternallyLinks" as defined
662 by [ebRIM].

663

```
664 <rim:ExternalLink externalURI="http://www.acme.com"  
665     id=${ACME_WEBSITE_EXTERNAL_ID}>  
666 <rim:Association  
667     id=${EXTERNALLYLINKS_ASSOCIATION_ID}  
668     associationType=${CANONICAL_ASSOCIATION_TYPE_EXTERNALLY_LINKS_ID}  
669     sourceObject=${ACME_WEBSITE_EXTERNAL_ID}  
670     targetObject=${ACME_ORG_ID}/>
```

671

Listing 9: Example of Reference to Web Content Using ExternalLink

672 3.1.8 Object Packaging

673 RegistryObjects may be packaged or organized in a hierarchical structure using a familiar file and folder
674 metaphor. RegistryPackage instances serve as folders while RegistryObject instances serve as files in
675 this metaphor. A RegistryPackage instances groups logically related RegistryObject instances together as
676 members of that RegistryPackage.

677 The following example creates a RegistryPackage for Services offered by ACME Organization organized
678 in RegistryPackages according to the nature of the Service. Each Service is referenced using the
679 ObjectRef type defined by [ebRIM].

680

```
681 <rim:RegistryPackage  
682     id=${ACME_SERVICES_PACKAGE_ID}>  
683     <rim:RegistryObjectList>  
684         <rim:ObjectRef id=${ACME_SERVICE1_ID}>  
685             <rim:RegistryPackage  
686                 id=${ACME_PURCHASING_SERVICES_PACKAGE_ID}>  
687                 <rim:ObjectRef id=${ACME_PURCHASING_SERVICE1_ID}>  
688                 <rim:ObjectRef id=${ACME_PURCHASING_SERVICE2_ID}>  
689             </rim:RegistryPackage>  
690             <rim:RegistryPackage  
691                 id=${ACME_HR_SERVICES_PACKAGE_ID}>  
692                 <rim:ObjectRef id=${ACME_HR_SERVICE1_ID}>  
693                 <rim:ObjectRef id=${ACME_HR_SERVICE2_ID}>  
694             </rim:RegistryPackage>  
695         </rim:RegistryObjectList>  
696     </rim:RegistryPackage>
```

697

Listing 10: Example of Object Packaging Using RegistryPackages

698 3.1.9 ExtrinsicObject

699 ExtrinsicObjects provide metadata that describes submitted content whose type is not intrinsically known
700 to the registry and therefore MUST be described by means of additional attributes (e.g., mime type).
701 Examples of content described by ExtrinsicObject include Collaboration Protocol Profiles, Business
702 Process descriptions, and schemas.

703 3.1.10 Service Description

704 Service description MAY be defined within the registry using the Service, ServiceBinding and
705 SpecificationLink classes defined by [ebRIM]. This MAY be used to publish service descriptions such as
706 WSDL and ebXML CPP/A.

707 3.2 Overview of [ebRS]

708 The [ebRS] specification defines the interfaces supported by an ebXML Registry and their bindings to
709 protocols such as SOAP and HTTP.

4 Representing OWL Lite Constructs in ebRIM

It is important to note that although the mapping described in this section is complex, this complexity is hidden from the ebXML registry user because the needed stored queries MUST already be available in the Registry as described in Chapter 6. As this profile aims to enhance ebXML registry semantics without causing any changes in the core ebXML Registry architecture specification [ebRIM], [ebRS], the stored queries proposed in this specification SHOULD be submitted to the ebXML Registry by using the Stored Query API of [ebRS].

The following ebRIM standard relational schema is used in coding the stored queries throughout this document.

```
ClassScheme (id, home, lid, objectType, status, versionName, comment_,...);
ClassificationNode(accessControlPolicy, id, lid, home, objectType, code, parent,
path,versionName, comment_...)
Association(accessControlPolicy, id, lid, home, objectType, associationType,
sourceObject, targetObject, isConfirmedBySourceOwner,versionName, comment_
isConfirmedByTargetOwner,...)
Name_(charset, lang, value, parent,...)
Classification (id, objectType, lid, home, classificationNode, versionName,
comment_, classificationScheme, classifiedObject, nodeRepresentation,...);
ExtrinsicObject (id, lid, home, objectType,...)
```

ebXML Registry Relations

Detailed explanation on how to represent some of the OWL Lite constructs in ebRIM is available from [Dogac, et. al. 2005]. Furthermore, [Dogac et. al. 2006] provides an implementation using the work presented in this document for healthcare applications.

4.1 Representing RDF Schema Features in ebRIM

4.1.1 owl:Class → rim:ClassificationNode

An owl:Class MUST be mapped to a rim:ClassificationNode as shown in the following examples:

```
<owl:Class rdf:ID="City">
</owl:Class>
```

Example owl:Class

```
<ClassificationScheme id=${GeographicalEntity}
name="GeographicalEntity"/>
<rim:ClassificationNode id=${City} code='City'
parent=${GeographicalEntity}>
</rim:ClassificationNode>
```

Example Corresponding ebRIM construct ClassificationNode

A ClassificationScheme should be created for each ontology, and the classes belonging to this ontology should be represented as the ClassificationNodes under this ClassificationScheme.

4.1.2 rdf:Property → rim:Association Type HasProperty

A new ebRIM Association Type called "HasProperty" MUST be defined. The domain of an rdf:Property, rdfs:domain, is the sourceObject in this Association Type and the range of an rdf:Property which is

759 rdfs:range, is the targetObject of the Association Type. Consider the following example which defines an
760 rdf:Property instance called "hasAirport" whose domain is "City" and whose range is "Airport" classes:

761

```
762 <rdf:Property rdf:ID="hasAirport">  
763   <rdfs:domain rdf:resource="#City"/>  
764   <rdfs:range rdf:resource="#AirPort"/>  
765 </rdf:Property>
```

766

Example rdf:Property

767

```
768 <rim:Association id=${hasAirport}  
769 associationType='urn:oasis:names:tc:ebxml-  
770 regrep:profile:webontology:AssociationType:OWL:HasProperty'  
771   sourceObject= ${city}  
772   targetObject=${Airport} >  
773 </rim:Association>
```

774

Example: ebRIM construct Association corresponding to rdf:Property

775 OWL specializes RDF Property to owl:ObjectProperty and owl:DatatypeProperty which are discussed in
776 the sections 4.3.1 and 4.3.2.

777 4.1.3 rdfs:subPropertyOf → rim:Association Type SubPropertyOf

778 In OWL, properties can be organized into property hierarchies by declaring a property to be a
779 subPropertyOf another property. As shown in the following example, "creditCardPayment" property may
780 be a "subPropertyOf" the property "paymentMethods":

781

```
782 <rdf:Property rdf:ID="creditCardPayment">  
783   <rdfs:subPropertyOf rdf:Resource="#paymentMethods"/>  
784 </rdf:Property>
```

785

Example rdfs:subPropertyOf

786 A new ebXML RIM Association Type called "SubPropertyOf" MUST be defined to represent
787 rdfs:subPropertyOf in ebRIM.

788

789 To express this semantics through ebXML RIM constructs, "creditCardPayment" Association is
790 associated with the "paymentMethods" the newly created "SubPropertyOf" ebXML Association Type as
791 shown in the following:

792

```
793 <rim:Association id=${subPropertyOfID}  
794 associationType='urn:oasis:names:tc:ebxml-  
795 regrep:profile:webontology:AssociationType:OWL:SubPropertyOf '  
796   sourceObject= ${creditCardPayment} targetObject=${paymentMethods} >  
797 </rim:Association>
```

798 Such a semantic enhancement brings the following processing need: given a property, it should be
799 possible to retrieve all of its super properties as described in Section 6.1.

800 4.1.4 rdfs:subClassOf → rim:Association Type SubClassOf

801 OWL relies on RDF Schema for building class hierarchies through the use of "rdfs:subClassOf" property
802 and allows multiple inheritance. In ebXML, a class hierarchy is represented by a ClassificationScheme. A
803 ClassificationScheme is constructed by connecting a ClassificationNode to its super class by using the
804 "parent" attribute of the ClassificationNode. However it is not possible to associate a ClassificationNode
805 with more than one different super classes by using "parent" attribute. In other words, an ebXML Class
806 hierarchy has a tree structure and therefore is not readily available to express multiple inheritance. There
807 is a need for additional mechanisms to express multiple inheritance in ebXML RIM. Therefore, a new
808 Association Type called "SubClassOf" MUST be defined in the Registry.

809 In the following OWL example, "AirReservationServices" service inherits both from "AirServices" service
810 and OWL-S ServiceProfile class.

```
811  
812 <owl:Class rdf:ID="AirReservationServices">  
813 <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-  
814 s/1.0/Profile.owl#Profile"/>  
815 <rdfs:subClassOf rdf:resource="#AirServices"/>  
816 </owl:Class>
```

817 Example rdfs:subClassOf

818 To express this semantics through ebXML RIM constructs, "AirReservationServices" ClassificationNode is
819 associated both with the "OWL-S Profile" and "AirServices" ClassificationNodes through the "targetObject"
820 and "sourceObject" attributes of the two instances of the newly created "SubClassOf" ebXML Association
821 Type as shown in the following:

```
822  
823 <rim:Association id=${subClassOf}  
824 associationType='urn:oasis:names:tc:ebxml-  
825 regrep:profile:webontology:AssociationType:OWL:SubClassOf'  
826 sourceObject= ${AirReservationServices} targetObject=${OWL-S_Profile} >  
827 </rim:Association>  
828 <rim:Association id=${subClassOf2}  
829 associationType='urn:oasis:names:tc:ebxml-  
830 regrep:profile:webontology:AssociationType:OWL:SubClassOf'  
831 sourceObject= ${AirReservationServices} targetObject=${AirServices} >  
832 </rim:Association>
```

833 Once such a semantics is defined, there is a need to process the objects in the registry according to the
834 semantics implied; that is, given a class, it should be possible to retrieve all of its subclasses and/or all of
835 its super classes. By making the required adhoc queries available in the registry, this need can be readily
836 served as described in Sections 6.2, 6.3, 6.4 and 6.5.

837 4.1.5 owl:Individual → rim:ExtrinsicObject

838 A class in OWL defines a group of individuals that belong together because they share some properties
839 [McGuinness, Harmelen]. For example, "TravelService" class may have the property "paymentMethod"
840 whose range may be "PossiblePaymentMethods" class as shown in the following example:

```
841  
842 <owl:Class rdf:ID="TravelWebService">  
843 </owl:Class>  
844  
845 <owl:ObjectProperty rdf:ID="paymentMethod">  
846 <rdfs:domain rdf:resource="#TravelWebService"/>  
847 <rdfs:range rdf:resource="#PossiblePaymentMethods"/>  
848 </owl:ObjectProperty >
```

849 Example owl:Class example

850 In OWL, individuals are instances of classes. For example, an instance of "TravelWebService" class may
851 be "MyTravelWebService". Properties may be used to relate one individual to another. For example,
852 "MyTravelService" inherits "paymentMethod" property and this property may map to an instance of
853 "PossiblePaymentMethods" class, such as "Cash" as shown in the following example:

```
854  
855 <TravelWebService rdf:ID="MyTravelWebService">  
856 <paymentMethod> Cash </paymentMethod>  
857 </TravelWebService>
```

858 Example owl:Individual example

859 In ebXML Registry the class instances can be stored in the Registry or in the Repository. This profile
860 recommends to store class instances in the Repository and to describe their metadata through
861 ExtrinsicObjects in the Registry.

862 4.2 Representing OWL (In)Equality Constructs in ebXML RIM

863 4.2.1 owl:equivalentClass, owl:equivalentProperty → rim:Association Type 864 EquivalentTo

865 In ebXML, the predefined "EquivalentTo" Association Type expresses the fact that the source
866 RegistryObject is equivalent to target RegistryObject. Therefore, "EquivalentTo" association MUST be
867 used to express "owl:equivalentClass" and "owl:equivalentProperty" properties since classes and
868 properties are all ebXML RegistryObjects.

869 The adhoc query for retrieving all the equivalent classes of a given ClassificationNode is represented in
870 Section 6.6. Additionally the adhoc query to retrieve all the equivalent properties (Association Type) of a
871 given property (Association Type) is presented in Section 6.7

872 4.2.2 owl:sameAs → rim:Association Type SameAs

873 ebXML Registry contains the metadata of the objects stored in the repository. This profile recommends
874 that the instances to be stored in repository and to be represented through "ExtrinsicObjects" in the
875 registry.

876 owl:sameAs construct is used to indicate that two instances in a knowledge base are the same. This
877 construct may be used to create a number of different names that refer to the same individual.

878

```
879 <rdf:Description rdf:about="#MyAirReservationService">  
880 <owl:sameAs rdf:resource="#THYAirReservationService"/>  
881 </rdf:Description>
```

882

Example owl:sameAs

883 This translates into two "ExtrinsicObjects" in the ebXML registry to be the same. For this purpose a new
884 Association Type called "SameAs" MUST be defined in the ebXML registry.

```
885 <rim:Association id=${sameAs1} associationType='urn:oasis:names:tc:ebxml-  
886 regrep:profile:webontology:AssociationType:OWL:SameAs'  
887 sourceObject=${MyAirReservationService} targetObject=${THYAirReservationService}  
888 >  
889 </rim:Association>
```

890

Example Corresponding ebRIM construct Association

891

892 Furthermore, the adhoc query presented in Section 6.8 MUST be available in the registry to retrieve all
893 the "ExtrinsicObjects" defined to be the same with a given ExtrinsicObject.

894 4.2.3 owl:differentFrom → rim:Association Type DifferentFrom

895 owl:differentFrom construct is used to indicate that two instances in a knowledge base are different from
896 one another. Explicitly stating that individuals are different can be important when using languages such
897 as OWL (and RDF) that do not assume that individuals have one and only one name [McGuinness,
898 Harmelen].

899

```
900 <rdf:Description rdf:about="#MyAirReservationService">  
901 <owl:differentFrom rdf:resource="#THYAirReservationService"/>  
902 </rdf:Description>
```

903

Example owl:differentFrom

904 This translates into declaring two "ExtrinsicObjects" in the ebXML registry to be different from each other.
905 For this purpose a new Association Type "DifferentFrom" MUST be defined in the ebXML registry to
906 explicitly indicate that the sourceRegistryObject is different from the targetRegistryObject.

```

907 <rim:Association id=${differentFrom1}
908 associationType='urn:oasis:names:tc:ebxml-
909 regrep:profile:webontology:AssociationType:OWL:DifferentFrom'
910 sourceObject= ${MyAirReservationService}
911 targetObject=${THYAirReservationService} >
912 </rim:Association>

```

Example Corresponding ebRIM construct Association

The adhoc query presented in Section 6.9 can be used to process this semantics.

4.2.4 owl:AllDifferent

owl:AllDifferent is a special built-in OWL class, for which the property owl:distinctMembers is defined, which links an instance of owl:AllDifferent to a list of individuals. The AllDifferent construct is particularly useful when there are sets of distinct objects and when modelers are interested in enforcing the unique names assumption within those sets of objects [McGuinness, Harmelen].

The following example states that the three instances of the "WebService" collection are all different from one another:

```

922 <owl:AllDifferent>
923   <owl:distinctMembers rdf:parseType="Collection">
924     <WebService rdf:about="#MyCarService"/>
925     <WebService rdf:about="#MyFlightService"/>
926     <WebService rdf:about="#MyHotelService"/>
927   </owl:distinctMembers>
928 </owl:AllDifferent>

```

Example owl:AllDifferent

owl:AllDifferent SHOULD be represented in ebRIM as follows: the RegistryObjects under consideration SHOULD be grouped as a RegistryPackage whose id is \${Collection}. Then the RegistryObjects in the collection MUST be associated with this RegistryPackage with "HasMember" Association Type. One slot of the registry package MUST be used to indicate that all members are different.

```

934
935 <rim:RegistryPackage id = ${Collection} >
936   <rim:Slot name=urn:oasis:names:tc:ebxml-
937 regrep:profile:webontology:slot:packagetype>
938     <rim:ValueList>
939       <rim:Value>allDifferent</rim:Value>
940     </rim:ValueList>
941   </rim:Slot>
942 </rim:RegistryPackage>
943 <rim:Association id = ${HasMemberRegistryPackageAssoc1}
944 associationType = "urn:oasis:names:tc:ebxml-
945 regrep:AssociationType:HasMember" sourceObject = ${Collection}
946 targetObject = ${MyCarService} />
947 <rim:Association id = ${HasMemberRegistryPackageAssoc2}
948 associationType = "urn:oasis:names:tc:ebxml-regrep:HasMember"
949 sourceObject = ${Collection}
950 targetObject = ${MyFlightService} />
951
952 <rim:Association id = ${HasMemberRegistryPackageAssoc3}
953 associationType = "urn:oasis:names:tc:ebxml-regrep:HasMember"
954 sourceObject = ${Collection}
955 targetObject = ${MyHotelService} />

```

Example Corresponding ebRIM Representation

The adhoc query presented in Section 6.10 can be used to process this semantics.

958 4.3 Representing OWL Property Characteristics in ebRIM

959 4.3.1 owl:ObjectProperty → rim:Association Type objectProperty

960 To represent OWL ObjectProperty in ebXML, a new type of Association called "ObjectProperty" MUST be
961 defined. Consider the following example which defines an object property "hasAirport" whose domain is
962 "City" and whose range is "Airport":

963

```
964 <owl:ObjectProperty rdf:ID="hasAirport">  
965   <rdfs:domain rdf:resource="#City"/>  
966   <rdfs:range rdf:resource="#AirPort"/>  
967 </owl:ObjectProperty>
```

968

Example owl:ObjectProperty

969

```
970 <rim:Association id=${hasAirport} associationType='urn:oasis:names:tc:ebxml-  
971   regrep:profile:webontology:AssociationType:OWL:ObjectProperty'  
972   sourceObject= ${City} targetObject=${Airport} >  
973 </rim:Association>
```

974

Example Corresponding ebRIM construct Association

975 Once such objectProperty definitions are stored in the ebXML registry, they can be retrieved through
976 ebXML query facilities by the user. The adhoc queries presented in Section 6.11 and 6.12 MUST be
977 available in the registry to facilitate this access.

978 4.3.2 owl:DatatypeProperty → rim:Association Type DatatypeProperty

979 Similarly, to represent OWL DatatypeProperty in ebXML, a new Association Type called
980 "DatatypeProperty" MUST be defined. Consider the following example which defines an datatype property
981 "hasPrice" whose domain is the "AirReservationServices" and whose range is "XMLSchema
982 nonNegativeInteger". How OWL XML Schema types are handled in ebXML RIM is described in Section
983 4.9.

```
984 <owl:DatatypeProperty rdf:ID="hasPrice">  
985   <rdfs:domain rdf:resource="#AirReservationServices"/>  
986   <rdfs:range  
987   rdf:resource="http://www.w3.org/2001/XMLSchema/nonNegativeInteger"/>  
988 </owl:DatatypeProperty>
```

989

Example owl:DatatypeProperty

990

```
991 <rim:Association id=${hasPrice}  
992   associationType='urn:oasis:names:tc:ebxml-  
993   regrep:profile:webontology:AssociationType:OWL:DatatypeProperty'  
994   sourceObject= ${AirReservationServices}  
995   targetObject=urn:www.w3.org:2001/XMLSchema:nonNegativeInteger >  
996 </rim:Association>
```

997

Example Corresponding ebRIM construct Association

998 The adhoc query presented in Section 6.14 MUST be available in the registry to facilitate the direct access
999 to datatype properties of a given classification node.

1000 4.3.3 owl:TransitiveProperty → rim:Association Type TransitiveProperty

1001 In OWL, if a property, P, is specified as transitive then for any x, y, and z:P(x,y) and P(y,z) implies P(x,z)
1002 [McGuinness, Harmelen]. Transitive property is a subproperty of ObjectProperty and MUST be defined as
1003 a new Association Type called "TransitiveProperty" in ebRIM.

1004 Consider the following example where "succeeds" is defined as a transitive property of
1005 "TravelWebService" class:

1006

```

1007 <owl:ObjectProperty rdf:ID="succeeds">
1008   <rdf:type rdf:resource="#owl:TransitiveProperty" />
1009   <rdfs:domain rdf:resource="#TravelWebService" />
1010   <rdfs:range rdf:resource="#TravelWebService" />
1011 </owl:ObjectProperty>

```

1012 **Example owl:TransitiveProperty**

```

1013
1014 <rim:Association id=${succeeds}
1015 associationType='urn:oasis:names:tc:ebxml-
1016 regrep:profile:webontology:AssociationType:OWL:TransitiveProperty'
1017 sourceObject= ${TravelWebService} targetObject=${TravelWebService} >
1018 </rim:Association>

```

1019 **Example Corresponding ebRIM construct Association**

1020 Assume the following two definitions which declare three Web service instances from TravelWebService
 1021 class where "MyHotelAvailabilityService" service succeeds "MyAirReservationService" and
 1022 "MyInsuranceService" succeeds "MyHotelAvailabilityService". Since "succeeds" is a transitive property, it
 1023 follows that "MyInsuranceService" succeeds "MyAirReservationService" although this fact is not explicitly
 1024 stated.

```

1025
1026 <TravelWebService rdf:ID="MyHotelAvailabilityService">
1027   <succeeds rdf:resource="#MyAirReservationService" />
1028 </TravelWebService>
1029
1030 <TravelWebService rdf:ID="MyInsuranceService">
1031   <succeeds rdf:resource="#MyHotelAvailabilityService" />
1032 </TravelWebService>

```

1033 **Example owl:TransitiveProperty instances**

1034 To make any use of this transitive property in ebXML registries, coding is necessary to find out the implied
 1035 information. The adhoc query presented in Section 6.16 MUST be available in the registry to handle this
 1036 semantics.

1037 **4.3.4 owl:inverseOf → rim:Association Type InverseOf**

1038 In OWL, one property may be stated to be the inverse of another property. If the property P1 is stated to
 1039 be the inverse of the property P2, then if X is related to Y by the P2 property, then Y is related to X by the
 1040 P1 property [McGuinness, Harmelen].

1041 Consider, for example, the "succeeds" property defined in Section 4.3.3. To denote that a certain Web
 1042 service instance precedes another during execution, we may define the "precedes" property as an inverse
 1043 of the "succeeds" property as follows:

```

1044
1045 <owl:ObjectProperty rdf:ID="precedes">
1046   <owl:inverseOf rdf:resource="#succeeds" />
1047 </owl:ObjectProperty>

```

1048 **Example owl:inverseOf Property**

```

1049
1050 <rim:Association id=${inverseOf1}
1051 associationType='urn:oasis:names:tc:ebxml-
1052 regrep:profile:webontology:AssociationType:OWL:InverseOf'
1053 sourceObject= ${precedes} targetObject=${succeeds} >
1054 </rim:Association>

```

1055 **Example Corresponding ebRIM construct Association**

1056 Assume that we want to find all the Web services which can succeed a given Web service. In such a
 1057 case, we need not only find all the Web services which succeeds this given Web service, that is the target
 1058 objects of "succeeds" Association instance, but we also need to find all the sourceObjects of the
 1059 "precedes" Association instance since "precedes" is declared to be the "inverseOf" succeeds Association

1060 instance. This can be achieved through the adhoc query presented in Section 6.19.

1061 Alternatively, one might use the additional semantics that this profile supports would be to cause inferred
1062 information to be produced and stored along with new data as that new data was inserted into the reg/rep.
1063 There is a trade off here: in this way, the extra work of inferring is only done at insertion/update time,
1064 instead of at query time. However, an insertion or an update will require all the inferred data to be inserted
1065 whether it will be used or not and hence will cause considerable maintenance overhead.

1066 4.3.5 owl:SymmetricProperty → rim:Association Type SymmetricProperty

1067 In OWL, if a property is symmetric, then if the pair (x,y) is an instance of the symmetric property P, then
1068 the pair (y,x) is also an instance of P [McGuinness, Harmelen]. Symmetric property is a subproperty of
1069 ObjectProperty in OWL. Consider the OWL class “WebService” and the “complements” symmetric
1070 property:

```
1071 <owl:Class rdf:ID="WebService">  
1072   <rdfs:subClassOf  
1073     rdf:resource="http://www.w3.org/2000/01/rdfschema#Resource"/>  
1074 </owl:Class>  
1075 <owl:SymmetricProperty rdf:ID="complements">  
1076   <rdfs:domain rdf:resource="#WebService"/>  
1077   <rdfs:range rdf:resource="#WebService"/>  
1078 </owl:SymmetricProperty>
```

1079 Example owl:SymmetricProperty

```
1080 <rim:Association id=${complements}  
1081 associationType='urn:oasis:names:tc:ebxml-  
1082 regrep:profile:webontology:AssociationType:OWL:SymetricProperty'  
1083 sourceObject= ${WebService} targetObject=${WebService} >  
1084 </rim:Association>
```

1085 Example Corresponding ebRIM construct Association

1086 Given that HotelReservationWebService complements AirReservationWebService, it is possible to
1087 deduce that AirReservationWebService complements HotelReservationWebService.

1088 owl:SymmetricProperty MUST be defined as a new type of Association in ebRIM called
1089 “SymmetricProperty”. Furthermore the adhoc query presented in Section 6.20 MUST be available in the
1090 Registry to retrieve symmetric Associations of a ClassificationNode.

1091 4.3.6 owl:FunctionalProperty → rim:Association Type FunctionalProperty

1092 In OWL, if a property is a FunctionalProperty, then it has no more than one value for each individual (it
1093 may have no values for an individual) [McGuinness, Harmelen]. The range of a FunctionalProperty can be
1094 either an Object or a datatype. Consider, for example, the “hasPrice” Functional property which has a
1095 unique price:

```
1096 <owl:DatatypeProperty rdf:ID="hasPrice">  
1097   <rdf:type rdf:resource="&owl;FunctionalProperty" />  
1098   <rdfs:domain rdf:resource="#AirReservationServices"/>  
1099   <rdfs:range  
1100     rdf:resource="http://www.w3.org/2001/XMLSchema/nonNegativeInteger"/>  
1101 </owl:DatatypeProperty>
```

1102 Example owl:FunctionalProperty

```
1103 <rim:Association id=${hasPrice}  
1104 associationType='urn:oasis:names:tc:ebxml-  
1105 regrep:profile:webontology:AssociationType:OWL:FunctionalProperty'  
1106 sourceObject= ${AirReservationServices}  
1107 targetObject=${uurn:www.w3.org:2001/XMLSchema:nonNegativeInteger} >  
1108 </rim:Association>
```

1109 Example Corresponding ebRIM construct Association

1110 ebXML RIM MUST contain a new Association Type called “FunctionalProperty” to express this semantics.
1111 Furthermore the he adhoc query presented in Section 6.21 MUST be available in the Registry to retrieve

1112 functional Associations of a ClassificationNode.

1113 4.3.7 owl:InverseFunctionalProperty → rim:Association Type 1114 InverseFunctionalProperty

1115 In OWL, if a property is inverse functional then the inverse of the property is functional. Thus the inverse
1116 of the property has at most one value for each individual [McGuinness, Harmelen]. InverseFunctional
1117 properties (IFPs) are like keys. An individual filling the range role in an inverseFunctional property
1118 instance identifies the individual in the domain role of that same property instance. In other words, if a
1119 semantic web tool encounters two individuals with the same value for an inverseFunctional property, it
1120 can be inferred that they are actually the same individual.

1121 As an example, the ObjectProperty “finalDestination” indicates that each flight arrives to only one airport
1122 as its final destination.

```
1123 <owl:ObjectProperty rdf:ID="finalDestination">  
1124   <rdf:type rdf:resource="&owl;InverseFunctionalProperty" />  
1125   <rdfs:domain rdf:resource="#Airport"/>  
1126   <rdfs:range rdf:resource="#Flight"/>  
1127 </owl:ObjectProperty>
```

1128 Example owl:InverseFunctionalProperty

```
1129 <rim:Association id=${finalDestination}  
1130 associationType='urn:oasis:names:tc:ebxml-  
1131 regrep:profile:webontology:AssociationType:OWL:InverseFunctionalProperty'  
1132 sourceObject= ${Airport} targetObject=${Flight} >  
1133 </rim:Association>
```

1134 Example Corresponding ebRIM construct Association

1135 ebRIM MUST contain a new Association Type called “InverseFunctionalProperty” to express this
1136 semantics. Furthermore the adhoc query presented in Section 6.22 MUST be available in the Registry to
1137 retrieve inverse functional Associations of a ClassificationNode.

1138 4.4 OWL Property Restrictions in ebXML RIM

1139 An important construct of OWL is "owl:Restriction". In RDF, a property has a global scope, that is, no
1140 matter what class the property is applied to, the range of the property is the same. "owl:Restriction", on the
1141 other hand, has a local scope; restriction is applied on the property within the scope of the class where it is
1142 defined. This makes property definitions more reusable by factoring out class specific characteristics of
1143 the property into the class description.

1144 For example, we may define a property "paymentMethod" for travel Web services in general and we may
1145 state that the range of this property is the class "PossiblePaymentMethods". Then, for
1146 "AirReservationServices", we may wish to restrict "paymentMethod" property to, say, "CreditCard" class as
1147 demonstrated in the following two examples:

```
1148  
1149 <owl:ObjectProperty rdf:ID="paymentMethod">  
1150   <rdfs:domain rdf:resource="#TravelWebService"/>  
1151   <rdfs:range rdf:resource="#PossiblePaymentMethods"/>  
1152 </owl:ObjectProperty >
```

1153 Example owl:ObjectProperty “paymentMethod”

```
1154  
1155 <owl:Class rdf:ID="AirReservationServices">  
1156   <rdfs:subClassOf>  
1157     <owl:Restriction>  
1158       <owl:onProperty rdf:resource="#paymentMethod"/>  
1159       <owl:allValuesFrom rdf:resource= "#CreditCard"/>  
1160     </owl:Restriction>  
1161   </rdfs:subClassOf>  
1162 </owl:Class>
```

1163 Example owl:Restriction on ObjectProperty “paymentMethod”

1164 A new Association Type of "restriction" SHOULD be defined to represent OWL restriction. A slot of this
1165 Association Type SHOULD indicate the whether the restriction is "allValuesFrom" or "someValuesFrom".
1166 When such restriction is submitted to the system, the registry MUST create a new Association instance,
1167 say, "paymentMethod_1" of AssociationType "ObjectProperty" is created whose sourceObject is
1168 "AirReservationServices" and the targetObject is "CreditCard". "paymentMethod_1" Association instance
1169 is related with the "paymentMethod" Association instance by using an instance of the Association Type
1170 "Restriction" as shown in the following example:

```
1171  
1172 <rim:Association id = ${paymentMethod_1}  
1173             associationType =  
1174 "urn:oasis:names:tc:ebxml-  
1175 regrep:profile:webontology:AssociationType:OWL:ObjectProperty"  
1176             sourceObject = ${AirReservationServices}  
1177             targetObject = ${CreditCard}>  
1178 </rim:Association>  
1179  
1180 <rim:Association id = ${paymentMethodRestriction}  
1181             associationType =  
1182 "urn:oasis:names:tc:ebxml-  
1183 regrep:profile:webontology:AssociationType:OWL:Restriction"  
1184             sourceObject = ${paymentMethod}  
1185             targetObject = ${paymentMethod_1}>  
1186   <rim:Slot name="urn:oasis:names:tc:ebxml-  
1187 regrep:profile:webontology:slot:restrictionType">  
1188     <rim:ValueList>  
1189       <rim:Value>allValuesFrom</rim:Value>  
1190     </rim:ValueList>  
1191   </rim:Slot>  
1192 </rim:Association>
```

1193 Example Handling owl:Restriction in ebXML Registry

1194 Obviously, this serves the purpose of reusing the "paymentMethod" property. Otherwise, a new property
1195 "paymentMethodCC" can be defined between "AirReservationServices" and the "CreditCard" classes as
1196 shown in the following:

```
1197  
1198 <owl:ObjectProperty rdf:ID="paymentMethodCC">  
1199   <rdfs:domain rdf:resource="#AirReservationServices"/>  
1200   <rdfs:range rdf:resource="#CreditCard"/>  
1201 </owl:ObjectProperty >
```

1202 Example owl:ObjectProperty "paymentMethodCC"

1203 4.5 Representing OWL Restricted Cardinality in ebXML RIM

1204 4.5.1 owl:minCardinality (only 0 or 1)

1205 In OWL, cardinality is stated on a property with respect to a particular class. If a minCardinality of 1 is
1206 stated on a property with respect to a class, then any instance of the class will have at least one value for
1207 the restricted property. This restriction is another way of saying that the property is required to have a
1208 value for all instances of the class. In OWL Lite, the only minimum cardinalities allowed are 0 or 1. A
1209 minimum cardinality of zero on a property just states (in the absence of any more specific information)
1210 that the property is optional with respect to a class [McGuinness, Harmelen].

1211 Consider for example the following OWL code which states that each instance of a "WebService" class
1212 must have at least one price:

```
1213 <owl:Class rdf:ID="WebService">  
1214   <rdfs:subClassOf>  
1215     <owl:Restriction>  
1216       <owl:onProperty rdf:resource="#hasPrice"/>  
1217       <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">  
1218 1 </owl:minCardinality>  
1219     </owl:Restriction>
```

```
1220 </rdfs:subClassOf>
1221 </owl:Class>
```

1222 Example owl:minCardinality

1223 In ebXML RIM, cardinalities of Association Types MUST be defined by associating a minCardinality slot
1224 with the Association Types as shown in the following example:

1225

```
1226 <rim:Association id = ${hasPriceMinCardinalityRestriction}
1227 associationType = "urn:oasis:names:tc:ebxml-
1228 regrep:profile:webontology:AssociationType:OWL:ObjectProperty"
1229 sourceObject = ${WebService}
1230 targetObject = ${Price}>
1231 <rim:Name>
1232 <rim:LocalizedString value = 'hasPrice' />
1233 </rim:Name>
1234 <rim:Slot name="urn:oasis:names:tc:ebxml-
1235 regrep:profile:webontology:slot:minCardinality">
1236 <rim:ValueList>
1237 <rim:Value>1</rim:Value>
1238 </rim:ValueList>
1239 </rim:Slot>
1240 </rim:Association>
```

1241 Example Representing owl:minCardinality in ebRIM

1242 4.5.2 owl:maxCardinality (only 0 or 1)

1243 In OWL, cardinality is stated on a property with respect to a particular class. If a maxCardinality of 1 is
1244 stated on a property with respect to a class, then any instance of that class will be related to at most one
1245 individual by that property. A maxCardinality 1 restriction is sometimes called a functional or unique
1246 property. It may be useful to state that certain classes have no values for a particular property. This
1247 situation is represented by a maximum cardinality of zero on the property [McGuinness, Harmelen].

1248 Consider for example the following OWL code which states that each instance of a "WebService" class
1249 can have at most one price:

```
1250 <owl:Class rdf:ID="WebService">
1251 <rdfs:subClassOf>
1252 <owl:Restriction>
1253 <owl:onProperty rdf:resource="#hasPrice"/>
1254 <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
1255 1 </owl:maxCardinality>
1256 </owl:Restriction>
1257 </rdfs:subClassOf>
1258 </owl:Class>
```

1259 Example owl:maxCardinality

1260 In ebXML RIM, cardinalities of Association Types MUST be defined by associating a maxCardinality slot
1261 with the Association Types as shown in the following example:

1262

```
1263 <rim:Association id = ${hasPriceMaxCardinalityRestriction}
1264 associationType = "urn:oasis:names:tc:ebxml-
1265 regrep:profile:webontology:AssociationType:OWL:ObjectProperty"
1266 sourceObject = ${WebService}"
1267 targetObject = ${Price}>
1268 <rim:Name>
1269 <rim:LocalizedString value = 'hasPrice' />
1270 </rim:Name>
1271 <rim:Slot name="urn:oasis:names:tc:ebxml-
1272 regrep:profile:webontology:slot:maxCardinality">
1273 <rim:ValueList>
1274 <rim:Value>1</rim:Value>
1275 </rim:ValueList>
1276 </rim:Slot>
```

1277 </rim:Association>

1278 Example Representing owl:maxCardinality in ebRIM

1279 4.5.3 owl:cardinality (only 0 or 1)

1280 In OWL Lite, cardinality is provided as a convenience when it is useful to state that a property on a class
1281 has both minCardinality 0 and maxCardinality 0 or both minCardinality 1 and maxCardinality 1
1282 [McGuinness, Harmelen].

1283 Consider for example the following OWL code which states that each instance of a “WebService” class
1284 must have exactly one price:

```
1285 <owl:Class rdf:ID="WebService">  
1286   <rdfs:subClassOf>  
1287     <owl:Restriction>  
1288       <owl:onProperty rdf:resource="#hasPrice"/>  
1289       <owl:Cardinality rdf:datatype="&xsd;nonNegativeInteger"> 1  
1290 </owl:Cardinality>  
1291     </owl:Restriction>  
1292   </rdfs:subClassOf>  
1293 </owl:Class>
```

1294 Example owl:Cardinality

1295 In ebXML RIM, cardinalities of Association Types MUST be defined by associating a Cardinality slot with
1296 the Association Types as shown in the following example:

```
1297  
1298 <rim:Association id = ${hasPriceCardinalityRestriction}  
1299 associationType = "urn:oasis:names:tc:ebxml-  
1300 regrep:profile:webontology:AssociationType:OWL:ObjectProperty"  
1301 sourceObject = ${WebService}  
1302 targetObject = ${Price}>  
1303   <rim:Name>  
1304     <rim:LocalizedString value = 'hasPrice' />  
1305   </rim:Name>  
1306   <rim:Slot name="urn:oasis:names:tc:ebxml-  
1307 regrep:profile:webontology:slot:cardinality">  
1308     <rim:ValueList>  
1309       <rim:Value>1</rim:Value>  
1310     </rim:ValueList>  
1311   </rim:Slot>  
1312 </rim:Association>
```

1313 Example Representing owl:Cardinality in ebRIM

1314 4.6 Representing OWL Class Intersection in ebXML RIM

1315 OWL provides the means to manipulate class extensions using basic set operators. In OWL lite, only
1316 “owl:intersectionOf” is available which defines a class that consists of exactly all objects that belong to all
1317 the classes specified in the intersection definition. In the following example, "AirReservationServices" is
1318 defined as the intersection of "AirServices" and "ReservationServices":

```
1319  
1320 <owl:Class rdf:ID="AirReservationServices">  
1321   <owl:intersectionOf rdf:parseType="Collection">  
1322     <owl:Class rdf:about="#AirServices" />  
1323     <owl:Class rdf:about="#ReservationServices" />  
1324   </owl:intersectionOf>  
1325 </owl:Class>
```

1326 Example owl:intersectionOf

1327 In ebXML RIM "owl:intersectionOf" set operator MUST be represented as follows:

- 1328
- A new ClassificationNode is created for representing the complex class.
 - The id's of the classes that are involved in the intersection definition are put as the “values” of the
- 1329

1330 multi-valued slot named as: "urn:oasis:names:tc:ebxml-
1331 regrep:profile:webontology:slot:intersectionOf".

1332

```
1333 <rim:ClassificationNode id = ${AirReservationServices} code =  
1334 "AirReservationServices">  
1335   <rim:Slot name=urn:oasis:names:tc:ebxml-  
1336     regrep:profile:webontology:slot:intersectionOf>  
1337     <rim:ValueList>  
1338       <rim:Value>${AirServices}</rim:Value>  
1339       <rim:Value>${ReservationServices}</rim:Value>  
1340     </rim:ValueList>  
1341   </rim:Slot>  
1342 </rim:ClassificationNode>  
1343
```

1344 **Example Defining Intersection of ClassificationNodes in ebRIM**

1345 When such a representation is used to create a complex class (a new ClassificationNode) in RIM, it
1346 becomes possible to infer that the objects (instances) classified by all of the classes
1347 (ClassificationNodes) specified in the Intersection definition, are also classified by this complex class. The
1348 adhoc query presented in Section 6.23 MUST be available in the ebXML Registry to retrieve the direct
1349 instances of the complex class and also the instances of the intersection of the classes.

1350

1351 **4.7 Representing OWL Versioning in ebXML RIM**

1352 **4.7.1 owl:versionInfo, owl:priorVersion**

1353 An owl:versionInfo statement generally has as its object a string giving information about this version, for
1354 example RCS/ CVS keywords. This statement does not contribute to the logical meaning of the ontology
1355 other than that given by the RDF(S) model theory [McGuinness, Harmelen].

1356 An owl:priorVersion statement contains a reference to another ontology. This identifies the specified
1357 ontology as a prior version of the containing ontology [McGuinness, Harmelen].

1358 In ebXML, since a RegistryObject MAY have several versions, a logical id (called lid) is also defined which
1359 is unique for different logical objects. However the lid attribute value MUST be the same for all versions of
1360 the same logical object. Therefore, almost for all the RegistryObjects the version information is kept
1361 through "versionName" and "comment" attributes of the "VersionInfo" ebRIM Class.

1362 "owl:version" information MUST be stored in the "versionName" and "comment" attributes of the
1363 VersionInfo ebRIM class.

1364 It should be noted that in freebXML implementation the versionInfo is flattened and the "versionName"
1365 and "comment_" are provided as direct attributes of database tables.

```
1366 <owl:Ontology rdf:about="">  
1367   <owl:versionInfo>v 1.17 2003/02/26 12:56:51 </owl:versionInfo>  
1368 </owl:Ontology>
```

1369 **Example owl:versionInfo**

```
1370 <rim:ClassificationScheme  
1371 lid= ${exampleOntology}  
1372 id=${exampleOntology} isInternal="true"  
1373 nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode">  
1374   <rim:versionInfo>  
1375     <rim:versionName>  
1376       <rim:LocalizedString charset="UTF-8" value="v 1.17 2003/02/26  
1377       12:56:51"/>  
1378     </rim:versionName>  
1379   </rim:versionInfo>  
1380 </rim:ClassificationScheme>
```

1381 **Example rim:versionName**

1382 4.8 Representing OWL Annotation Properties in ebXML RIM

1383 4.8.1 rdfs:label

1384 rdfs:label is an instance of rdf:Property that may be used to provide a human-readable version of a
1385 resource's name [Brickley, Guha].

1386 In ebXML RIM, human readable names of resources are provided through rim:Name. rdfs:label MUST be
1387 expressed through rim:Name.

1388

```
1389 <owl:Class rdf:ID="AirReservationServices">  
1390   <rdfs:label>Air Reservation Services</rdfs:label>  
1391 </owl:Class>
```

1392 Example rdfs:label

1393

```
1394 <rim:ClassificationNode id = ${AirReservationServices} code =  
1395 'AirReservationServices'  
1396   <rim:Name>  
1397     <rim:LocalizedString value = 'Air Reservation Services' />  
1398   </rim:Name>  
1399 </rim:ClassificationNode>
```

1400 Example rim:Name

1401 4.8.2 rdfs:comment

1402 rdfs:comment is an instance of rdf:Property that may be used to provide a human-readable description of
1403 a resource [Brickley, Guha].

1404 In ebXML RIM, this construct MUST be expressed through rim:Description.

1405

```
1406 <owl:Class rdf:ID="AirReservationServices">  
1407   <rdfs:comment>Open Travel Alliance Air Reservation Services  
1408   </rdfs:comment>  
1409 </owl:Class>
```

1410 Example rdfs:comment

1411

```
1412 <rim:ClassificationNode id = ${AirReservationServices} code =  
1413 'AirReservationServices'  
1414   <rim:Description>  
1415     <rim:LocalizedString value = 'Open Travel Alliance Air  
1416     Reservation Services' />  
1417   </rim:Description>  
1418 </rim:ClassificationNode>
```

1419 Example: rim:Description

1420 4.8.3 rdfs:seeAlso

1421 rdfs:seeAlso is an instance of rdf:Property that is used to indicate a resource that might provide additional
1422 information about the subject resource [Brickley, Guha].

1423 This construct MUST be expressed in ebXML RIM by defining a new Association Type called "SeeAlso" to
1424 express this semantics.

1425

```
1426 <owl:Class rdf:ID="AirReservationServices">  
1427   <rdfs:seeAlso rdf:resource="http://www.opentravel.org" />  
1428 </owl:Class>
```

1429 Example rdfs:seeAlso

```

1430 <rim:ClassificationNode id = ${AirReservationServices} code =
1431 'AirReservationServices'>
1432 </rim:ClassificationNode>
1433
1434 <rim:ExternalLink id = ${exampleExternalLink}
1435     externalURI= "http://www.opentravel.org" >
1436 </rim:ExternalLink>
1437
1438 <rim:Association id = ${seeAlsoID}
1439     associationType = 'urn:oasis:names:tc:ebxml-
1440 regrep:profile:webontology:AssociationType:OWL:SeeAlso'
1441     sourceObject = ${AirReservationServices}
1442     targetObject = ${exampleExternalLink} />

```

1443 **Example rim:seeAlsoExternalLink**

1444 4.9 OWL Datatypes in ebXML RIM

1445 OWL allows the use of XML Schema datatypes to describe part of the datatype domain by simply
1446 including their URIs within an OWL ontology [McGuinness, Harmelen]. In ebXML, XML Schema datatypes
1447 MAY be used by providing an external link from the registry.

1448 The following example demonstrates how XML Schema datatype “integer” can be referenced through an
1449 ExternalLink whose id is 'urn:www.w3.org:2001/XMLSchema:integer' and how to define a
1450 DatatypeProperty, namely, “hasPrice”, whose target object is the defined to be ExternalLink 'integer':

```

1451 <rim:ExternalLink id = urn:www.w3.org:2001/XMLSchema:integer
1452     externalURI="http://www.w3.org/2001/XMLSchema#integer" >
1453     <rim:Name> <rim:LocalizedString value = "XML Schema integer"/>
1454     </rim:Name>
1455 </rim:ExternalLink>
1456 <rim:Association id = ${hasPrice} associationType =
1457 'urn:oasis:names:tc:ebxml-
1458 regrep:AssociationType:DatatypeProperty'
1459     sourceObject = ${AirReservationServices}
1460     targetObject = urn:www.w3.org:2001/XMLSchema:integer >
1461     <rim:Name> <rim:LocalizedString value ="hasPrice"/></rim:Name>
1462 </rim:Association>
1463

```

1464 **Example Corresponding ebRIM construct Association**

1465 5 Cataloging Service Profile

1466 The ebXML Registry provides the ability for a content cataloging service to be configured for any type of
1467 content. The cataloging service serves the following purposes:

- 1468 • Automates the mapping from the source information model (in this case OWL) to ebRIM. This
1469 hides the complexity of the mapping from the OWL publisher and eliminates the need for any
1470 special UI tools to be provided by the registry implementor for publishing OWL documents.
- 1471 • Selectively converts content into ebRIM compatible metadata when the content is cataloged after
1472 being published. The generated metadata enables the selected content to be used as
1473 parameter(s) in content specific parameterized queries.

1474 This section describes the cataloging service for cataloging OWL content.

1475 An OWL document, when published to an ebXML Registry implementing the OWL Profile, **MUST** be
1476 cataloged as specified in this section using a OWL Content Cataloging Service as defined by [ebRS].

1477 5.1 Invocation Control File

1478 The OWL cataloging service **MAY** optionally support an invocation control file that declaratively specifies
1479 the transforms necessary to catalog published OWL documents.

1480 5.2 Input Metadata

1481 The OWL cataloging service **MUST** be pre-configured to be automatically invoked when the following
1482 types of metadata are published, as defined by the [ebRS] specifications.

1483 These are the only types of metadata that **MAY** describe a OWL document being published:

- 1484 • An `ExtrinsicObject` whose `ObjectType` references the canonical OWL `ClassificationNode`
1485 specified in Section 7. The `ExtrinsicObject` **MUST** have an OWL document as its `RepositoryItem`.
- 1486 • An `ExternalLink` whose `ObjectType` references the canonical OWL `ClassificationNode` specified in
1487 Section 7. In case of `ExternalLink` the OWL document **MUST** be resolvable via a URL described
1488 by the value of the `externalURI` attribute of the `ExternalLink`. Recall that, in the `ExternalLink` case
1489 the OWL document is not stored in the repository.

1490

```
1491 <rim:ExtrinsicObject id="urn:acmeinc:ebxml:registry:3.0:owl">  
1492 ...  
1493 </rim:ExtrinsicObject>
```

1494 Example of ExtrinsicObject Input Metadata

1495

```
1496 <rim:ExternalLink  
1497   id="urn:acmeinc:ebxml:registry:3.0:owl"  
1498   externalURI="http://www.acme.com/owl/ebXMLRegistryService.owl"  
1499   >  
1500 ...  
1501 </rim:ExternalLink>
```

1502 Example of ExternalLink Input Metadata

1503 5.3 Input Content

1504 The OWL cataloging service expects an OWL document as its input content. The input content **MUST** be
1505 processed by the OWL cataloging service regardless of whether it is a `RepositoryItem` for an
1506 `ExtrinsicObject` or whether it is content external to repository that is referenced by an `ExternalLink`.

1507

1508 5.4 Output Metadata

1509 This section describes the metadata produced by the OWL cataloging service produces as output.

1510 5.4.1 owl:Class → rim:ClassificationNode

1511 The OWL Cataloging service MUST automatically produce a rim:ClassificationNode instance for each
1512 owl:class element within the input OWL or its imports, as specified in the owl:Class →
1513 rim:ClassificationNode mapping earlier in this document.

1514 5.4.2 rdf:Property → rim:Association Type HasProperty

1515 The OWL Cataloging service MUST automatically produce an rim:Association instance with
1516 associationType HasProperty for each rdf:Property element within the input OWL or its imports, as
1517 specified in the rdf:Property → rim:Association Type HasProperty mapping earlier in this document.

1518 5.4.3 rdfs:subPropertyOf → rim:Association Type SubPropertyOf

1519 The OWL Cataloging service MUST automatically produce an rim:Association instance with
1520 associationType SubPropertyOf for each rdfs:subPropertyOf element within the input OWL or its imports,
1521 as specified in the rdfs:subPropertyOf → rim:Association Type SubPropertyOf mapping earlier in this
1522 document.

1523 5.4.4 rdfs:subClassOf → rim:Association Type subClassOf

1524 The OWL Cataloging service MUST automatically produce an rim:Association instance with
1525 associationType subClassOf for each rdfs:subClassOf element within the input OWL or its imports, as
1526 specified in the rdfs:subClassOf → rim:Association Type subClassOf mapping earlier in this document.

1527 5.4.5 owl:Individual → rim:ExtrinsicObject

1528 The OWL Cataloging service MUST automatically produce rim:ExtrinsicObject instances for each
1529 owl:Individual element within the input OWL or its imports, as specified in the owl:Individual →
1530 rim:ExtrinsicObject mapping earlier in this document.

1531 5.4.6 owl:equivalentClass, owl:equivalentProperty → rim:Association Type 1532 EquivalentTo

1533 The OWL Cataloging service MUST automatically produce rim:Association instances with
1534 associationType EquivalentTo for each owl:equivalentClass or owl:equivalentProperty element within the
1535 input OWL or its imports, as specified in the owl:equivalentClass, owl:equivalentProperty →
1536 rim:Association Type EquivalentTo mapping earlier in this document.

1537 5.4.7 owl:sameAs → rim:Association Type SameAs

1538 The OWL Cataloging service MUST automatically produce rim:Association instances with
1539 associationType SameAs for each owl:sameAs element within the input OWL or its imports, as specified
1540 in the owl:sameAs → rim:Association Type SameAs mapping earlier in this document.

1541 5.4.8 owl:differentFrom → rim:Association Type DifferentFrom

1542 The OWL Cataloging service MUST automatically produce rim:Association instances with
1543 associationType DifferentFrom for each owl:differentFrom element within the input OWL or its imports, as
1544 specified in the owl:differentFrom → rim:Association Type DifferentFrom mapping earlier in this
1545 document.

1546 5.4.9 owl:AllDifferent → rim:RegistryPackage

1547 The OWL Cataloging service MUST automatically produce rim:RegistryPackage instances for each

1548 owl:AllDifferent element within the input OWL or its imports, as specified in the owl:AllDifferent →
1549 rim:RegistryPackage mapping earlier in this document.

1550 [5.4.10 owl:ObjectProperty → rim:Association Type ObjectProperty](#)

1551 The OWL Cataloging service MUST automatically produce rim:Association instances with
1552 associationType ObjectProperty for each owl:ObjectProperty element within the input OWL or its imports,
1553 as specified in the owl:ObjectProperty → rim:Association Type ObjectProperty mapping earlier in this
1554 document.

1555 [5.4.11 owl:DatatypeProperty → rim:Association Type DatatypeProperty](#)

1556 The OWL Cataloging service MUST automatically produce rim:Association instances with
1557 associationType datatypeProperty for each owl:DatatypeProperty element within the input OWL or its
1558 imports, as specified in the owl:DatatypeProperty → rim:Association Type datatypeProperty mapping
1559 earlier in this document.

1560 [5.4.12 owl:TransitiveProperty → rim:Association Type TransitiveProperty](#)

1561 The OWL Cataloging service MUST automatically produce rim:Association instances with
1562 associationType TransitiveProperty for each owl:TransitiveProperty element within the input OWL or its
1563 imports, as specified in the owl:TransitiveProperty → rim:Association Type TransitiveProperty mapping
1564 earlier in this document.

1565 [5.4.13 owl:inverseOf → rim:Association Type InverseOf](#)

1566 The OWL Cataloging service MUST automatically produce rim:Association instances with
1567 associationType InverseOf for each owl:inverseOf element within the input OWL or its imports, as
1568 specified in the owl:inverseOf → rim:Association Type InverseOf mapping earlier in this document.

1569 [5.4.14 owl:SymmetricProperty → rim:Association Type SymetricProperty](#)

1570 The OWL Cataloging service MUST automatically produce rim:Association instances with
1571 associationType SymetricProperty for each owl:SymetricProperty element within the input OWL or its
1572 imports, as specified in the owl:SymetricProperty → rim:Association Type SymetricProperty mapping
1573 earlier in this document.

1574 [5.4.15 owl:FunctionalProperty → rim:Association Type FunctionalProperty](#)

1575 The OWL Cataloging service MUST automatically produce rim:Association instances with
1576 associationType FunctionalProperty for each owl:FunctionalProperty element within the input OWL or its
1577 imports, as specified in the owl:FunctionalProperty → rim:Association Type FunctionalProperty mapping
1578 earlier in this document.

1579 [5.4.16 owl:InverseFunctionalProperty → rim:Association Type 1580 InverseFunctionalProperty](#)

1581 The OWL Cataloging service MUST automatically produce rim:Association instances with
1582 associationType InverseFunctionalProperty for each owl:InverseFunctionalProperty element within the
1583 input OWL or its imports, as specified in the owl:InverseFunctionalProperty → rim:Association Type
1584 InverseFunctionalProperty mapping earlier in this document.

1585 [5.4.17 owl:minCardinality \(only 0 or 1\)](#)

1586 The OWL Cataloging service MUST automatically add a slot with name minCardinality to the relevant
1587 rim:Association instances for each owl:minCardinality element within the input OWL or its imports, as
1588 specified in section 4.5.1 where how to represent owl:minCardinality is described.

1589 **5.4.18 owl:maxCardinality (only 0 or 1)**

1590 The OWL Cataloging service MUST automatically add a slot with name maxCardinality to the relevant
1591 rim:Association instances for each owl:maxCardinality element within the input OWL or its imports, as
1592 specified in section 4.5.2 where how to represent owl:maxCardinality is described.

1593 **5.4.19 owl:cardinality**

1594 The OWL Cataloging service MUST automatically add a slot with name cardinality to the relevant
1595 rim:Association instances for each owl:cardinality element within the input OWL or its imports, as
1596 specified in section 4.5.3 where how to represent owl:cardinality is described.

1597 **5.4.20 owl:intersectionOf**

1598 The OWL Cataloging service MUST automatically produce a rim:RegistryPackage and a rim:Association
1599 instances with type IntersectionOf for each owl:intersectionOf element within the input OWL or its imports,
1600 as specified in section 4.6 where how to represent owl:intersectionOf is described.

1601 **5.4.21 rdfs:label**

1602 The OWL Cataloging service MUST automatically produce a rim:Name instance for each rdfs:label
1603 element within the input OWL or its imports, as specified in section 4.8.1 where how to represent
1604 rdfs:label is described.

1605 **5.4.22 rdfs:comment**

1606 The OWL Cataloging service MUST automatically produce a rim:Description instance for each
1607 rdfs:comment element within the input OWL or its imports, as specified in section 4.8.2 where how to
1608 represent rdfs:comment is described.

1609 **5.4.23 rdfs:seeAlso**

1610 The OWL Cataloging service MUST automatically produce a rim:ExternalLink and a rim:Association with
1611 type SeeAlso instances for each rdfs:seeAlso element within the input OWL or its imports, as specified in
1612 section 4.8.3 where how to represent rdfs:seeAlso is described.

6 Discovery Profile

1613

1614 The ebXML Registry provides the ability for a user defined parameterized queries to be configured for
1615 each type of content. The queries may be as complex or simple as the discovery use case requires. The
1616 complexity of the parameterized queries may hidden from the registry client by storing them within the
1617 ebXML Registry as instances of the AdhocQuery class, and being invoked by simply providing their
1618 parameters. Query parameters are often pattern strings that may contain wildcard characters '%' (matches any number of characters) and '_' (matches exactly one character) as described by [ebRS].

1620 An ebXML Registry SHOULD provide a graphical user interface that displays any configured
1621 parameterized query as a form which contains an appropriate field for entering each query parameter.

1622 This chapter defines the queries that MUST be supported by an ebXML Registry implementing the OWL
1623 Profile for processing the semantics provided in the OWL content. An implementation MAY also support
1624 additional discovery queries for OWL content, some of which have already identified in this section.

1625 The queries defined in this chapter are parameterized queries stored in the Registry as instances of the
1626 AdhocQuery type, in the same manner as any other RegistryObject.

1627 In the subsequent section each query is described simply in terms of its supported parameters that serve
1628 as its search criteria. The actual AdhocQuery instances are much more complex in comparison but they
1629 are not exposed to the client making the query. Details on these queries are specified canonically in
1630 section 7.3 .

1631 Some of the queries that are necessary to process the semantics involved in OWL documents requires
1632 SQL recursion mechanism which is available through SQL 99 Standard. Since SQL 92, does not support
1633 recursion mechanism, those queries are stated to be implemented optionally. Additionally for these types
1634 of discovery queries, references to the "stored procedures" are presented in Section 7.3 for the interested
1635 users.

6.1 All SuperProperties Discovery Query

1637 As presented in Section 4.1.3, a new ebXML RIM Association Type called "SubPropertyOf" MUST be
1638 defined to represent rdfs:subPropertyOf in ebRIM. Such a semantic enhancement brings the following
1639 processing need: given a property, it should be possible to retrieve all of its super properties. This requires
1640 a recursion mechanism in SQL queries.

1641 The AllSuperProperties discovery query MAY be implemented by an ebXML Registry implementing this
1642 profile. It allows the discovery of all super properties of a given property instance (Association instance in
1643 ebXML terminology) recursively in a property hierarchy (hierarchy of Association Types) in an ebXML
1644 Registry implementation supporting recursion. The canonical query corresponding to this discovery query
1645 is presented in Section 7.3.1.

6.1.1 Parameter \$propertyName

1647 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1648 value of Associations that have associationType of Property.

6.1.2 Example of All SuperProperties Discovery Query

1650 The following example illustrates how to find all the super properties of a given property having a name
1651 containing "creditCardPayment" if the query is implemented as an AdHoc Query.

1652

```
1653 <rs:RequestSlotList>  
1654   <rim:Slot  
1655     name="urn:oasis:names:tc:ebxml-  
1656   regrep:3.0:rs:AdhocQueryRequest:queryId">  
1657     <rim:ValueList>  
1658       <rim:Value>urn:oasis:names:tc:ebxml-  
1659   regrep:profile:webontology:query:FindAllSuperProperties</rim:Value>  
1660     </rim:ValueList>  
1661   </rim:Slot>
```

```

1662     <rim:Slot name="urn:oasis:names:tc:ebxml-
1663 regrep:rs:AdhocQueryRequest:queryId">
1664         <rim:ValueList>
1665             <rim:Value>urn:oasis:names:tc:ebxml-
1666 regrep:profile:webontology:query:FindAllSuperProperties</rim:Value>
1667         </rim:ValueList>
1668     </rim:Slot>
1669     <rim:Slot name="$propertyName">
1670         <rim:ValueList>
1671             <rim:Value>%creditCardPayment%</rim:Value>
1672         </rim:ValueList>
1673     </rim:Slot>
1674 </rs:RequestSlotList>
1675
1676 <query:ResponseOption returnComposedObjects="true"
1677     returnType="LeafClassWithRepositoryItem"/>
1678
1679 <rim:AdhocQuery id="temporaryId">
1680     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1681 regrep:QueryLanguage:SQL-92">
1682     </rim:QueryExpression>
1683 </rim:AdhocQuery>

```

1684 Example of All SuperProperties Discovery Query

1685 6.2 Immediate SuperClass Discovery Query

1686 The Immediate SuperClass discovery query MUST be implemented by an ebXML Registry implementing
1687 this profile. It allows the discovery of all of the immediate super classes of a given class. The canonical
1688 query corresponding to this discovery query is presented in Section 7.3.2.

1689 6.2.1 Parameter \$className

1690 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1691 value of ClassificationNodes.

1692 6.2.2 Example of Immediate SuperClass Discovery Query

1693 The following example illustrates how to find all the immediate super classes of a given class that have a
1694 name containing the string "AirReservationServices".

```

1695 <rs:RequestSlotList>
1696     <rim:Slot
1697         name="urn:oasis:names:tc:ebxml-
1698 regrep:3.0:rs:AdhocQueryRequest:queryId">
1699         <rim:ValueList>
1700             <rim:Value>urn:oasis:names:tc:ebxml-
1701 regrep:profile:webontology:query:FindImmediateSuperClasses</rim:Value>
1702         </rim:ValueList>
1703     </rim:Slot>
1704     <rim:Slot name="urn:oasis:names:tc:ebxml-
1705 regrep:rs:AdhocQueryRequest:queryId">
1706         <rim:ValueList>
1707             <rim:Value>urn:oasis:names:tc:ebxml-
1708 regrep:profile:webontology:query:FindImmediateSuperClasses</rim:Value>
1709         </rim:ValueList>
1710     </rim:Slot>
1711     <rim:Slot name="$className">
1712         <rim:ValueList>
1713             <rim:Value>%AirReservationServices%</rim:Value>
1714         </rim:ValueList>
1715     </rim:Slot>
1716 </rs:RequestSlotList>
1717
1718 <query:ResponseOption returnComposedObjects="true"
1719     returnType="LeafClassWithRepositoryItem"/>

```

```

1720 <rim:AdhocQuery id="temporaryId">
1721   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1722   regrep:QueryLanguage:SQL-92">
1723     </rim:QueryExpression>
1724   </rim:AdhocQuery>
1725

```

1726 Example of Immediate SuperClass Discovery Query

1727 6.3 Immediate SubClass Discovery Query

1728 The Immediate SubClass discovery query MUST be implemented by an ebXML Registry implementing
1729 this profile. It allows the discovery of all of the immediate subclasses of a given class. The canonical
1730 query corresponding to this discovery query is presented in Section 7.3.3.

1731 6.3.1 Parameter \$className

1732 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1733 value of ClassificationNode.

1734 6.3.2 Example of Immediate SubClass Discovery Query

1735 The following example illustrates how to find all the immediate subclasses of a given class that have a
1736 name containing the string "AirServices" .

```

1737 <rs:RequestSlotList>
1738   <rim:Slot
1739     name="urn:oasis:names:tc:ebxml-
1740   regrep:3.0:rs:AdhocQueryRequest:queryId">
1741     <rim:ValueList>
1742       <rim:Value>urn:oasis:names:tc:ebxml-
1743   regrep:profile:webontology:query:FindImmediateSubClasses</rim:Value>
1744     </rim:ValueList>
1745   </rim:Slot>
1746   <rim:Slot name="urn:oasis:names:tc:ebxml-
1747   regrep:rs:AdhocQueryRequest:queryId">
1748     <rim:ValueList>
1749       <rim:Value>urn:oasis:names:tc:ebxml-
1750   regrep:profile:webontology:query:FindImmediateSubClasses</rim:Value>
1751     </rim:ValueList>
1752   </rim:Slot>
1753   <rim:Slot name="$className">
1754     <rim:ValueList>
1755       <rim:Value>%AirServices%</rim:Value>
1756     </rim:ValueList>
1757   </rim:Slot>
1758 </rs:RequestSlotList>
1759
1760 <query:ResponseOption returnComposedObjects="true"
1761   returnType="LeafClassWithRepositoryItem"/>
1762
1763 <rim:AdhocQuery id="temporaryId">
1764   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1765   regrep:QueryLanguage:SQL-92">
1766     </rim:QueryExpression>
1767   </rim:AdhocQuery>

```

1768 Example of Immediate SubClass Discovery Query

1769 6.4 All SuperClasses Discovery Query

1770 It should be noted that, given a class, finding its immediate subclasses, super classes is necessary but
1771 not sufficient. Given a class, it should be possible to retrieve all of its subclasses, and all of its super
1772 classes. This requires a recursion mechanism in SQL queries.

1773 The All SuperClasses discovery query MAY be implemented by an ebXML Registry implementing this
1774 profile. It allows the discovery of all super classes of a given ClassificationNode recursively in an ebXML
1775 Registry implementation supporting recursion. The canonical query corresponding to this discovery query
1776 is presented in Section 7.3.4.

1777 6.4.1 Parameter \$className

1778 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1779 value of ClassificationNode.

1780 6.4.2 Example of All SuperClasses Discovery Query

1781 The following example illustrates how to find all the super classes of a given class recursively that have a
1782 name containing the string "AirReservationServices" if the query is implemented as an Adhoc Query .

```
1783 <rs:RequestSlotList>  
1784   <rim:Slot  
1785     name="urn:oasis:names:tc:ebxml-  
1786   regrep:3.0:rs:AdhocQueryRequest:queryId">  
1787     <rim:ValueList>  
1788       <rim:Value>urn:oasis:names:tc:ebxml-  
1789   regrep:profile:webontology:query:FindAllSuperClasses</rim:Value>  
1790     </rim:ValueList>  
1791   </rim:Slot>  
1792   <rim:Slot name="urn:oasis:names:tc:ebxml-  
1793   regrep:rs:AdhocQueryRequest:queryId">  
1794     <rim:ValueList>  
1795       <rim:Value>urn:oasis:names:tc:ebxml-  
1796   regrep:profile:webontology:query:FindAllSuperClasses</rim:Value>  
1797     </rim:ValueList>  
1798   </rim:Slot>  
1799   <rim:Slot name="$className">  
1800     <rim:ValueList>  
1801       <rim:Value>%AirReservationServices%</rim:Value>  
1802     </rim:ValueList>  
1803   </rim:Slot>  
1804 </rs:RequestSlotList>  
1805  
1806 <query:ResponseOption returnComposedObjects="true"  
1807   returnType="LeafClassWithRepositoryItem"/>  
1808  
1809 <rim:AdhocQuery id="temporaryId">  
1810   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
1811   regrep:QueryLanguage:SQL-92">  
1812     </rim:QueryExpression>  
1813 </rim:AdhocQuery>
```

1814 Example of All SuperClasses Discovery Query

1815 6.5 All SubClasses Discovery Query

1816 The All SubClasses discovery query MAY be implemented by an ebXML Registry implementing this
1817 profile. It allows the discovery of all subclasses of a given ClassificationNode recursively in an ebXML
1818 Registry implementation supporting recursion. The canonical query corresponding to this discovery query
1819 is presented in Section 7.3.5.

1820 6.5.1 Parameter \$className

1821 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1822 value of ClassificationNode.

1823 6.5.2 Example of All SubClasses Discovery Query

1824 The following example illustrates how to find all the subclasses of a given class recursively that have a

1825 name containing the string "AirServices" , if the query is implemented as an Adhoc Query.

```
1826 <rs:RequestSlotList>
1827   <rim:Slot
1828     name="urn:oasis:names:tc:ebxml-
1829   regrep:3.0:rs:AdhocQueryRequest:queryId">
1830     <rim:ValueList>
1831       <rim:Value>urn:oasis:names:tc:ebxml-
1832   regrep:profile:webontology:query:FindAllSubClasses</rim:Value>
1833     </rim:ValueList>
1834   </rim:Slot>
1835   <rim:Slot name="urn:oasis:names:tc:ebxml-
1836   regrep:rs:AdhocQueryRequest:queryId">
1837     <rim:ValueList>
1838       <rim:Value>urn:oasis:names:tc:ebxml-
1839   regrep:profile:webontology:query:FindAllSubClasses</rim:Value>
1840     </rim:ValueList>
1841   </rim:Slot>
1842   <rim:Slot name="$className">
1843     <rim:ValueList>
1844       <rim:Value>%AirServices%</rim:Value>
1845     </rim:ValueList>
1846   </rim:Slot>
1847 </rs:RequestSlotList>
1848
1849 <query:ResponseOption returnComposedObjects="true"
1850   returnType="LeafClassWithRepositoryItem"/>
1851
1852 <rim:AdhocQuery id="temporaryId">
1853   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1854   regrep:QueryLanguage:SQL-92">
1855     </rim:QueryExpression>
1856 </rim:AdhocQuery>
```

1857 Example of All SubClasses Discovery Query

1858 6.6 EquivalentClasses Discovery Query

1859 The EquivalentClasses discovery query MUST be implemented by an ebXML Registry implementing this
1860 profile. It allows the discovery of all the equivalent classes of a given ClassificationNode.

1861 6.6.1 Parameter \$className

1862 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1863 value of ClassificationNodes.

1864 6.6.2 Example of EquivalentClasses Discovery Query

1865 The following example illustrates how to find all the equivalent classes of a given class that have a name
1866 containing the string "AirServices" .

```
1867 <rs:RequestSlotList>
1868   <rim:Slot
1869     name="urn:oasis:names:tc:ebxml-
1870   regrep:3.0:rs:AdhocQueryRequest:queryId">
1871     <rim:ValueList>
1872       <rim:Value>urn:oasis:names:tc:ebxml-
1873   regrep:profile:webontology:query:FindEquivalentClasses</rim:Value>
1874     </rim:ValueList>
1875   </rim:Slot>
1876   <rim:Slot name="urn:oasis:names:tc:ebxml-
1877   regrep:rs:AdhocQueryRequest:queryId">
1878     <rim:ValueList>
1879       <rim:Value>urn:oasis:names:tc:ebxml-
1880   regrep:profile:webontology:query:FindEquivalentClasses</rim:Value>
1881     </rim:ValueList>
```

```

1882     </rim:Slot>
1883     <rim:Slot name="$className">
1884         <rim:ValueList>
1885             <rim:Value>%AirServices%</rim:Value>
1886         </rim:ValueList>
1887     </rim:Slot>
1888 </rs:RequestSlotList>
1889
1890 <query:ResponseOption returnComposedObjects="true"
1891     returnType="LeafClassWithRepositoryItem"/>
1892
1893 <rim:AdhocQuery id="temporaryId">
1894     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1895     regrep:QueryLanguage:SQL-92">
1896     </rim:QueryExpression>
1897 </rim:AdhocQuery>

```

1898 Example of Equivalent Classes Discovery Query

1899 6.7 EquivalentProperties Discovery Query

1900 The EquivalentProperties discovery query MUST be implemented by an ebXML Registry implementing
1901 this profile. It allows the discovery of all the equivalent properties of a given Association that have
1902 associationType of Property. The canonical query corresponding to this discovery query is presented in
1903 Section 7.3.7.

1904 6.7.1 Parameter \$propertyName

1905 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1906 value of Associations that have associationType of Property

1907 6.7.2 Example of EquivalentProperties Discovery Query

1908 The following example illustrates how to find all the equivalent properties(Association Type) of a given
1909 property (Association Type) that have a name containing the string "paymentMethods" .

```

1910 <rs:RequestSlotList>
1911     <rim:Slot
1912         name="urn:oasis:names:tc:ebxml-
1913     regrep:3.0:rs:AdhocQueryRequest:queryId">
1914         <rim:ValueList>
1915             <rim:Value>urn:oasis:names:tc:ebxml-
1916     regrep:profile:webontology:query:FindEquivalentProperties</rim:Value>
1917         </rim:ValueList>
1918     </rim:Slot>
1919     <rim:Slot name="urn:oasis:names:tc:ebxml-
1920     regrep:rs:AdhocQueryRequest:queryId">
1921         <rim:ValueList>
1922             <rim:Value>urn:oasis:names:tc:ebxml-
1923     regrep:profile:webontology:query:FindEquivalentProperties</rim:Value>
1924         </rim:ValueList>
1925     </rim:Slot>
1926     <rim:Slot name="$propertyName">
1927         <rim:ValueList>
1928             <rim:Value>%paymentMethods%</rim:Value>
1929         </rim:ValueList>
1930     </rim:Slot>
1931 </rs:RequestSlotList>
1932
1933 <query:ResponseOption returnComposedObjects="true"
1934     returnType="LeafClassWithRepositoryItem"/>
1935
1936 <rim:AdhocQuery id="temporaryId">
1937     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1938     regrep:QueryLanguage:SQL-92">

```

```
1939     </rim:QueryExpression>
1940 </rim:AdhocQuery>
```

1941 Example of Equivalent Properties Discovery Query

1942 6.8 SameExtrinsicObjects Discovery Query

1943 The SameExtrinsicObjects discovery query MUST be implemented by an ebXML Registry implementing
1944 this profile. It allows the discovery of all the "ExtrinsicObjects" defined to be the same with a given
1945 ExtrinsicObject. The canonical query corresponding to this discovery query is presented in Section 7.3.8.

1946 6.8.1 Parameter \$extrinsicObjectName

1947 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1948 value of ExtrinsicObjects.

1949 6.8.2 Example of SameExtrinsicObjects Discovery Query

1950 The following example illustrates how to find all the ExtrinsicObjects that are defined to be the same as
1951 the ExtrinsicObject that have a name containing the string "MyDocument" .

```
1952 <rs:RequestSlotList>
1953   <rim:Slot
1954     name="urn:oasis:names:tc:ebxml-
1955     regrep:3.0:rs:AdhocQueryRequest:queryId">
1956     <rim:ValueList>
1957       <rim:Value>urn:oasis:names:tc:ebxml-
1958       regrep:profile:webontology:query:FindTheSameExtrinsicObjects</rim:Value>
1959     </rim:ValueList>
1960   </rim:Slot>
1961   <rim:Slot name="urn:oasis:names:tc:ebxml-
1962   regrep:rs:AdhocQueryRequest:queryId">
1963     <rim:ValueList>
1964       <rim:Value>urn:oasis:names:tc:ebxml-
1965       regrep:profile:webontology:query:FindTheSameExtrinsicObjects</rim:Value>
1966     </rim:ValueList>
1967   </rim:Slot>
1968   <rim:Slot name="$extrinsicObjectName">
1969     <rim:ValueList>
1970       <rim:Value>%MyDocument%</rim:Value>
1971     </rim:ValueList>
1972   </rim:Slot>
1973 </rs:RequestSlotList>
1974
1975 <query:ResponseOption returnComposedObjects="true"
1976   returnType="LeafClassWithRepositoryItem"/>
1977
1978 <rim:AdhocQuery id="temporaryId">
1979   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1980   regrep:QueryLanguage:SQL-92">
1981     </rim:QueryExpression>
1982 </rim:AdhocQuery>
```

1984 Example of SameExtrinsicObjects Discovery Query

1985 6.9 DifferentExtrinsicObjects Discovery Query

1986 The DifferentExtrinsicObjects discovery query MUST be implemented by an ebXML Registry
1987 implementing this profile. It allows the discovery of all the "ExtrinsicObjects" defined to be the different
1988 from a given ExtrinsicObject. The canonical query corresponding to this discovery query is presented in
1989 Section 7.3.9.

1990 6.9.1 Parameter \$extrinsicObjectName

1991 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1992 value of ExtrinsicObjects.

1993 6.9.2 Example of DifferentExtrinsicObjects Discovery Query

1994 The following example illustrates how to find all the ExtrinsicObjects that are defined to be different from
1995 the ExtrinsicObject that have a name containing the string "MyDocument" .

```
1996 <rs:RequestSlotList>  
1997   <rim:Slot  
1998     name="urn:oasis:names:tc:ebxml-  
1999 regrep:3.0:rs:AdhocQueryRequest:queryId">  
2000     <rim:ValueList>  
2001       <rim:Value>urn:oasis:names:tc:ebxml-  
2002 regrep:profile:webontology:query:FindDifferentExtrinsicObjects</rim:Value  
2003 >  
2004     </rim:ValueList>  
2005   </rim:Slot>  
2006   <rim:Slot name="urn:oasis:names:tc:ebxml-  
2007 regrep:rs:AdhocQueryRequest:queryId">  
2008     <rim:ValueList>  
2009       <rim:Value>urn:oasis:names:tc:ebxml-  
2010 regrep:profile:webontology:query:FindDifferentExtrinsicObjects</rim:Value  
2011 >  
2012     </rim:ValueList>  
2013   </rim:Slot>  
2014   <rim:Slot name="$extrinsicObjectName">  
2015     <rim:ValueList>  
2016       <rim:Value>%MyDocument%</rim:Value>  
2017     </rim:ValueList>  
2018   </rim:Slot>  
2019 </rs:RequestSlotList>  
2020  
2021 <query:ResponseOption returnComposedObjects="true"  
2022   returnType="LeafClassWithRepositoryItem"/>  
2023  
2024 <rim:AdhocQuery id="temporaryId">  
2025   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
2026 regrep:QueryLanguage:SQL-92">  
2027     </rim:QueryExpression>  
2028 </rim:AdhocQuery>  
2029
```

2030 Example of DifferentExtrinsicObjects Discovery Query

2031 6.10 AllDifferentRegistryObject Discovery Query

2032 The AllDifferentRegistryObjects discovery query MUST be implemented by an ebXML Registry
2033 implementing this profile. Given a RegistryObject, it allows the discovery of all the other member
2034 "RegistryObjects" of a Registry package that are defined to be the different from each other through a
2035 allDifferent slot. The canonical query corresponding to this discovery query is presented in Section
2036 7.3.10.

2037 6.10.1 Parameter \$registryObjectName

2038 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2039 value of RegistryObjects.

2040 6.10.2 Example of AllDifferentRegistryObjects Discovery Query

2041 The following example illustrates how to find all the RegistryObjects that are defined to be different from
2042 the RegistryObject that have a name containing the string "MyDocument" .

```

2043
2044 <rs:RequestSlotList>
2045   <rim:Slot
2046     name="urn:oasis:names:tc:ebxml-
2047   regrep:3.0:rs:AdhocQueryRequest:queryId">
2048     <rim:ValueList>
2049       <rim:Value>urn:oasis:names:tc:ebxml-
2050   regrep:profile:webontology:query:FindAllDifferent</rim:Value>
2051     </rim:ValueList>
2052   </rim:Slot>
2053   <rim:Slot name="urn:oasis:names:tc:ebxml-
2054   regrep:rs:AdhocQueryRequest:queryId">
2055     <rim:ValueList>
2056       <rim:Value>urn:oasis:names:tc:ebxml-
2057   regrep:profile:webontology:query:FindAllDifferent</rim:Value>
2058     </rim:ValueList>
2059   </rim:Slot>
2060   <rim:Slot name="$registryObjectName">
2061     <rim:ValueList>
2062       <rim:Value>%MyDocument%</rim:Value>
2063     </rim:ValueList>
2064   </rim:Slot>
2065 </rs:RequestSlotList>
2066
2067 <query:ResponseOption returnComposedObjects="true"
2068   returnType="LeafClassWithRepositoryItem"/>
2069
2070 <rim:AdhocQuery id="temporaryId">
2071   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2072   regrep:QueryLanguage:SQL-92">
2073     </rim:QueryExpression>
2074 </rim:AdhocQuery>

```

2075 Example of AllDifferentRegistryObjects Discovery Query

2076 6.11 ObjectProperties Discovery Query

2077 The ObjectProperties discovery query MUST be implemented by an ebXML Registry implementing this
2078 profile. It allows the discovery of all of the objectProperties of a given classification node. The canonical
2079 query corresponding to this discovery query is presented in Section 7.3.11.

2080 6.11.1 Parameter \$className

2081 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2082 value of ClassificationNodes.

2083 6.11.2 Example of ObjectProperties Discovery Query

2084 The following example illustrates how to find all the object properties of a given classification node having
2085 a name containing "AirServices" .

```

2086
2087 <rs:RequestSlotList>
2088   <rim:Slot
2089     name="urn:oasis:names:tc:ebxml-
2090   regrep:3.0:rs:AdhocQueryRequest:queryId">
2091     <rim:ValueList>
2092       <rim:Value>urn:oasis:names:tc:ebxml-
2093   regrep:profile:webontology:query:FindObjectProperties</rim:Value>
2094     </rim:ValueList>
2095   </rim:Slot>
2096   <rim:Slot name="urn:oasis:names:tc:ebxml-
2097   regrep:rs:AdhocQueryRequest:queryId">
2098     <rim:ValueList>

```

```

2099     <rim:Value>urn:oasis:names:tc:ebxml-
2100 regrep:profile:webontology:query:FindObjectProperties</rim:Value>
2101   </rim:ValueList>
2102 </rim:Slot>
2103   <rim:Slot name="$className">
2104     <rim:ValueList>
2105       <rim:Value>%AirServices%</rim:Value>
2106     </rim:ValueList>
2107   </rim:Slot>
2108 </rs:RequestSlotList>
2109
2110 <query:ResponseOption returnComposedObjects="true"
2111   returnType="LeafClassWithRepositoryItem"/>
2112
2113 <rim:AdhocQuery id="temporaryId">
2114   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2115 regrep:QueryLanguage:SQL-92">
2116     </rim:QueryExpression>
2117 </rim:AdhocQuery>

```

Example of ObjectProperties Discovery Query

2119 6.12 ImmediateInheritedObjectProperties Discovery Query

2120 The ImmediateInheritedObjectProperties discovery query MUST be implemented by an ebXML Registry
 2121 implementing this profile. It allows the discovery of all of the objectProperties of a given classification node
 2122 including the ones inherited from its immediate super classes. The canonical query corresponding to this
 2123 discovery query is presented in Section 7.3.12.

2124 6.12.1 Parameter \$className

2125 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
 2126 value of ClassificationNodes.

2127 6.12.2 Example of ImmediateInheritedObjectProperties Discovery Query

2128 The following example illustrates how to find all the object properties of a given classification node having
 2129 a name containing "AirServices" including the ones inherited from its immediate super classes.

```

2130
2131 <rs:RequestSlotList>
2132   <rim:Slot
2133     name="urn:oasis:names:tc:ebxml-
2134 regrep:3.0:rs:AdhocQueryRequest:queryId">
2135     <rim:ValueList>
2136       <rim:Value>urn:oasis:names:tc:ebxml-
2137 regrep:profile:webontology:query:FindImmediateInheritedObjectProperties</
2138 rim:Value>
2139     </rim:ValueList>
2140   </rim:Slot>
2141   <rim:Slot name="urn:oasis:names:tc:ebxml-
2142 regrep:rs:AdhocQueryRequest:queryId">
2143     <rim:ValueList>
2144       <rim:Value>urn:oasis:names:tc:ebxml-
2145 regrep:profile:webontology:query:FindImmediateInheritedObjectProperties</
2146 rim:Value>
2147     </rim:ValueList>
2148   </rim:Slot>
2149   <rim:Slot name="$className">
2150     <rim:ValueList>
2151       <rim:Value>%AirServices%</rim:Value>
2152     </rim:ValueList>
2153   </rim:Slot>
2154 </rs:RequestSlotList>
2155

```

```

2156 <query:ResponseOption returnComposedObjects="true"
2157     returnType="LeafClassWithRepositoryItem"/>
2158
2159 <rim:AdhocQuery id="temporaryId">
2160   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2161   regrep:QueryLanguage:SQL-92">
2162     </rim:QueryExpression>
2163   </rim:AdhocQuery>

```

Example of ImmediateInheritedObjectProperties Discovery Query

2165 6.13 AllInheritedObjectProperties Discovery Query

2166 It should be noted that, given a class, finding the object properties inherited from immediate super classes
 2167 is necessary but not sufficient. Given a class, it should be possible to retrieve all of the object properties
 2168 inherited from its super classes. This requires a recursion mechanism in SQL queries.

2169 The AllInheritedObjectProperties discovery query MAY be implemented by an ebXML Registry
 2170 implementing this profile. It allows the discovery of all inherited ObjectProperties recursively of a given
 2171 ClassificationNode in a ClassificationScheme in an ebXML Registry implementation supporting recursion.

2172 The canonical query corresponding to this discovery query is presented in Section 7.3.13.

2173 6.13.1 Parameter \$className

2174 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
 2175 value of ClassificationNodes.

2176 6.13.2 Example of AllInheritedObjectProperties Discovery Query

2177 The following example illustrates how to find all the object properties of a given classification node having
 2178 a name containing "AirReservationServices" including the ones inherited from all of its super classes
 2179 recursively, if the query is implemented as an Adhoc Query.

```

2180
2181 <rs:RequestSlotList>
2182   <rim:Slot
2183     name="urn:oasis:names:tc:ebxml-
2184     regrep:3.0:rs:AdhocQueryRequest:queryId">
2185     <rim:ValueList>
2186       <rim:Value>urn:oasis:names:tc:ebxml-
2187       regrep:profile:webontology:query:FindAllInheritedObjectProperties</rim:Va
2188       lue>
2189     </rim:ValueList>
2190   </rim:Slot>
2191   <rim:Slot name="urn:oasis:names:tc:ebxml-
2192   regrep:rs:AdhocQueryRequest:queryId">
2193     <rim:ValueList>
2194       <rim:Value>urn:oasis:names:tc:ebxml-
2195       regrep:profile:webontology:query:FindAll
2196       InheritedObjectProperties</rim:Value>
2197     </rim:ValueList>
2198   </rim:Slot>
2199   <rim:Slot name="$className">
2200     <rim:ValueList>
2201       <rim:Value>%AirReservationServices%</rim:Value>
2202     </rim:ValueList>
2203   </rim:Slot>
2204 </rs:RequestSlotList>
2205
2206 <query:ResponseOption returnComposedObjects="true"
2207     returnType="LeafClassWithRepositoryItem"/>
2208
2209 <rim:AdhocQuery id="temporaryId">
2210   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2211   regrep:QueryLanguage:SQL-92">

```

2212 </rim:QueryExpression>
2213 </rim:AdhocQuery>

2214 Example of AllInheritedObjectProperties Discovery Query

2215 6.14 DatatypeProperties Discovery Query

2216 The DatatypeProperties discovery query MUST be implemented by an ebXML Registry implementing this
2217 profile. It allows the discovery of all of the datatypeProperties of a given classification node. The
2218 canonical query corresponding to this discovery query is presented in Section 7.3.14.

2219 6.14.1 Parameter \$className

2220 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2221 value of ClassificationNodes.

2222 6.14.2 Example of DatatypeProperties Discovery Query

2223 The following example illustrates how to find all the datatype properties of a given classification node
2224 having a name containing "AirReservationServices" .

2225

```
2226 <rs:RequestSlotList>  
2227   <rim:Slot  
2228     name="urn:oasis:names:tc:ebxml-  
2229     regrep:3.0:rs:AdhocQueryRequest:queryId">  
2230     <rim:ValueList>  
2231       <rim:Value>urn:oasis:names:tc:ebxml-  
2232       regrep:profile:webontology:query:FindDatatypeProperties</rim:Value>  
2233     </rim:ValueList>  
2234   </rim:Slot>  
2235   <rim:Slot name="urn:oasis:names:tc:ebxml-  
2236   regrep:rs:AdhocQueryRequest:queryId">  
2237     <rim:ValueList>  
2238       <rim:Value>urn:oasis:names:tc:ebxml-  
2239       regrep:profile:webontology:query:FindDatatypeProperties</rim:Value>  
2240     </rim:ValueList>  
2241   </rim:Slot>  
2242   <rim:Slot name="$className">  
2243     <rim:ValueList>  
2244       <rim:Value>%AirReservationServices%</rim:Value>  
2245     </rim:ValueList>  
2246   </rim:Slot>  
2247 </rs:RequestSlotList>  
2248  
2249 <query:ResponseOption returnComposedObjects="true"  
2250   returnType="LeafClassWithRepositoryItem"/>  
2251  
2252 <rim:AdhocQuery id="temporaryId">  
2253   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
2254   regrep:QueryLanguage:SQL-92">  
2255     </rim:QueryExpression>  
2256 </rim:AdhocQuery>
```

2257 Example of DatatypeProperties Discovery Query

2258 6.15 AllInheritedDatatypeProperties Discovery Query

2259 It should be noted that, given a class, finding the datatype properties inherited from immediate super
2260 classes is necessary but not sufficient. Given a class, it should be possible to retrieve all of the datatype
2261 properties inherited from its super classes. This requires a recursion mechanism in SQL queries.

2262 The AllInheritedDatatypeProperties discovery query MAY be implemented by an ebXML Registry
2263 implementing this profile. It allows the discovery of all inherited DatatypeProperties recursively of a given
2264 ClassificationNode in a ClassificationScheme in an ebXML Registry implementation supporting recursion.

2265 The canonical query corresponding to this discovery query is presented in Section 7.3.15.

2266 6.15.1 Parameter \$className

2267 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2268 value of ClassificationNodes.

2269 6.15.2 Example of AllInheritedDatatypeProperties Discovery Query

2270 The following example illustrates how to find all the datatype properties of a given classification node
2271 having a name containing "AirReservationServices" including the ones inherited from all of its super
2272 classes recursively, if the query is implemented as an Adhoc Query.

2273

```
2274 <rs:RequestSlotList>
2275   <rim:Slot
2276     name="urn:oasis:names:tc:ebxml-
2277   regrep:3.0:rs:AdhocQueryRequest:queryId">
2278     <rim:ValueList>
2279       <rim:Value>urn:oasis:names:tc:ebxml-
2280   regrep:profile:webontology:query:FindAllInheritedDatatypeProperties</rim:
2281   Value>
2282     </rim:ValueList>
2283   </rim:Slot>
2284   <rim:Slot name="urn:oasis:names:tc:ebxml-
2285   regrep:rs:AdhocQueryRequest:queryId">
2286     <rim:ValueList>
2287       <rim:Value>urn:oasis:names:tc:ebxml-
2288   regrep:profile:webontology:query:FindAllInheritedDatatypeProperties</rim:
2289   Value>
2290     </rim:ValueList>
2291   </rim:Slot>
2292   <rim:Slot name="$className">
2293     <rim:ValueList>
2294       <rim:Value>%AirReservationServices %</rim:Value>
2295     </rim:ValueList>
2296   </rim:Slot>
2297 </rs:RequestSlotList>
2298
2299 <query:ResponseOption returnComposedObjects="true"
2300   returnType="LeafClassWithRepositoryItem"/>
2301
2302 <rim:AdhocQuery id="temporaryId">
2303   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2304   regrep:QueryLanguage:SQL-92">
2305     </rim:QueryExpression>
2306 </rim:AdhocQuery>
```

2307 Example of AllInheritedDatatypeProperties Discovery Query

2308 6.16 TransitiveRelationships Discovery Query

2309 To make any use of the transitive property in ebXML registries, coding is necessary to find out the implied
2310 information. The TransitiveRelationships discovery query MUST be implemented by an ebXML Registry
2311 implementing this profile to handle this semantics.

2312 Given a class which is a source of a transitive property, this discovery query retrieves not only the target
2313 objects of a given transitive property, but if the target objects have the same property, it retrieves their
2314 target objects too. The canonical query corresponding to this discovery query is presented in Section
2315 7.3.16.

2316 6.16.1 Parameter \$className

2317 This parameter's value SHALL specify a string containing a pattern to match against the name attribute

2318 value of ClassificationNodes.

2319 6.16.2 Parameter \$propertyName

2320 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2321 value of Associations that have associationType of Property

2322 6.16.3 Example of TransitiveRelationships Discovery Query

2323 The following example illustrates how to retrieve all the target objects of the "succeeds" property of the
2324 "AirReservationServices" including the target objects implied by a transitive property relationship.

2325

```
2326 <rs:RequestSlotList>
2327   <rim:Slot
2328     name="urn:oasis:names:tc:ebxml-
2329   regrep:3.0:rs:AdhocQueryRequest:queryId">
2330     <rim:ValueList>
2331       <rim:Value>urn:oasis:names:tc:ebxml-
2332   regrep:profile:webontology:query:FindTransitiveRelationships</rim:Value>
2333     </rim:ValueList>
2334   </rim:Slot>
2335   <rim:Slot name="urn:oasis:names:tc:ebxml-
2336   regrep:rs:AdhocQueryRequest:queryId">
2337     <rim:ValueList>
2338       <rim:Value>urn:oasis:names:tc:ebxml-
2339   regrep:profile:webontology:query:FindTransitiveRelationships</rim:Value>
2340     </rim:ValueList>
2341   </rim:Slot>
2342   <rim:Slot name="$className">
2343     <rim:ValueList>
2344       <rim:Value>%AirReservationServices%</rim:Value>
2345     </rim:ValueList>
2346   </rim:Slot>
2347   <rim:Slot name="$propertyName">
2348     <rim:ValueList>
2349       <rim:Value>%succeeds%</rim:Value>
2350     </rim:ValueList>
2351   </rim:Slot>
2352 </rs:RequestSlotList>
2353
2354 <query:ResponseOption returnComposedObjects="true"
2355   returnType="LeafClassWithRepositoryItem"/>
2356
2357 <rim:AdhocQuery id="temporaryId">
2358   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2359   regrep:QueryLanguage:SQL-92">
2360     </rim:QueryExpression>
2361 </rim:AdhocQuery>
```

2362

Example of TransitiveRelationships Discovery Query

2363 6.17 TargetObjects Discovery Query

2364 The TargetObjects discovery query MUST be implemented by an ebXML Registry implementing this
2365 profile. It allows the discovery of the targetObjects from the Registry, given a Classification Node
2366 (sourceObject) and a property name (Association Type). The canonical query corresponding to this
2367 discovery query is presented in Section 7.3.17.

2368 6.17.1 Parameter \$className

2369 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2370 value of ClassificationNodes.

2371 6.17.2 Parameter \$propertyName

2372 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2373 value of Associations that have associationType of Property.

2374 6.17.3 Example of TargetObjects Discovery Query

2375 The following example illustrates how to retrieve all the target objects of the "paymentMethod" property of
2376 the "AirReservationServices".

2377

```
2378 <rs:RequestSlotList>
2379   <rim:Slot
2380     name="urn:oasis:names:tc:ebxml-
2381   regrep:3.0:rs:AdhocQueryRequest:queryId">
2382     <rim:ValueList>
2383       <rim:Value>urn:oasis:names:tc:ebxml-
2384   regrep:profile:webontology:query:FindTargetObjects</rim:Value>
2385     </rim:ValueList>
2386   </rim:Slot>
2387   <rim:Slot name="urn:oasis:names:tc:ebxml-
2388   regrep:rs:AdhocQueryRequest:queryId">
2389     <rim:ValueList>
2390       <rim:Value>urn:oasis:names:tc:ebxml-
2391   regrep:profile:webontology:query:FindTargetObjects</rim:Value>
2392     </rim:ValueList>
2393   </rim:Slot>
2394   <rim:Slot name="$className">
2395     <rim:ValueList>
2396       <rim:Value>%AirReservationServices%</rim:Value>
2397     </rim:ValueList>
2398   </rim:Slot>
2399   <rim:Slot name="$propertyName">
2400     <rim:ValueList>
2401       <rim:Value>%paymentMethod%</rim:Value>
2402     </rim:ValueList>
2403   </rim:Slot>
2404 </rs:RequestSlotList>
2405
2406 <query:ResponseOption returnComposedObjects="true"
2407   returnType="LeafClassWithRepositoryItem"/>
2408
2409 <rim:AdhocQuery id="temporaryId">
2410   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2411   regrep:QueryLanguage:SQL-92">
2412     </rim:QueryExpression>
2413 </rim:AdhocQuery>
```

2414

Example of TargetObjects Discovery Query

2415

2416 6.18 TargetObjectsInverseOf Discovery Query

2417 The TargetObjectsInverseOf discovery query MUST be implemented by an ebXML Registry implementing
2418 this profile. Given a Classification Node (sourceObject) and a property name (Association Type), this
2419 query retrieves the source objects of the properties which are stated to be inverseOf the property name
2420 given as a parameter, and considering the Classification Node name as the targetObject of these
2421 properties. The canonical query corresponding to this discovery query is presented in Section 7.3.18.

2422 6.18.1 Parameter \$className

2423 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2424 value of ClassificationNodes.

2425 6.18.2 Parameter \$propertyName

2426 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2427 value of Associations that have associationType of Property.

2428 6.18.3 Example of TargetObjectsInverseOf Discovery Query

2429 The following example illustrates how to retrieve all the source objects of the properties which are stated
2430 to the the inverseOf the property "succeeds", considering the "AirReservationServices" as the target object
2431 of these properties.

2432

```
2433 <rs:RequestSlotList>  
2434   <rim:Slot  
2435     name="urn:oasis:names:tc:ebxml-  
2436   regrep:3.0:rs:AdhocQueryRequest:queryId">  
2437     <rim:ValueList>  
2438       <rim:Value>urn:oasis:names:tc:ebxml-  
2439   regrep:profile:webontology:query:FindTOinverseOf</rim:Value>  
2440     </rim:ValueList>  
2441   </rim:Slot>  
2442   <rim:Slot name="urn:oasis:names:tc:ebxml-  
2443   regrep:rs:AdhocQueryRequest:queryId">  
2444     <rim:ValueList>  
2445       <rim:Value>urn:oasis:names:tc:ebxml-  
2446   regrep:profile:webontology:query:FindTOinverseOf</rim:Value>  
2447     </rim:ValueList>  
2448   </rim:Slot>  
2449   <rim:Slot name="$className">  
2450     <rim:ValueList>  
2451       <rim:Value>%AirReservationServices%</rim:Value>  
2452     </rim:ValueList>  
2453   </rim:Slot>  
2454   <rim:Slot name="$propertyName">  
2455     <rim:ValueList>  
2456       <rim:Value>%succeeds%</rim:Value>  
2457     </rim:ValueList>  
2458   </rim:Slot>  
2459 </rs:RequestSlotList>  
  
2460  
2461 <query:ResponseOption returnComposedObjects="true"  
2462   returnType="LeafClassWithRepositoryItem"/>  
2463  
2464 <rim:AdhocQuery id="temporaryId">  
2465   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
2466   regrep:QueryLanguage:SQL-92">  
2467     </rim:QueryExpression>  
2468 </rim:AdhocQuery>
```

2469 Example of TargetObjectsInverseOf Discovery Query

2470

2471 6.19 InverseRanges Discovery Query

2472 The InverseRanges discovery query MUST be implemented by an ebXML Registry implementing this
2473 profile to handle this semantics. Given a Classification Node (sourceObject) and a property name
2474 (Association Type), this query retrieves not only the target objects of this property, but also the source
2475 objects of the properties which are stated to be inverseOf the property name given as a parameter, and
2476 considering the Classification Node name as the targetObject of these properties. This query can be
2477 thought as the union of the queries presented in Sections 6.17 and 6.18. The canonical query
2478 corresponding to this discovery query is presented in Section 7.3.19.

2479 6.19.1 Parameter \$className

2480 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2481 value of ClassificationNodes.

2482 6.19.2 Parameter \$propertyName

2483 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2484 value of Associations that have associationType of Property

2485 6.19.3 Example of InverseRanges Discovery Query

2486 Consider, for example, the "succeeds" property defined in Section 4.3.3. To denote that a certain Web
2487 service instance precedes another during execution, we may define the "precedes" property as an inverse
2488 of the "succeeds" property as follows:

2489

```
2490 <owl:ObjectProperty rdf:ID="precedes">  
2491   <owl:inverseOf rdf:resource="#succeeds" />  
2492 </owl:ObjectProperty>
```

2493 **Example owl:inverseOf Property**

2494 Assume that we want to find all the Web services which can succeed a given Web service. In such a
2495 case, we need not only find all the Web services which succeeds this given Web service, that is the target
2496 objects of "succeeds" Association instance, but we also need to find all the sourceObjects of the
2497 "precedes" Association instance since "precedes" is declared to be the "inverseOf" succeeds Association
2498 instance.

2499 The following example illustrates how to retrieve all the services that "succeeds" "AirReservationServices"
2500 by also making use of its "precedes" property.

2501

```
2502 <rs:RequestSlotList>  
2503   <rim:Slot  
2504     name="urn:oasis:names:tc:ebxml-  
2505     regrep:3.0:rs:AdhocQueryRequest:queryId">  
2506     <rim:ValueList>  
2507       <rim:Value>urn:oasis:names:tc:ebxml-  
2508       regrep:profile:webontology:query:FindInverseRanges</rim:Value>  
2509     </rim:ValueList>  
2510   </rim:Slot>  
2511   <rim:Slot name="urn:oasis:names:tc:ebxml-  
2512   regrep:rs:AdhocQueryRequest:queryId">  
2513     <rim:ValueList>  
2514       <rim:Value>urn:oasis:names:tc:ebxml-  
2515       regrep:profile:webontology:query:FindInverseRanges</rim:Value>  
2516     </rim:ValueList>  
2517   </rim:Slot>  
2518   <rim:Slot name="$className">  
2519     <rim:ValueList>  
2520       <rim:Value>%AirReservationServices%</rim:Value>  
2521     </rim:ValueList>  
2522   </rim:Slot>  
2523   <rim:Slot name="$propertyName">  
2524     <rim:ValueList>  
2525       <rim:Value>%succeeds%</rim:Value>  
2526     </rim:ValueList>  
2527   </rim:Slot>  
2528 </rs:RequestSlotList>  
2529  
2530 <query:ResponseOption returnComposedObjects="true"  
2531   returnType="LeafClassWithRepositoryItem"/>  
2532  
2533 <rim:AdhocQuery id="temporaryId">
```

```

2534     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2535     regrep:QueryLanguage:SQL-92">
2536     </rim:QueryExpression>
2537 </rim:AdhocQuery>

```

2538 Example of InverseRanges Discovery Query

2539 6.20 SymmetricProperties Discovery Query

2540 The SymmetricProperties discovery query MUST be implemented by an ebXML Registry implementing
2541 this profile. It allows the discovery of all of the Symmetric Properties of a given classification node. The
2542 canonical query corresponding to this discovery query is presented in Section 7.3.20.

2543 6.20.1 Parameter \$className

2544 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2545 value of ClassificationNodes.

2546 6.20.2 Example of SymmetricProperties Discovery Query

2547 The following example illustrates how to find all the symmetric properties of a given classification node
2548 having a name containing "AirReservationServices" .

```

2549
2550 <rs:RequestSlotList>
2551   <rim:Slot
2552     name="urn:oasis:names:tc:ebxml-
2553     regrep:3.0:rs:AdhocQueryRequest:queryId">
2554     <rim:ValueList>
2555       <rim:Value>urn:oasis:names:tc:ebxml-
2556       regrep:profile:webontology:query:FindSymmetricProperties</rim:Value>
2557     </rim:ValueList>
2558   </rim:Slot>
2559   <rim:Slot name="urn:oasis:names:tc:ebxml-
2560   regrep:rs:AdhocQueryRequest:queryId">
2561     <rim:ValueList>
2562       <rim:Value>urn:oasis:names:tc:ebxml-
2563       regrep:profile:webontology:query:FindSymmetricProperties</rim:Value>
2564     </rim:ValueList>
2565   </rim:Slot>
2566   <rim:Slot name="$className">
2567     <rim:ValueList>
2568       <rim:Value>%AirReservationServices%</rim:Value>
2569     </rim:ValueList>
2570   </rim:Slot>
2571 </rs:RequestSlotList>
2572
2573 <query:ResponseOption returnComposedObjects="true"
2574   returnType="LeafClassWithRepositoryItem"/>
2575
2576 <rim:AdhocQuery id="temporaryId">
2577   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2578   regrep:QueryLanguage:SQL-92">
2579   </rim:QueryExpression>
2580 </rim:AdhocQuery>

```

2581 Example of SymmetricProperties Discovery Query

2582 6.21 FunctionalProperties Discovery Query

2583 The FunctionalProperties discovery query MUST be implemented by an ebXML Registry implementing
2584 this profile. It allows the discovery of all of the Functional Properties of a given classification node. The
2585 canonical query corresponding to this discovery query is presented in Section 7.3.21.

2586 6.21.1 Parameter \$className

2587 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2588 value of ClassificationNodes.

2589 6.21.2 Example of FunctionalProperties Discovery Query

2590 The following example illustrates how to find all the functional properties of a given classification node
2591 having a name containing "AirReservationServices" .

2592

```
2593 <rs:RequestSlotList>  
2594   <rim:Slot  
2595     name="urn:oasis:names:tc:ebxml-  
2596   regrep:3.0:rs:AdhocQueryRequest:queryId">  
2597     <rim:ValueList>  
2598       <rim:Value>urn:oasis:names:tc:ebxml-  
2599   regrep:profile:webontology:query:FindFunctionalProperties</rim:Value>  
2600     </rim:ValueList>  
2601   </rim:Slot>  
2602   <rim:Slot name="urn:oasis:names:tc:ebxml-  
2603   regrep:rs:AdhocQueryRequest:queryId">  
2604     <rim:ValueList>  
2605       <rim:Value>urn:oasis:names:tc:ebxml-  
2606   regrep:profile:webontology:query:FindFunctionalProperties</rim:Value>  
2607     </rim:ValueList>  
2608   </rim:Slot>  
2609   <rim:Slot name="$className">  
2610     <rim:ValueList>  
2611       <rim:Value>%AirReservationServices%</rim:Value>  
2612     </rim:ValueList>  
2613   </rim:Slot>  
2614 </rs:RequestSlotList>  
  
2615 <query:ResponseOption returnComposedObjects="true"  
2616   returnType="LeafClassWithRepositoryItem"/>  
2617  
2618 <rim:AdhocQuery id="temporaryId">  
2619   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
2620   regrep:QueryLanguage:SQL-92">  
2621     </rim:QueryExpression>  
2622   </rim:QueryExpression>  
2623 </rim:AdhocQuery>
```

2624

Example of Functional Properties Discovery Query

2625 6.22 InverseFunctionalProperties Discovery Query

2626 The InverseFunctionalProperties discovery query MUST be implemented by an ebXML Registry
2627 implementing this profile. It allows the discovery of all of the Inverse Functional Properties of a given
2628 classification node. The canonical query corresponding to this discovery query is presented in Section
2629 7.3.22.

2630 6.22.1 Parameter \$className

2631 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2632 value of ClassificationNodes.

2633 6.22.2 Example of InverseFunctionalProperties Discovery Query

2634 The following example illustrates how to find all the inverse functional properties of a given classification
2635 node having a name containing "AirReservationServices" .

2636

```
2637 <rs:RequestSlotList>
```

```

2638     <rim:Slot
2639         name="urn:oasis:names:tc:ebxml-
2640 regrep:3.0:rs:AdhocQueryRequest:queryId">
2641         <rim:ValueList>
2642             <rim:Value>urn:oasis:names:tc:ebxml-
2643 regrep:profile:webontology:query:FindInverseFunctionalProperties</rim:Val
2644 ue>
2645         </rim:ValueList>
2646     </rim:Slot>
2647     <rim:Slot name="urn:oasis:names:tc:ebxml-
2648 regrep:rs:AdhocQueryRequest:queryId">
2649         <rim:ValueList>
2650             <rim:Value>urn:oasis:names:tc:ebxml-
2651 regrep:profile:webontology:query:FindInverseFunctionalProperties</rim:Val
2652 ue>
2653         </rim:ValueList>
2654     </rim:Slot>
2655     <rim:Slot name="$className">
2656         <rim:ValueList>
2657             <rim:Value>%AirReservationServices%</rim:Value>
2658         </rim:ValueList>
2659     </rim:Slot>
2660 </rs:RequestSlotList>
2661
2662 <query:ResponseOption returnComposedObjects="true"
2663     returnType="LeafClassWithRepositoryItem"/>
2664
2665 <rim:AdhocQuery id="temporaryId">
2666     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2667 regrep:QueryLanguage:SQL-92">
2668     </rim:QueryExpression>
2669 </rim:AdhocQuery>

```

2670 Example of InverseFunctional Properties Discovery Query

2671 6.23 Instances Discovery Query

2672 When an intersection definition is used to create a complex class (a new ClassificationNode) in RIM as
2673 described in Section 4.6, it becomes possible to infer that the objects (instances) classified by all of the
2674 classes (ClassificationNodes) specified in the Intersection definition, are also classified by this complex
2675 class.

2676 The Instances discovery query MUST be implemented by an ebXML Registry implementing this profile. It
2677 allows the discovery of all of the direct instances of a given classification node and if it is a complex class
2678 which is an intersection two classes, it also allows to retrieve the intersection of the instances of both of
2679 the classes involved in the intersection definition. The canonical query corresponding to this discovery
2680 query is presented in Section 7.3.23.

2681 6.23.1 Parameter \$className

2682 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2683 value of ClassificationNodes.

2684 6.23.2 Example of Instances Discovery Query

2685 Consider the "AirReservationServices" definition presented in Section 4.6. The following example
2686 illustrates how to find all the direct instances of the "AirReservationServices" and also the instances
2687 classified by both "AirServices" and also the "ReservationServices".

```

2688
2689 <rs:RequestSlotList>
2690     <rim:Slot
2691         name="urn:oasis:names:tc:ebxml-
2692 regrep:3.0:rs:AdhocQueryRequest:queryId">
2693         <rim:ValueList>

```

```

2694         <rim:Value>urn:oasis:names:tc:ebxml-
2695 regrep:profile:webontology:query:FindInstances</rim:Value>
2696     </rim:ValueList>
2697 </rim:Slot>
2698     <rim:Slot name="urn:oasis:names:tc:ebxml-
2699 regrep:rs:AdhocQueryRequest:queryId">
2700     <rim:ValueList>
2701         <rim:Value>urn:oasis:names:tc:ebxml-
2702 regrep:profile:webontology:query:FindInstances</rim:Value>
2703     </rim:ValueList>
2704 </rim:Slot>
2705     <rim:Slot name="$className">
2706     <rim:ValueList>
2707         <rim:Value>%AirReservationServices%</rim:Value>
2708     </rim:ValueList>
2709 </rim:Slot>
2710 </rs:RequestSlotList>
2711
2712 <query:ResponseOption returnComposedObjects="true"
2713     returnType="LeafClassWithRepositoryItem"/>
2714
2715 <rim:AdhocQuery id="temporaryId">
2716     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2717 regrep:QueryLanguage:SQL-92">
2718     </rim:QueryExpression>
2719 </rim:AdhocQuery>

```

Example of Instances Discovery Query

2720

7 Canonical Metadata Definitions

2721

2722 This chapter specifies the canonical metadata defined by this profile.

7.1 ObjectType Extensions

2723

2724 The following new extensions to the canonical ObjectType ClassificationScheme are described by this
2725 profile:

2726

```
2727 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2728 regrep:ObjectType:RegistryObject:ExtrinsicObject"  
2729 lid="urn:oasis:names:tc:ebxml-  
2730 regrep:profile:webontology:ObjectType:RegistryObject:ExtrinsicObject:OWL"  
2731 code="OWL" id="urn:oasis:names:tc:ebxml-  
2732 regrep:profile:webontology:ObjectType:RegistryObject:ExtrinsicObject:OWL"  
2733 >  
2734 <rim:Name>  
2735 <rim:LocalizedString charset="UTF-8" value="OWL"/>  
2736 </rim:Name>  
2737 </rim:ClassificationNode>
```

7.2 AssociationType Extensions

2738

2739 The following new extensions to the AssociationType ClassificationScheme are described by this profile:

2740

```
2741 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2742 regrep:classificationScheme:AssociationType"  
2743 lid="urn:oasis:names:tc:ebxml-  
2744 regrep:profile:webontology:AssociationType:OWL" code="OWL"  
2745 id="urn:oasis:names:tc:ebxml-  
2746 regrep:profile:webontology:AssociationType:OWL">  
2747 <rim:Name>  
2748 <rim:LocalizedString charset="UTF-8" value="OWL"/>  
2749 </rim:Name>  
2750 </rim:ClassificationNode>  
2751 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2752 regrep:profile:webontology:AssociationType:OWL"  
2753 lid="urn:oasis:names:tc:ebxml-  
2754 regrep:profile:webontology:AssociationType:OWL:ObjectProperty"  
2755 code="ObjectProperty" id="urn:oasis:names:tc:ebxml-  
2756 regrep:profile:webontology:AssociationType:OWL:ObjectProperty">  
2757 <rim:Name>  
2758 <rim:LocalizedString charset="UTF-8"  
2759 value="ObjectProperty"/>  
2760 </rim:Name>  
2761 </rim:ClassificationNode>  
2762 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2763 regrep:profile:webontology:AssociationType:OWL"  
2764 lid="urn:oasis:names:tc:ebxml-  
2765 regrep:profile:webontology:AssociationType:OWL:HasProperty"  
2766 code="Property" id="urn:oasis:names:tc:ebxml-  
2767 regrep:profile:webontology:AssociationType:OWL:HasProperty">  
2768 <rim:Name>  
2769 <rim:LocalizedString charset="UTF-8" value="Property"/>  
2770 </rim:Name>  
2771 </rim:ClassificationNode>  
2772 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2773 regrep:profile:webontology:AssociationType:OWL"  
2774 lid="urn:oasis:names:tc:ebxml-  
2775 regrep:profile:webontology:AssociationType:OWL:SubPropertyOf"  
2776 code="SubPropertyOf" id="urn:oasis:names:tc:ebxml-  
2777 regrep:profile:webontology:AssociationType:OWL:SubPropertyOf">  
2778 <rim:Name>
```

```

2779         <rim:LocalizedString charset="UTF-8" value="SubPropertyOf"/>
2780     </rim:Name>
2781 </rim:ClassificationNode>
2782 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2783 regrep:profile:webontology:AssociationType:OWL"
2784 lid="urn:oasis:names:tc:ebxml-
2785 regrep:profile:webontology:AssociationType:OWL:SubClassOf"
2786 code="SubClassOf" id="urn:oasis:names:tc:ebxml-
2787 regrep:profile:webontology:AssociationType:OWL:SubClassOf">
2788     <rim:Name>
2789         <rim:LocalizedString charset="UTF-8" value="SubClassOf"/>
2790     </rim:Name>
2791 </rim:ClassificationNode>
2792
2793 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2794 regrep:profile:webontology:AssociationType:OWL "
2795 lid="urn:oasis:names:tc:ebxml-
2796 regrep:profile:webontology:AssociationType:OWL:IntersectionOf"
2797 code="IntersectionOf" id="urn:oasis:names:tc:ebxml-
2798 regrep:profile:webontology:AssociationType:OWL:IntersectionOf">
2799     <rim:Name>
2800         <rim:LocalizedString charset="UTF-8"
2801 value="IntersectionOf"/>
2802     </rim:Name>
2803 </rim:ClassificationNode>
2804
2805 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2806 regrep:profile:webontology:AssociationType:OWL"
2807 lid="urn:oasis:names:tc:ebxml-
2808 regrep:profile:webontology:AssociationType:OWL:SameAs" code="SameAs"
2809 id="urn:oasis:names:tc:ebxml-
2810 regrep:profile:webontology:AssociationType:OWL:SameAs">
2811     <rim:Name>
2812         <rim:LocalizedString charset="UTF-8" value="SameAs"/>
2813     </rim:Name>
2814 </rim:ClassificationNode>
2815
2816 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2817 regrep:profile:webontology:AssociationType:OWL "
2818 lid="urn:oasis:names:tc:ebxml-
2819 regrep:profile:webontology:AssociationType:OWL:Restriction"
2820 code="restriction" id="urn:oasis:names:tc:ebxml-
2821 regrep:profile:webontology:AssociationType:OWL:Restriction">
2822     <rim:Name>
2823         <rim:LocalizedString charset="UTF-8" value="Restriction"/>
2824     </rim:Name>
2825 </rim:ClassificationNode>
2826
2827 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2828 regrep:profile:webontology:AssociationType:OWL "
2829 lid="urn:oasis:names:tc:ebxml-
2830 regrep:profile:webontology:AssociationType:OWL:DifferentFrom"
2831 code="DifferentFrom" id="urn:oasis:names:tc:ebxml-
2832 regrep:profile:webontology:AssociationType:OWL:DifferentFrom">
2833     <rim:Name>
2834         <rim:LocalizedString charset="UTF-8" value="DifferentFrom"/>
2835     </rim:Name>
2836 </rim:ClassificationNode>
2837
2838 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2839 regrep:profile:webontology:AssociationType:OWL "
2840 lid="urn:oasis:names:tc:ebxml-
2841 regrep:profile:webontology:AssociationType:OWL:DatatypeProperty"
2842 code="DatatypeProperty" id="urn:oasis:names:tc:ebxml-
2843 regrep:profile:webontology:AssociationType:OWL:DatatypeProperty">
2844     <rim:Name>

```

```

2845         <rim:LocalizedString charset="UTF-8"
2846 value="DatatypeProperty"/>
2847         </rim:Name>
2848     </rim:ClassificationNode>
2849
2850     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2851     regrep:profile:webontology:AssociationType:OWL "
2852     lid="urn:oasis:names:tc:ebxml-
2853     regrep:profile:webontology:AssociationType:OWL:TransitiveProperty"
2854     code="TransitiveProperty" id="urn:oasis:names:tc:ebxml-
2855     regrep:profile:webontology:AssociationType:OWL:TransitiveProperty">
2856         <rim:Name>
2857             <rim:LocalizedString charset="UTF-8"
2858             value="TransitiveProperty"/>
2859             </rim:Name>
2860         </rim:ClassificationNode>
2861
2862     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2863     regrep:profile:webontology:AssociationType:OWL "
2864     lid="urn:oasis:names:tc:ebxml-
2865     regrep:profile:webontology:AssociationType:OWL:InverseOf"
2866     code="InverseOf" id="urn:oasis:names:tc:ebxml-
2867     regrep:profile:webontology:AssociationType:OWL:InverseOf">
2868         <rim:Name>
2869             <rim:LocalizedString charset="UTF-8" value="InverseOf"/>
2870             </rim:Name>
2871         </rim:ClassificationNode>
2872
2873     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2874     regrep:profile:webontology:AssociationType:OWL "
2875     lid="urn:oasis:names:tc:ebxml-
2876     regrep:profile:webontology:AssociationType:OWL:SymmetricProperty"
2877     code="SymmetricProperty" id="urn:oasis:names:tc:ebxml-
2878     regrep:profile:webontology:AssociationType:OWL:SymmetricProperty">
2879         <rim:Name>
2880             <rim:LocalizedString charset="UTF-8"
2881             value="SymmetricProperty"/>
2882             </rim:Name>
2883         </rim:ClassificationNode>
2884
2885     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2886     regrep:profile:webontology:AssociationType:OWL "
2887     lid="urn:oasis:names:tc:ebxml-
2888     regrep:profile:webontology:AssociationType:OWL:FunctionalProperty"
2889     code="FunctionalProperty" id="urn:oasis:names:tc:ebxml-
2890     regrep:profile:webontology:AssociationType:OWL:FunctionalProperty">
2891         <rim:Name>
2892             <rim:LocalizedString charset="UTF-8"
2893             value="FunctionalProperty"/>
2894             </rim:Name>
2895         </rim:ClassificationNode>
2896
2897     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2898     regrep:profile:webontology:AssociationType:OWL "
2899     lid="urn:oasis:names:tc:ebxml-
2900     regrep:profile:webontology:AssociationType:OWL:InverseFunctionalProperty"
2901     code="InverseFunctionalProperty" id="urn:oasis:names:tc:ebxml-
2902     regrep:profile:webontology:AssociationType:OWL:InverseFunctionalProperty"
2903     >
2904         <rim:Name>
2905             <rim:LocalizedString charset="UTF-8"
2906             value="InverseFunctionalProperty"/>
2907             </rim:Name>
2908     </rim:ClassificationNode>

```

```

2909 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2910 regrep:profile:webontology:AssociationType:OWL "
2911 lid="urn:oasis:names:tc:ebxml-
2912 regrep:profile:webontology:AssociationType:OWL:SeeAlso" code="SeeAlso"
2913 id="urn:oasis:names:tc:ebxml-
2914 regrep:profile:webontology:AssociationType:OWL:SeeAlso">
2915   <rim:Name>
2916     <rim:LocalizedString charset="UTF-8" value="SeeAlso"/>
2917   </rim:Name>
2918 </rim:ClassificationNode>

```

Extensions to the AssociationType ClassificationScheme

2920 7.3 Canonical Queries

2921 The following new canonical queries are described by this profile. Note that while these queries are
 2922 complex, the complexity is hidden from clients by exposing only the query parameters to them.

2923 7.3.1 All SuperProperties Discovery Query

2924 Recursion is not supported by SQL-92, for this reason the stored procedure for this query coded in SQL
 2925 99 Standard is available from:

2926 <http://www.srdc.metu.edu.tr/ebxml/ebXMLRegistryProfileForOWL/StoredProceduresSupportingebXMLRegistryProfileforOWL.htm>.

2928

2929 7.3.2 Immediate SuperClass Discovery Query

```

2930 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2931 regrep:profile:webontology:query:FindImmediateSuperClasses"
2932 id="urn:oasis:names:tc:ebxml-
2933 regrep:profile:webontology:query:FindImmediateSuperClasses">
2934   <rim:Name>
2935     <rim:LocalizedString
2936 value="label.FindImmediateSuperClasses"/>
2937   </rim:Name>
2938   <rim:Description>
2939     <rim:LocalizedString
2940 value="label.FindImmediateSuperClasses.desc"/>
2941   </rim:Description>
2942   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2943 regrep:QueryLanguage:SQL-92">
2944     SELECT C2.*
2945     FROM ClassificationNode C2, Association A, Name_ N,
2946 ClassificationNode C1
2947     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2948 regrep:profile:webontology:AssociationType:OWL:SubClassOf' AND
2949     C1.id = N.parent AND
2950     N.value LIKE '$className' AND
2951     A.sourceObject = C1.id AND
2952     A.targetObject = C2.id
2953   </rim:QueryExpression>
2954 </rim:AdhocQuery>
2955

```

2956 **The Adhoc Query retrieving immediate super classes of a given classification node**

2957 7.3.3 Immediate SubClass Discovery Query

```

2958 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2959 regrep:profile:webontology:query:FindImmediateSubClasses"
2960 id="urn:oasis:names:tc:ebxml-
2961 regrep:profile:webontology:query:FindImmediateSubClasses">
2962   <rim:Name>

```

```

2963         <rim:LocalizedString value="label.FindImmediateSubClasses"/>
2964     </rim:Name>
2965     <rim:Description>
2966         <rim:LocalizedString
2967 value="label.FindImmediateSubClasses.desc"/>
2968     </rim:Description>
2969     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2970 regrep:QueryLanguage:SQL-92">
2971         SELECT C2.*
2972         FROM ClassificationNode C2, Association A, Name_ N,
2973 ClassificationNode C1
2974         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2975 regrep:profile:webontology:AssociationType:OWL:SubClassOf' AND
2976         C1.id = N.parent AND
2977         N.value LIKE '$className' AND
2978         A.sourceObject = C2.id AND
2979         A.targetObject = C1.id
2980     </rim:QueryExpression>
2981 </rim:AdhocQuery>

```

2982 **The Adhoc Query retrieving immediate subclasses of a given classification node**

2983 7.3.4 All SuperClasses Discovery Query

2984 Recursion is not supported by SQL-92, for this reason the stored procedure for this query coded in SQL
2985 99 Standard is available from:

2986 <http://www.srdc.metu.edu.tr/ebxml/ebXMLRegistryProfileForOWL/StoredProceduresSupportingebXMLRegistryProfileforOWL.htm>.

2988

2989 7.3.5 All SubClasses Discovery Query

2990 Recursion is not supported by SQL-92, for this reason the stored procedure for this query coded in SQL
2991 99 Standard is available from:

2992 <http://www.srdc.metu.edu.tr/ebxml/ebXMLRegistryProfileForOWL/StoredProceduresSupportingebXMLRegistryProfileforOWL.htm>.

2993

2994 7.3.6 EquivalentClasses Discovery Query

```

2995 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2996 regrep:profile:webontology:query:FindEquivalentClasses"
2997 id="urn:oasis:names:tc:ebxml-
2998 regrep:profile:webontology:query:FindEquivalentClasses">
2999     <rim:Name>
3000         <rim:LocalizedString value="label.FindEquivalentClasses"/>
3001     </rim:Name>
3002     <rim:Description>
3003         <rim:LocalizedString
3004 value="label.FindEquivalentClasses.desc"/>
3005     </rim:Description>
3006     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3007 regrep:QueryLanguage:SQL-92">
3008         SELECT C2.*
3009         FROM ClassificationNode C2, Association A, Name_ N,
3010 ClassificationNode C
3011         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3012 regrep:profile:webontology:AssociationType:OWL:EquivalentTo' AND
3013         C.id = N.parent AND
3014         N.value LIKE '$className' AND
3015         A.sourceObject = C.id AND
3016         A.targetObject = C2.id
3017     </rim:QueryExpression>
3018 </rim:AdhocQuery>

```

3019 **Adhoc Query retrieving all the equivalent classes of a given classification node**

3020 7.3.7 EquivalentProperties Discovery Query

```
3021 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3022 regrep:profile:webontology:query:FindEquivalentProperties"
3023 id="urn:oasis:names:tc:ebxml-
3024 regrep:profile:webontology:query:FindEquivalentProperties">
3025   <rim:Name>
3026     <rim:LocalizedString
3027 value="label.FindEquivalentProperties"/>
3028   </rim:Name>
3029   <rim:Description>
3030     <rim:LocalizedString
3031 value="label.FindEquivalentProperties.desc"/>
3032   </rim:Description>
3033   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3034 regrep:QueryLanguage:SQL-92">
3035     SELECT A3.*
3036     FROM Association A3, Association A1, Name_ N, Association
3037 A2
3038     WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
3039 regrep:profile:webontology:AssociationType:OWL:EquivalentTo' AND
3040     A2.id = N.parent AND
3041     N.value LIKE '$propertyName' AND
3042     A1.sourceObject = A2.id AND
3043     A1.targetObject = A3.id
3044   </rim:QueryExpression>
3045 </rim:AdhocQuery>
```

3046 **Adhoc Query retrieving all the equivalent Association Type of a given Association Type**

3047 7.3.8 SameExtrinsicObjects Discovery Query

```
3048 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3049 regrep:profile:webontology:query:FindTheSameExtrinsicObjects"
3050 id="urn:oasis:names:tc:ebxml-
3051 regrep:profile:webontology:query:FindTheSameExtrinsicObjects">
3052   <rim:Name>
3053     <rim:LocalizedString
3054 value="label.FindTheSameExtrinsicObjects"/>
3055   </rim:Name>
3056   <rim:Description>
3057     <rim:LocalizedString
3058 value="label.FindTheSameExtrinsicObjects.desc"/>
3059   </rim:Description>
3060   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3061 regrep:QueryLanguage:SQL-92">
3062     SELECT E2.*
3063     FROM ExtrinsicObject E2, Association A, Name_ N,
3064 ExtrinsicObject E
3065     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3066 regrep:profile:webontology:AssociationType:OWL:SameAs' AND
3067     E.id = N.parent AND
3068     N.value LIKE '$extrinsicObjectName' AND
3069     A.sourceObject = E.id AND
3070     A.targetObject = E2.id
3071   </rim:QueryExpression>
3072 </rim:AdhocQuery>
```

3073 **Adhoc Query retrieving all the "ExtrinsicObjects" defined to be the same with a given**
3074 **ExtrinsicObject**

3075 7.3.9 DifferentExtrinsicObjects Discovery Query

```
3076 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3077 regrep:profile:webontology:query:FindDifferentExtrinsicObjects"
3078 id="urn:oasis:names:tc:ebxml-
3079 regrep:profile:webontology:query:FindDifferentExtrinsicObjects">
3080   <rim:Name>
3081     <rim:LocalizedString
3082 value="label.FindDifferentExtrinsicObjects"/>
3083   </rim:Name>
3084   <rim:Description>
3085     <rim:LocalizedString
3086 value="label.FindDifferentExtrinsicObjects.desc"/>
3087   </rim:Description>
3088   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3089 regrep:QueryLanguage:SQL-92">
3090     SELECT E2.*
3091     FROM ExtrinsicObject E2, Association A, Name_ N,
3092 ExtrinsicObject E
3093     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3094 regrep:profile:webontology:AssociationType:OWL:DifferentFrom' AND
3095     E.id = N.parent AND
3096     N.value LIKE '$extrinsicObjectName' AND
3097     A.sourceObject = E.id AND
3098     A.targetObject = E2.id
3099   </rim:QueryExpression>
3100 </rim:AdhocQuery>
```

3101 **Adhoc Query retrieving all the "ExtrinsicObjects" defined to be different from a given**
3102 **ExtrinsicObject**

3103 7.3.10 AllDifferentRegistryObject Discovery Query

```
3104 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3105 regrep:profile:webontology:query:FindAllDifferent"
3106 id="urn:oasis:names:tc:ebxml-
3107 regrep:profile:webontology:query:FindAllDifferent">
3108   <rim:Name>
3109     <rim:LocalizedString value="label.FindAllDifferent"/>
3110   </rim:Name>
3111   <rim:Description>
3112     <rim:LocalizedString value="label.FindAllDifferent.desc"/>
3113   </rim:Description>
3114   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3115 regrep:QueryLanguage:SQL-92">
3116     SELECT RO2.*
3117     FROM RegistryObject RO2, Association A1, Association A2,
3118 Name_ N, RegistryObject RO,
3119 RegistryPackage RP<!--, Slot S-->
3120     WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
3121 regrep:profile:webontology:AssociationType:OWL:HasMember' AND
3122     RO.id = N.parent AND
3123     N.value LIKE '$registryObjectName' AND
3124     A1.sourceObject = RP.id AND
3125     <!-- S.parent = RP.id AND
3126     S.name_ LIKE 'packageType' AND S.value LIKE
3127 'allDifferent' AND -->
3128     A1.targetObject = RO.id AND
3129     A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3130 regrep:profile:webontology:AssociationType:OWL:HasMember' AND
3131     A2.sourceObject = RP.id AND
3132     A2.targetObject != RO.id AND
3133     A2.targetObject = RO2.id
3134   </rim:QueryExpression>
3135 </rim:AdhocQuery>
```

3136 **Adhoc Query retrieving all the "RegistryObjects" defined to be different from a given**

3137 RegistryObject through a “allDifferent” construct

3138 7.3.11 ObjectProperties Discovery Query

```
3139 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3140 regrep:profile:webontology:query:FindObjectProperties"
3141 id="urn:oasis:names:tc:ebxml-
3142 regrep:profile:webontology:query:FindObjectProperties">
3143   <rim:Name>
3144     <rim:LocalizedString value="label.FindObjectProperties"/>
3145   </rim:Name>
3146   <rim:Description>
3147     <rim:LocalizedString
3148 value="label.FindObjectProperties.desc"/>
3149   </rim:Description>
3150   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3151 regrep:QueryLanguage:SQL-92">
3152     SELECT A.*
3153     FROM Association A, Name_N, ClassificationNode C
3154     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3155 regrep:profile:webontology:AssociationType:OWL:ObjectProperty' AND
3156     C.id = N.parent AND
3157     N.value LIKE '$className' AND
3158     A.sourceObject = C.id
3159   </rim:QueryExpression>
3160 </rim:AdhocQuery>
```

3161 **Adhoc Query retrieving all the object properties of a given classification node**

3162 7.3.12 ImmediateInheritedObjectProperties Discovery Query

```
3163 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3164 regrep:profile:webontology:query:FindImmediateInheritedObjectProperties"
3165 id="urn:oasis:names:tc:ebxml-
3166 regrep:profile:webontology:query:FindImmediateInheritedObjectProperties">
3167   <rim:Name>
3168     <rim:LocalizedString
3169 value="label.FindImmediateInheritedObjectProperties"/>
3170   </rim:Name>
3171   <rim:Description>
3172     <rim:LocalizedString
3173 value="label.FindImmediateInheritedObjectProperties.desc"/>
3174   </rim:Description>
3175   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3176 regrep:QueryLanguage:SQL-92">
3177     SELECT A2.*
3178     FROM Association A, Name_N, ClassificationNode C1,
3179 ClassificationNode C2, Association A2
3180     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3181 regrep:profile:webontology:AssociationType:OWL:SubClassOf' AND
3182     C1.id = N.parent AND
3183     N.value LIKE '$className' AND
3184     A.sourceObject = C1.id AND
3185     A.targetObject = C2.id AND
3186     A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3187 regrep:profile:webontology:AssociationType:OWL:ObjectProperty' AND
3188     A2.sourceObject=C2.id
3189   </rim:QueryExpression>
3190 </rim:AdhocQuery>
```

3191 **Adhoc Query retrieving all of the properties of a given classification node including the ones**
3192 **inherited from its immediate super classes**

3193 7.3.13 AllInheritedObjectProperties Discovery Query

3194 Recursion is not supported by SQL-92, for this reason the stored procedure for this query coded in SQL

3195 99 Standard is available from:
3196 <http://www.srdc.metu.edu.tr/ebxml/ebXMLRegistryProfileForOWL/StoredProceduresSupportingebXMLRegistryProfileforOWL.htm>
3197

3198 7.3.14 DatatypeProperties Discovery Query

```
3199 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-  
3200 regrep:profile:webontology:query:FindDatatypeProperties"  
3201 id="urn:oasis:names:tc:ebxml-  
3202 regrep:profile:webontology:query:FindDatatypeProperties">  
3203   <rim:Name>  
3204     <rim:LocalizedString value="label.FindDatatypeProperties"/>  
3205   </rim:Name>  
3206   <rim:Description>  
3207     <rim:LocalizedString  
3208 value="label.FindDatatypeProperties.desc"/>  
3209   </rim:Description>  
3210   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
3211 regrep:QueryLanguage:SQL-92">  
3212     SELECT A.*  
3213     FROM Association A, Name_N, ClassificationNode C  
3214     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-  
3215 regrep:profile:webontology:AssociationType:OWL:DatatypeProperty' AND  
3216     C.id = N.parent AND  
3217     N.value LIKE '$className' AND  
3218     A.sourceObject = C.id  
3219   </rim:QueryExpression>  
3220 </rim:AdhocQuery>
```

3221 **Adhoc Query retrieving all the datatype properties of a given classification node**

3222 7.3.15 AllInheritedDatatypeProperties Discovery Query

3223 Recursion is not supported by SQL-92, for this reason the stored procedure for this query coded in SQL
3224 99 Standard is available from:

3225 <http://www.srdc.metu.edu.tr/ebxml/ebXMLRegistryProfileForOWL/StoredProceduresSupportingebXMLRegistryProfileforOWL.htm>
3226

3227 7.3.16 TransitiveRelationships Discovery Query

```
3228 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-  
3229 regrep:profile:webontology:query:FindTransitiveRelationships"  
3230 id="urn:oasis:names:tc:ebxml-  
3231 regrep:profile:webontology:query:FindTransitiveRelationships">  
3232   <rim:Name>  
3233     <rim:LocalizedString  
3234 value="label.FindTransitiveRelationships"/>  
3235   </rim:Name>  
3236   <rim:Description>  
3237     <rim:LocalizedString  
3238 value="label.FindTransitiveRelationships.desc"/>  
3239   </rim:Description>  
3240   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
3241 regrep:QueryLanguage:SQL-92">  
3242     SELECT C.*  
3243     FROM ClassificationNode C, Association A1, Association A2,  
3244     Name_N1, Name_N2, Name_N3  
3245     WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-  
3246 regrep:profile:webontology:AssociationType:OWL:TransitiveProperty' AND  
3247     A1.id = N1.parent AND  
3248     N1.value LIKE '$propertyName' AND  
3249     A1.sourceObject = N3.parent AND  
3250     N3.value LIKE '$className' AND  
3251     A2.sourceObject = A1.targetObject AND
```

```

3252         A2.id = N2.parent AND
3253         N2.value LIKE '$propertyName' AND
3254         A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3255 regrep:profile:webontology:AssociationType:OWL:TransitiveProperty' AND
3256         A2.targetObject = C.id
3257         <!-- UNION
3258         SELECT C.*
3259         FROM ClassificationNode C, Association A1, Name_ N1, Name_
3260 N3
3261         WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
3262 regrep:profile:webontology:AssociationType:OWL:TransitiveProperty' AND
3263         A1.id = N1.parent AND
3264         N1.value LIKE '$propertyName' AND
3265         A1.sourceObject = N3.parent AND
3266         N3.value LIKE '$className' AND
3267         A1.targetObject = C.id -->
3268     </rim:QueryExpression>
3269 </rim:AdhocQuery>

```

3270 **Adhoc Query retrieving the objects in transitive relationship with a given object**

3271 7.3.17 TargetObjects Discovery Query

```

3272 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3273 regrep:profile:webontology:query:FindTargetObjects"
3274 id="urn:oasis:names:tc:ebxml-
3275 regrep:profile:webontology:query:FindTargetObjects">
3276     <rim:Name>
3277         <rim:LocalizedString value="label.FindTargetObjects"/>
3278     </rim:Name>
3279     <rim:Description>
3280         <rim:LocalizedString value="label.FindTargetObjects.desc"/>
3281     </rim:Description>
3282     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3283 regrep:QueryLanguage:SQL-92">
3284         SELECT C2.*
3285         FROM ClassificationNode C2, Association A, Name_ N, Name_
3286 N2, ClassificationNode C1
3287         WHERE A.id=N2.parent AND
3288         N2.value LIKE '$propertyName' AND
3289         C1.id = N.parent AND
3290         N.value LIKE '$className' AND
3291         A.sourceObject = C1.id AND
3292         A.targetObject = C2.id
3293     </rim:QueryExpression>
3294 </rim:AdhocQuery>

```

3295 **Adhoc Query retrieving the Target Objects from the Registry, given a Source Object and an**
3296 **Association**

3297 7.3.18 TargetObjectsInverseOf Discovery Query

```

3298 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3299 regrep:profile:webontology:query:FindTOinverseOf"
3300 id="urn:oasis:names:tc:ebxml-
3301 regrep:profile:webontology:query:FindTOinverseOf">
3302     <rim:Name>
3303         <rim:LocalizedString value="label.FindTOinverseOf"/>
3304     </rim:Name>
3305     <rim:Description>
3306         <rim:LocalizedString value="label.FindTOinverseOf.desc"/>
3307     </rim:Description>
3308     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3309 regrep:QueryLanguage:SQL-92">
3310         SELECT C2.*
3311         FROM ClassificationNode C2, Association A1, Association A2,
3312 Association A3, Name_ N, Name_ N2, ClassificationNode C1

```

```

3313         WHERE A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3314 regrep:profile:webontology:AssociationType:OWL:InverseOf' AND
3315         A1.id = N.parent AND
3316         N.value LIKE '$propertyName' AND
3317         A2.sourceObject = A1.id AND
3318         A2.targetObject = A3.id AND
3319         C1.id = N2.parent AND
3320         N2.value LIKE '$className' AND
3321         A3.targetObject = C1.id AND
3322         A3.sourceObject = C2.id
3323     </rim:QueryExpression>
3324 </rim:AdhocQuery>

```

3325 **Adhoc query retrieving the Source Objects of an Association which is in "inverseOf"**
3326 **relationship to this Association**

3327 7.3.19 InverseRanges Discovery Query

```

3328 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3329 regrep:profile:webontology:query:FindInverseRanges"
3330 id="urn:oasis:names:tc:ebxml-
3331 regrep:profile:webontology:query:FindInverseRanges">
3332     <rim:Name>
3333         <rim:LocalizedString value="label.FindInverseRanges"/>
3334     </rim:Name>
3335     <rim:Description>
3336         <rim:LocalizedString value="label.FindInverseRanges.desc"/>
3337     </rim:Description>
3338     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3339 regrep:QueryLanguage:SQL-92">
3340         <!-- SELECT C2.*
3341         FROM Association A, Name_ N, Name_ N2, ClassificationNode
3342 C1, ClassificationNode C2
3343         WHERE A.id=N2.parent AND
3344         N2.value LIKE '$propertyName' AND
3345         C1.id = N.parent AND
3346         N.value LIKE '$className' AND
3347         A.sourceObject = C1.id AND
3348         A.targetObject = C2.id
3349         UNION -->
3350         SELECT C2.*
3351         FROM ClassificationNode C2, Association A1, Association A2,
3352 Association A3, Name_ N, NAME_ N2, ClassificationNode C1
3353         WHERE A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3354 regrep:profile:webontology:AssociationType:OWL:InverseOf' AND
3355         A1.id = N.parent AND
3356         N.value LIKE '$propertyName' AND
3357         A2.sourceObject = A1.id AND
3358         A2.targetObject = A3.id AND
3359         C1.id = N2.parent AND
3360         N2.value LIKE '$className' AND
3361         A1.sourceObject = C1.id AND
3362         A3.sourceObject = C2.id
3363     </rim:QueryExpression>
3364 </rim:AdhocQuery>

```

3365 **Adhoc Query Retrieving both the Target Objects of a given Association and the Source**
3366 **Objects of an Association which is in "inverseOf" relationship to this Association**

3367 7.3.20 SymmetricProperties Discovery Query

```

3368 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3369 regrep:profile:webontology:query:FindSymmetricProperties"
3370 id="urn:oasis:names:tc:ebxml-
3371 regrep:profile:webontology:query:FindSymmetricProperties">
3372     <rim:Name>
3373         <rim:LocalizedString value="label.FindSymmetricProperties"/>

```

```

3374     </rim:Name>
3375     <rim:Description>
3376         <rim:LocalizedString
3377 value="label.FindSymmetricProperties.desc"/>
3378     </rim:Description>
3379     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3380 regrep:QueryLanguage:SQL-92">
3381         SELECT A.*
3382         FROM Association A, Name_N, ClassificationNode C
3383         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3384 regrep:profile:webontology:AssociationType:OWL:SymmetricProperty' AND
3385         C.id = N.parent AND
3386         N.value LIKE '$className' AND
3387         A.sourceObject = C.id
3388     </rim:QueryExpression>
3389 </rim:AdhocQuery>

```

3390 **Adhoc Query retrieving all the Symmetric properties of a given classification node**

3391 7.3.21 FunctionalProperties Discovery Query

```

3392 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3393 regrep:profile:webontology:query:FindFunctionalProperties"
3394 id="urn:oasis:names:tc:ebxml-
3395 regrep:profile:webontology:query:FindFunctionalProperties">
3396     <rim:Name>
3397         <rim:LocalizedString
3398 value="label.FindFunctionalProperties"/>
3399     </rim:Name>
3400     <rim:Description>
3401         <rim:LocalizedString
3402 value="label.FindFunctionalProperties.desc"/>
3403     </rim:Description>
3404     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3405 regrep:QueryLanguage:SQL-92">
3406         SELECT A.*
3407         FROM Association A, Name_N, ClassificationNode C
3408         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3409 regrep:profile:webontology:AssociationType:OWL:FunctionalProperty' AND
3410         C.id = N.parent AND
3411         N.value LIKE '$className' AND
3412         A.sourceObject = C.id
3413     </rim:QueryExpression>
3414 </rim:AdhocQuery>

```

3415 **Adhoc Query retrieving all the Functional properties of a given classification node**

3416 7.3.22 InverseFunctionalProperties Discovery Query

```

3417 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3418 regrep:profile:webontology:query:FindInverseFunctionalProperties"
3419 id="urn:oasis:names:tc:ebxml-
3420 regrep:profile:webontology:query:FindInverseFunctionalProperties">
3421     <rim:Name>
3422         <rim:LocalizedString
3423 value="label.FindInverseFunctionalProperties"/>
3424     </rim:Name>
3425     <rim:Description>
3426         <rim:LocalizedString
3427 value="label.FindInverseFunctionalProperties.desc"/>
3428     </rim:Description>
3429     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3430 regrep:QueryLanguage:SQL-92">
3431         SELECT A.*
3432         FROM Association A, Name_N, ClassificationNode C

```

```

3433         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3434 regrep:profile:webontology:AssociationType:OWL:InverseFunctionalProperty'
3435         ' AND
3436             C.id = N.parent AND
3437             N.value LIKE '$className' AND
3438             A.sourceObject = C.id
3439         </rim:QueryExpression>
3440 </rim:AdhocQuery>

```

3441 **Adhoc Query retrieving all the Inverse Functional properties of a given classification node**

3442 7.3.23 Instances Discovery Query Discovery Query

```

3443 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3444 regrep:profile:webontology:query:FindInstances"
3445 id="urn:oasis:names:tc:ebxml-
3446 regrep:profile:webontology:query:FindInstances">
3447   <rim:Name>
3448     <rim:LocalizedString value="label.FindInstances"/>
3449   </rim:Name>
3450   <rim:Description>
3451     <rim:LocalizedString value="label.FindInstances.desc"/>
3452   </rim:Description>
3453   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3454 regrep:QueryLanguage:SQL-92">
3455     <!-- SELECT S.* FROM Service S, (
3456       SELECT S1.value AS id
3457       FROM Slot S1, Name_ N, ClassificationNode C
3458       WHERE S1.parent = C.id AND
3459
3460
3461       C.id = N.parent AND
3462       N.value LIKE '$className' AND
3463       S1.name_ LIKE 'urn:oasis:names:tc:ebxml-
3464 regrep:profile:webontology:slot:intersectionOf '
3465     )
3466     AS T1, (
3467       SELECT S1.value AS id
3468       FROM Slot S1, Name_ N, ClassificationNode C
3469       WHERE S1.parent = C.id AND
3470       C.id = N.parent AND
3471       N.value LIKE '$className' AND
3472       S1.name_ LIKE 'urn:oasis:names:tc:ebxml-
3473 regrep:profile:webontology:slot:intersectionOf '
3474     ) AS T2
3475     WHERE S.id IN (
3476       SELECT classifiedObject
3477       FROM Classification
3478       WHERE classificationNode=T1.id
3479     INTERSECT
3480       SELECT classifiedObject
3481       FROM Classification
3482       WHERE classificationNode=T2.id
3483     ) AND T1.id!=T2.id
3484     UNION -->
3485     SELECT S.*
3486     FROM Service S, Classification C, ClassificationNode CN,
3487     Name_ N
3488     WHERE S.id = C. classifiedObject AND
3489     C.classificationNode = CN.id AND
3490     N.value LIKE '$className' AND
3491     N.parent = CN.id
3492   </rim:QueryExpression>
3493 </rim:AdhocQuery>

```

3494 **Adhoc Query Retrieving the instances of intersected classes**

3495 8 OWL Profile References

3496 8.1 Normative References

- 3497 [Bechhofer, Harmelen, Hendler, Horrocks, McGuinness, Patel-Schneider, Stein]
3498 Bechhofer, S., Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., Stein, L.
3499 A., OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004
3500 <http://www.w3.org/TR/2004/REC-owl-ref-20040210/>
3501
3502 [Brickley, Guha] Brickley, D., Guha, R.V., RDF Vocabulary Description Language 1.0: RDF Schema
3503 W3C Recommendation 10 February 2004
3504 <http://www.w3.org/TR/rdf-schema/>
3505
3506 [DAML+OIL] <http://www.daml.org/>
3507 [ebRIM] ebXML Registry Information Model version 3.0
3508 <http://docs.oasis-open.org/regrep/regrep-rim/v3.0/regrep-rim-3.0-os.pdf>
3509
3510 [ebRS] ebXML Registry Services Specification version 3.0
3511 <http://docs.oasis-open.org/regrep/regrep-rs/v3.0/regrep-rs-3.0-os.pdf>
3512 [ebRR-DPT] Deployment Profile Template For ebXML Registry 3.0 OASIS Specifications V_0.1.2
3513 [ebMS-DPT] Deployment Profile Template For OASIS Specification ebXML Message Service 2.0
3514 [McGuinness, Harmelen] McGuinness, D. L., Harmelen, F., OWL Web Ontology Language Overview,
3515 W3C Recommendation 10 February 2004, <http://www.w3.org/TR/owl-features/>
3516 [OWL] Web Ontology Language (OWL), <http://www.w3.org/2004/OWL/>
3517 [RDF] Resource Description Framework, <http://www.w3.org/TR/rdf-concepts/>
3518 [RDFS] RDF Vocabulary Description Language 1.0: RDF Schema <http://www.w3.org/TR/rdf-schema/>
3519 [Smith, Welty, McGuinness] Smith, M. K., Welty, C., McGuinness, D. L.,
3520 OWL Web Ontology Language Guide, W3C Recommendation 10 February 2004,
3521 <http://www.w3.org/TR/owl-guide/>
3522 [SQL 92] SQL ISO/IEC 9075:1992 Information technology - Database languages - SQL.
3523 [SQL 99] ISO/IEC 9075:1999(E) Information technology - Database languages – SQL.
3524 [UML] Unified Modeling Language version 1.5
3525 <http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.pdf>
3526 [WSDL] WSDL Specification
3527 <http://www.w3.org/TR/wsdl>

3528 **8.2 Informative References**

3529 [Dogac, et. al. 2005] Dogac A., Kabak Y., Laleci G. C. Mattocks, F. Najmi, J. Pollock
3530 Enhancing ebXML Registries to Make them OWL Aware
3531 Distributed and Parallel Databases Journal, Springer-Verlag, Vol. 18, No. 1, July 2005, pp. 9-36.

3532
3533 [Dogac et. al. 2006] Dogac A., Laleci G., Kabak Y., Unal S., Beale T., Heard S., Elkin P., Najmi F.,
3534 Mattocks C., Webber D., Kernberg M.
3535 Exploiting ebXML Registry Semantic Constructs for Handling Archetype Metadata in Healthcare
3536 Informatics
3537 International Journal of Metadata, Semantics and Ontologies, Volume 1, No. 1, 2006.

3538
3539 [IMPL] ebXML Registry 3.0 Implementations
3540 freebXML Registry: A royalty free, open source ebXML Registry Implementation
3541 <http://ebxmlrr.sourceforge.net>

3542
3543 [LeeHendler]
3544 Berners-Lee, T., Hendler, J., Lassila, O., "The Semantic Web", Scientific American, May 2001.

3545
3546 [StaabStuder] Staab, S., Studer, R., Handbook on Ontologies, Springer, 2004.

3547

3548 **Appendix A**

3549 **Contributors:**

Name	Affiliation
Farrukh Najmi	Sun Micro Systems
Carl Mattocks	MetLife
Jeff Pollock	Network Inference
Evan Wallace	NIST
Dave RR Webber	Individual
Nikola Stojanovic	GS1 US
Ivan Bedini	France Telecom
Yildiray Kabak	-
Gokce Banu Laleci	-

3550