



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

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:

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director. OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.

Copyright (c) OASIS Open 2006. All Rights Reserved.

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 paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document 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 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

1 Table of Contents

75		
76	1 Table of Contents.....	3
77	1 Introduction.....	10
78	1.1 Terminology.....	11
79	1.2 Conventions.....	11
80	1.3 Recommended Enhancements.....	11
81	2 OWL Overview.....	12
82	2.1 Semantic Web Languages upon which OWL is Layered.....	12
83	2.2 OWL Lite Constructs.....	13
84	2.2.1 RDF Schema Features.....	13
85	2.2.2 (In)Equality.....	13
86	2.2.3 Property Characteristics	13
87	2.2.4 Property Restrictions.....	13
88	2.2.5 Restricted Cardinality.....	13
89	2.2.6 Class Intersection.....	13
90	2.2.7 Versioning.....	14
91	2.2.8 Annotation Properties	14
92	2.2.9 Datatypes	14
93	2.3 OWL DL Constructs.....	14
94	2.3.1 Class Axioms.....	14
95	2.3.2 Boolean Combinations of Class Expressions	14
96	2.3.3 Arbitrary Cardinality	14
97	2.3.4 Filler Information.....	14
98	3 ebXML Registry Overview.....	15
99	3.1 Overview of [ebRIM].....	15
100	3.1.1 RegistryObject.....	16
101	3.1.2 Object Identification.....	16
102	3.1.3 Object Naming and Description.....	17
103	3.1.4 Object Attributes.....	17
104	3.1.4.1 Slot Attributes.....	17
105	3.1.5 Object Classification.....	18
106	3.1.6 Object Association.....	18
107	3.1.7 Object References To Web Content.....	19
108	3.1.8 Object Packaging.....	19
109	3.1.9 ExtrinsicObject	20
110	3.1.10 Service Description.....	20
111	3.2 Overview of [ebRS].....	20
112	4 Representing OWL Lite Constructs in ebRIM	21
113	4.1 Representing RDF Schema Features in ebRIM.....	21
114	4.1.1 owl:Class → rim:ClassificationNode.....	21
115	4.1.2 rdf:Property → rim:Association Type HasProperty.....	21
116	4.1.3 rdfs:subPropertyOf → rim:Association Type SubPropertyOf.....	22
117	4.1.4 rdfs:subClassOf → rim:Association Type SubClassOf.....	22
118	4.1.5 owl:Individual → rim:ExtrinsicObject.....	23
119	4.2 Representing OWL (In)Equality Constructs in ebXML RIM.....	24

120	4.2.1 owl:equivalentClass, owl:equivalentProperty → rim:Association Type EquivalentTo	24
121	4.2.2 owl:sameAs → rim:Association Type SameAs.....	24
122	4.2.3 owl:differentFrom → rim:Association Type DifferentFrom.....	24
123	4.2.4 owl:AllDifferent.....	25
124	4.3 Representing OWL Property Characteristics in ebRIM.....	26
125	4.3.1 owl:ObjectProperty → rim:Association Type objectProperty.....	26
126	4.3.2 owl:DatatypeProperty → rim:Association Type DatatypeProperty.....	26
127	4.3.3 owl:TransitiveProperty → rim:Association Type TransitiveProperty.....	26
128	4.3.4 owl:inverseOf → rim:Association Type InverseOf.....	27
129	4.3.5 owl:SymmetricProperty→ rim:Association Type SymmetricProperty.....	28
130	4.3.6 owl:FunctionalProperty→ rim:Association Type FunctionalProperty.....	28
131	4.3.7 owl:InverseFunctionalProperty→ rim:Association Type InverseFunctionalProperty.....	29
132	4.4 OWL Property Restrictions in ebXML RIM.....	29
133	4.5 Representing OWL Restricted Cardinality in ebXML RIM.....	30
134	4.5.1 owl:minCardinality (only 0 or 1).....	30
135	4.5.2 owl:maxCardinality (only 0 or 1).....	31
136	4.5.3 owl:cardinality (only 0 or 1).....	32
137	4.6 Representing OWL Class Intersection in ebXML RIM.....	32
138	4.7 Representing OWL Versioning in ebXML RIM.....	33
139	4.7.1 owl:versionInfo, owl:priorVersion.....	33
140	4.8 Representing OWL Annotation Properties in ebXML RIM.....	34
141	4.8.1 rdfs:label.....	34
142	4.8.2 rdfs:comment.....	34
143	4.8.3 rdfs:seeAlso.....	34
144	4.9 OWL Datatypes in ebXML RIM.....	35
145	5 Cataloging Service Profile.....	36
146	5.1 Invocation Control File.....	36
147	5.2 Input Metadata.....	36
148	5.3 Input Content.....	36
149	5.4 Output Metadata.....	37
150	5.4.1 owl:Class → rim:ClassificationNode.....	37
151	5.4.2 rdf:Property → rim:Association Type HasProperty.....	37
152	5.4.3 rdfs:subPropertyOf → rim:Association Type SubPropertyOf.....	37
153	5.4.4 rdfs:subClassOf → rim:Association Type subClassOf.....	37
154	5.4.5 owl:Individual → rim:ExtrinsicObject.....	37
155	5.4.6 owl:equivalentClass, owl:equivalentProperty → rim:Association Type EquivalentTo	37
156	5.4.7 owl:sameAs → rim:Association Type SameAs	37
157	5.4.8 owl:differentFrom → rim:Association Type DifferentFrom.....	37
158	5.4.9 owl:AllDifferent → rim:RegistryPackage.....	37
159	5.4.10 owl:ObjectProperty → rim:Association Type ObjectProperty.....	38
160	5.4.11 owl:DatatypeProperty → rim:Association Type DatatypeProperty.....	38
161	5.4.12 owl:TransitiveProperty → rim:Association Type TransitiveProperty.....	38
162	5.4.13 owl:inverseOf → rim:Association Type InverseOf.....	38
163	5.4.14 owl:SymmetricProperty→ rim:Association Type SymetricProperty.....	38
164	5.4.15 owl:FunctionalProperty→ rim:Association Type FunctionalProperty.....	38
165	5.4.16 owl:InverseFunctionalProperty→ rim:Association Type InverseFunctionalProperty.....	38

166	5.4.17 owl:minCardinality (only 0 or 1).....	38
167	5.4.18 owl:maxCardinality (only 0 or 1).....	39
168	5.4.19 owl:cardinality.....	39
169	5.4.20 owl:intersectionOf.....	39
170	5.4.21 rdfs:label.....	39
171	5.4.22 rdfs:comment.....	39
172	5.4.23 rdfs:seeAlso.....	39
173	6 Discovery Profile.....	40
174	6.1 All SuperProperties Discovery Query.....	40
175	6.1.1 Parameter \$propertyName.....	40
176	6.1.2 Example of All SuperProperties Discovery Query.....	40
177	6.2 Immediate SuperClass Discovery Query.....	41
178	6.2.1 Parameter \$className.....	41
179	6.2.2 Example of Immediate SuperClass Discovery Query.....	41
180	6.3 Immediate SubClass Discovery Query.....	42
181	6.3.1 Parameter \$className.....	42
182	6.3.2 Example of Immediate SubClasses Discovery Query.....	42
183	6.4 All SuperClasses Discovery Query.....	42
184	6.4.1 Parameter \$className.....	43
185	6.4.2 Example of All SuperClasses Discovery Query.....	43
186	6.5 All SubClasses Discovery Query.....	43
187	6.5.1 Parameter \$className.....	43
188	6.5.2 Example of All SubClasses Discovery Query.....	43
189	6.6 EquivalentClasses Discovery Query.....	44
190	6.6.1 Parameter \$className.....	44
191	6.6.2 Example of EquivalentClasses Discovery Query.....	44
192	6.7 EquivalentProperties Discovery Query.....	45
193	6.7.1 Parameter \$propertyName.....	45
194	6.7.2 Example of EquivalentProperties Discovery Query.....	45
195	6.8 SameExtrinsicObjects Discovery Query.....	46
196	6.8.1 Parameter \$extrinsicObjectName.....	46
197	6.8.2 Example of SameExtrinsicObjects Discovery Query.....	46
198	6.9 DifferentExtrinsicObjects Discovery Query.....	46
199	6.9.1 Parameter \$extrinsicObjectName.....	47
200	6.9.2 Example of DifferentExtrinsicObjects Discovery Query.....	47
201	6.10 AllDifferentRegistryObject Discovery Query.....	47
202	6.10.1 Parameter \$registryObjectName.....	47
203	6.10.2 Example of AllDifferentRegistryObjects Discovery Query.....	47
204	6.11 ObjectProperties Discovery Query.....	48
205	6.11.1 Parameter \$className.....	48
206	6.11.2 Example of ObjectProperties Discovery Query.....	48
207	6.12 ImmediateInheritedObjectProperties Discovery Query.....	49
208	6.12.1 Parameter \$className.....	49
209	6.12.2 Example of ImmediateInheritedObjectProperties Discovery Query.....	49
210	6.13 AllInheritedObjectProperties Discovery Query.....	50
211	6.13.1 Parameter \$className.....	50

212	6.13.2 Example of AllInheritedObjectProperties Discovery Query.....	50
213	6.14 DatatypeProperties Discovery Query.....	51
214	6.14.1 Parameter \$className.....	51
215	6.14.2 Example of DatatypeProperties Discovery Query.....	51
216	6.15 AllInheritedDatatypeProperties Discovery Query.....	51
217	6.15.1 Parameter \$className.....	52
218	6.15.2 Example of AllInheritedDatatypeProperties Discovery Query.....	52
219	6.16 TransitiveRelationships Discovery Query.....	52
220	6.16.1 Parameter \$className.....	53
221	6.16.2 Parameter \$propertyName.....	53
222	6.16.3 Example of TransitiveRelationships Discovery Query.....	53
223	6.17 TargetObjects Discovery Query.....	53
224	6.17.1 Parameter \$className.....	54
225	6.17.2 Parameter \$propertyName.....	54
226	6.17.3 Example of TargetObjects Discovery Query.....	54
227	6.18 TargetObjectsInverseOf Discovery Query.....	54
228	6.18.1 Parameter \$className.....	55
229	6.18.2 Parameter \$propertyName.....	55
230	6.18.3 Example of TargetObjectsInverseOf Discovery Query.....	55
231	6.19 InverseRanges Discovery Query.....	55
232	6.19.1 Parameter \$className.....	56
233	6.19.2 Parameter \$propertyName.....	56
234	6.19.3 Example of InverseRanges Discovery Query.....	56
235	6.20 SymmetricProperties Discovery Query.....	57
236	6.20.1 Parameter \$className.....	57
237	6.20.2 Example of SymmetricProperties Discovery Query.....	57
238	6.21 FunctionalProperties Discovery Query.....	57
239	6.21.1 Parameter \$className.....	58
240	6.21.2 Example of FunctionalProperties Discovery Query.....	58
241	6.22 InverseFunctionalProperties Discovery Query.....	58
242	6.22.1 Parameter \$className.....	58
243	6.22.2 Example of InverseFunctionalProperties Discovery Query.....	58
244	6.23 Instances Discovery Query.....	59
245	6.23.1 Parameter \$className.....	59
246	6.23.2 Example of Instances Discovery Query.....	59
247	7 Canonical Metadata Definitions.....	61
248	7.1 ObjectType Extensions.....	61
249	7.2 AssociationType Extensions.....	61
250	7.3 Canonical Queries.....	64
251	7.3.1 All SuperProperties Discovery Query.....	64
252	7.3.2 Immediate SuperClass Discovery Query.....	64
253	7.3.3 Immediate SubClass Discovery Query.....	64
254	7.3.4 All SuperClasses Discovery Query.....	65
255	7.3.5 All SubClasses Discovery Query.....	65
256	7.3.6 EquivalentClasses Discovery Query.....	65
257	7.3.7 EquivalentProperties Discovery Query.....	66

258	7.3.8 SameExtrinsicObjects Discovery Query.....	66
259	7.3.9 DifferentExtrinsicObjects Discovery Query.....	67
260	7.3.10 AllDifferentRegistryObject Discovery Query.....	67
261	7.3.11 ObjectProperties Discovery Query.....	68
262	7.3.12 ImmediateInheritedObjectProperties Discovery Query.....	68
263	7.3.13 AllInheritedObjectProperties Discovery Query.....	69
264	7.3.14 DatatypeProperties Discovery Query.....	69
265	7.3.15 AllInheritedDatatypeProperties Discovery Query.....	69
266	7.3.16 TransitiveRelationships Discovery Query.....	69
267	7.3.17 TargetObjects Discovery Query.....	70
268	7.3.18 TargetObjectsInverseOf Discovery Query.....	70
269	7.3.19 InverseRanges Discovery Query.....	71
270	7.3.20 SymmetricProperties Discovery Query.....	72
271	7.3.21 FunctionalProperties Discovery Query.....	72
272	7.3.22 InverseFunctionalProperties Discovery Query.....	72
273	7.3.23 Instances Discovery Query Discovery Query.....	73
274	8 OWL Profile References.....	75
275	8.1 Normative References.....	75
276	8.2 Informative References.....	76
277	Appendix A.....	76
278		

Illustration Index

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

279

Index of Tables

280

1 Introduction

281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
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

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.

329 1.1 Terminology

330 The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT,
331 RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in IETF RFC
332 2119 [RFC2111].

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

336 1.2 Conventions

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

- 339 • UML Diagrams

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

- 342 • Identifier Placeholders

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

348

349

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

350 1.3 Recommended Enhancements

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

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

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

2 OWL Overview

360

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

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

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

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

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

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

2.1 Semantic Web Languages upon which OWL is Layered

379

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

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

390

391 Table 1: Semantic Web namespace table

392

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#

393

394

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

398

399

400 2.2 OWL Lite Constructs

401 2.2.1 RDF Schema Features

- 402 • Class (Thing, Nothing)
- 403 • rdfs:subClassOf
- 404 • rdf:Property
- 405 • rdfs:subPropertyOf
- 406 • rdfs:domain
- 407 • rdfs:range
- 408 • Individual

409 2.2.2 (In)Equality

- 410 • equivalentClass
- 411 • equivalentProperty
- 412 • sameAs
- 413 • differentFrom
- 414 • AllDifferent
- 415 • distinctMembers

416 2.2.3 Property Characteristics

- 417 • ObjectProperty
- 418 • DatatypeProperty
- 419 • inverseOf
- 420 • TransitiveProperty
- 421 • SymmetricProperty
- 422 • FunctionalProperty
- 423 • InverseFunctionalProperty

424 2.2.4 Property Restrictions

- 425 • Restriction
- 426 • onProperty
- 427 • allValuesFrom
- 428 • someValuesFrom

429 2.2.5 Restricted Cardinality

- 430 • minCardinality (only 0 or 1)
- 431 • maxCardinality (only 0 or 1)
- 432 • cardinality (only 0 or 1)

433 2.2.6 Class Intersection

- 434 • intersectionOf

435 2.2.7 Versioning

- 436 • versionInfo
- 437 • priorVersion
- 438 • backwardCompatibleWith
- 439 • incompatibleWith
- 440 • DeprecatedClass
- 441 • DeprecatedProperty

442 2.2.8 Annotation Properties

- 443 • rdfs:label
- 444 • rdfs:comment
- 445 • rdfs:seeAlso
- 446 • rdfs:isDefinedBy
- 447 • AnnotationProperty
- 448 • OntologyProperty

449 2.2.9 Datatypes

- 450 • xsd datatypes

451 2.3 OWL DL Constructs

452 2.3.1 Class Axioms

- 453 • oneOf, dataRange
- 454 • disjointWith
- 455 • equivalentClass (applied to class expressions)
- 456 • rdfs:subClassOf (applied to class expressions)

457 2.3.2 Boolean Combinations of Class Expressions

- 458 • unionOf
- 459 • complementOf
- 460 • intersectionOf

461 2.3.3 Arbitrary Cardinality

- 462 • minCardinality
- 463 • maxCardinality
- 464 • cardinality

465 2.3.4 Filler Information

- 466 • hasValue
- 467

468

3 ebXML Registry Overview

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

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

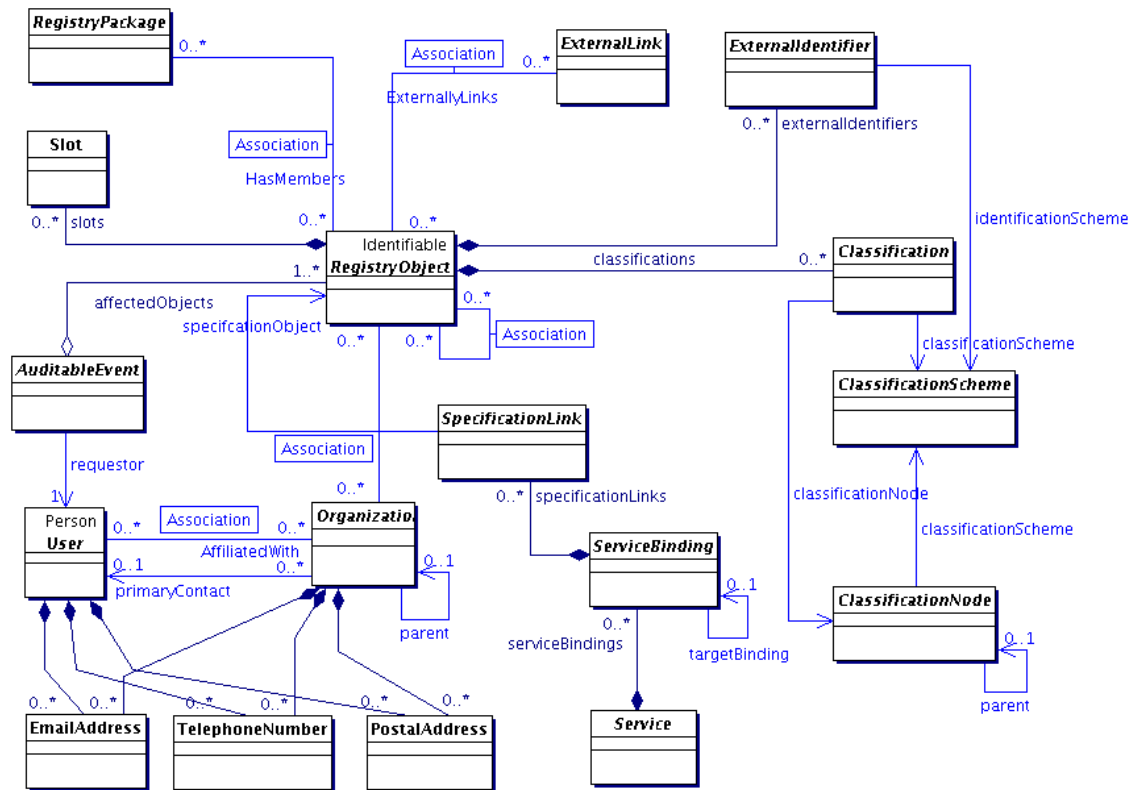
472 The information presented is informative and is not intended to replace the normative information defined
473 by ebXML Registry.

3.1 Overview of [ebRIM]

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

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

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



482

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

483

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

488

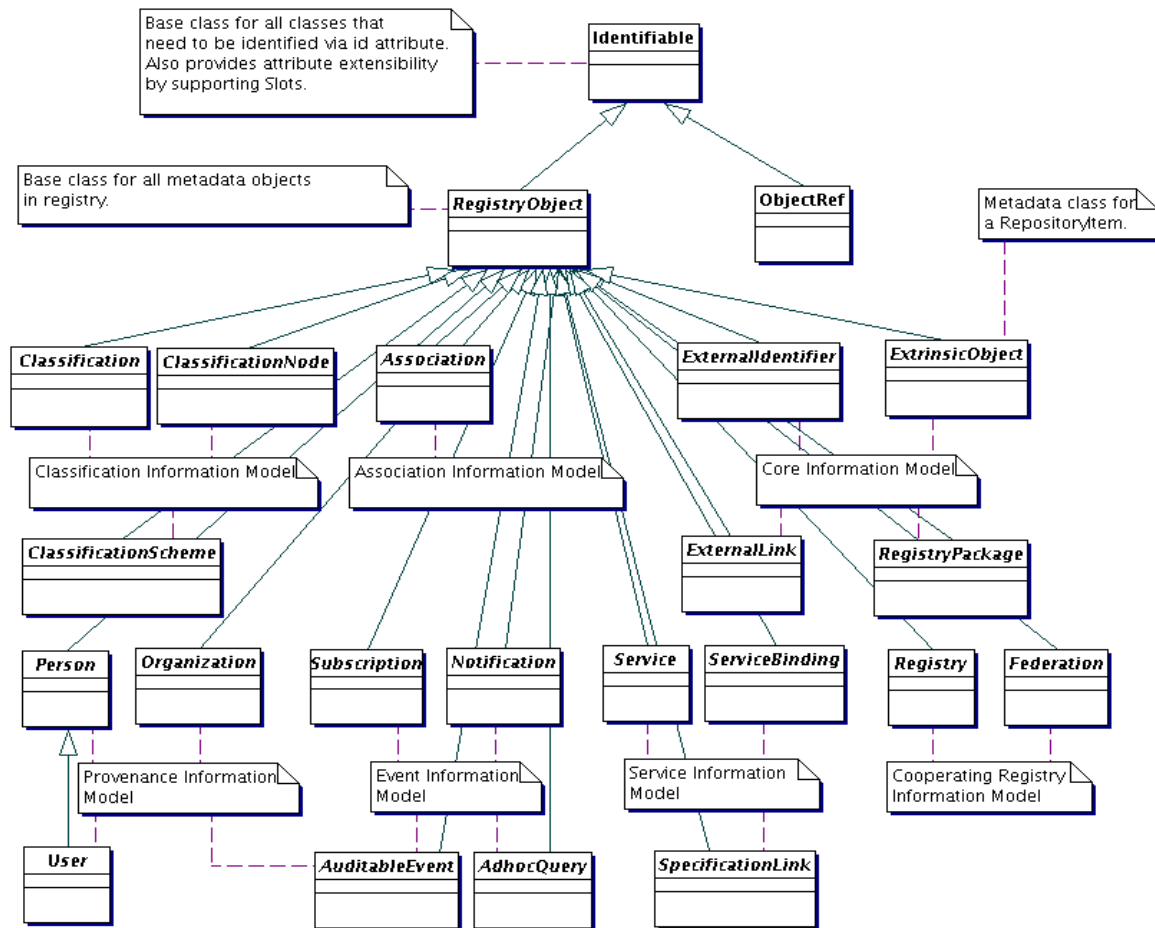


Figure 2: ebXML Registry Information Model, Inheritance View

490

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

492 3.1.1 RegistryObject

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

496 3.1.2 Object Identification

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

498

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

500

Listing 1: Example of id attribute

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

503

```
504 <rim:Organization id="uurn:oasis:Organization">
```

505

Listing 2: Example of human friendly id attribute

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

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

510

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

513

Listing 3: Example of lid Attribute

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

516

```
517 <rim:Organization id=${ACME_ORG_ID}  
518   lid="urn:acme:ACMEOrganization">  
519  
520   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}  
521     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}  
522     value="ACME"/>  
523   </rim:ExternalIdentifier>  
524  
525 </rim:Organization>
```

526

Listing 4: Example of ExternalIdentifier

527 3.1.3 Object Naming and Description

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

530

```
531 <rim:Organization id=${ACME_ORG_ID}  
532   lid="urn:acme:ACMEOrganization">  
533  
534   <rim:Name>  
535     <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>  
536   </rim:Name>  
537   <rim:Description>  
538     <rim:LocalizedString value="ACME is a provider of Java software."  
539       xml:lang="en-US"/>  
540   </rim:Description>  
541  
542   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}  
543     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}  
544     value="ACME"/>  
545   </rim:ExternalIdentifier>  
546 </rim:Organization>
```

547

Listing 5: Example of Name and Description

548

549 3.1.4 Object Attributes

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

552 3.1.4.1 Slot Attributes

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

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

559

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

```

562 <rim:Slot name="urn:acme:slot:NASDAQSymbol">
563   <rim:ValueList>
564     <rim:Value>ACME</rim:Value>
565   </rim:ValueList>
566 </rim:Slot>
567
568   <rim:Name>
569     <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
570   </rim:Name>
571   <rim:Description>
572     <rim:LocalizedString value="ACME makes Java. Provider of free Java
573 software."
574                               xml:lang="en-US"/>
575   </rim:Description>
576   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
577     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
578     value="ACME"/>
579   </rim:ExternalIdentifier>
580 </rim:Organization>
581

```

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.

```

588
589 <rim:Organization id=${ACME_ORG_ID}
590   lid="urn:acme:ACMEOrganization">
591   <rim:Slot name="urn:acme:slot:NASDAQSymbol">
592     <rim:ValueList>
593       <rim:Value>ACME</rim:Value>
594     </rim:ValueList>
595   </rim:Slot>
596   <rim:Name>
597     <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
598   </rim:Name>
599   <rim:Description>
600     <rim:LocalizedString value="ACME makes Java. Provider of free Java
601 software." xml:lang="en-US"/>
602   </rim:Description>
603   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
604     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
605     value="ACME"/>
606   </rim:ExternalIdentifier>
607
608   <!--Classify Organization as a Software Publisher using NAICS Taxonomy-->
609   <rim:Classification id=${CLASSIFICATION_ID}
610     classificationNode=${NAICS_SOFTWARE_PUBLISHER_NODE_ID}
611     classifiedObject=${ACME_ORG_ID}>
612
613 </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:

622 [http://www.oasis-](http://www.oasis-open.org/committees/regrep/documents/3.0/canonical/SubmitObjectsRequest_AssociationTypeScheme.xml)
623 [open.org/committees/regrep/documents/3.0/canonical/SubmitObjectsRequest_AssociationTypeScheme.x](http://www.oasis-open.org/committees/regrep/documents/3.0/canonical/SubmitObjectsRequest_AssociationTypeScheme.xml)
624 [ml](http://www.oasis-open.org/committees/regrep/documents/3.0/canonical/SubmitObjectsRequest_AssociationTypeScheme.xml)

625 [ebRIM] allows this scheme to be extensible.

626 The following example shows an Association between the ACME Organization instance and a Service
627 instance with the associationType of “OffersService”. This indicates that ACME Organization offers the
628 specified service (Service instance is not shown).

629

```
630 <rim:Association  
631     id=${ASSOCIATION_ID}  
632     associationType=${CANONICAL_ASSOCIATION_TYPE_OFFERS_SERVICE_ID}  
633     sourceObject=${ACME_ORG_ID}  
634     targetObject=${ACME_SERVICE1_ID}/>
```

635

Listing 8: Example of Object Association

636 3.1.7 Object References To Web Content

637 Any RegistryObject MAY reference web content that are maintained outside the registry using association
638 to an ExternalLink instance that contains the URL to the external web content. The following example
639 shows the ACME Organization with an Association to an ExternalLink instance which contains the URL to
640 ACME’s web site. The associationType of the Association MUST be of type “ExternallyLinks” as defined
641 by [ebRIM].

642

```
643 <rim:ExternalLink externalURI="http://www.acme.com"  
644     id=${ACME_WEBSITE_EXTERNAL_ID}>  
645 <rim:Association  
646     id=${EXTERNALLYLINKS_ASSOCIATION_ID}  
647     associationType=${CANONICAL_ASSOCIATION_TYPE_EXTERNALLY_LINKS_ID}  
648     sourceObject=${ACME_WEBSITE_EXTERNAL_ID}  
649     targetObject=${ACME_ORG_ID}/>
```

650

Listing 9: Example of Reference to Web Content Using ExternalLink

651 3.1.8 Object Packaging

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

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

659

```
660 <rim:RegistryPackage  
661     id=${ACME_SERVICES_PACKAGE_ID}>  
662     <rim:RegistryObjectList>  
663         <rim:ObjectRef id=${ACME_SERVICE1_ID}>  
664             <rim:RegistryPackage  
665                 id=${ACME_PURCHASING_SERVICES_PACKAGE_ID}>  
666                 <rim:ObjectRef id=${ACME_PURCHASING_SERVICE1_ID}>  
667                 <rim:ObjectRef id=${ACME_PURCHASING_SERVICE2_ID}>  
668             </rim:RegistryPackage>  
669             <rim:RegistryPackage  
670                 id=${ACME_HR_SERVICES_PACKAGE_ID}>  
671                 <rim:ObjectRef id=${ACME_HR_SERVICE1_ID}>  
672                 <rim:ObjectRef id=${ACME_HR_SERVICE2_ID}>  
673             </rim:RegistryPackage>  
674         </rim:RegistryObjectList>  
675     </rim:RegistryPackage>
```

676

Listing 10: Example of Object Packaging Using RegistryPackages

677 3.1.9 ExtrinsicObject

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

682 3.1.10 Service Description

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

686 3.2 Overview of [ebRS]

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

4 Representing OWL Lite Constructs in ebRIM

689

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

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

698

```
699 ClassScheme (id, home, lid, objectType, status, versionName, comment_,...);  
700  
701 ClassificationNode(accessControlPolicy, id, lid, home, objectType, code, parent,  
702 path,versionName, comment_...)  
703  
704 Association(accessControlPolicy, id, lid, home, objectType, associationType,  
705 sourceObject, targetObject, isConfirmedBySourceOwner,versionName, comment_  
706 isConfirmedByTargetOwner,...)  
707  
708 Name_(charset, lang, value, parent,...)  
709  
710 Classification (id, objectType, lid, home, classificationNode, versionName,  
711 comment_, classificationScheme, classifiedObject, nodeRepresentation,...);  
712  
713 ExtrinsicObject (id, lid, home, objectType,...)
```

714

ebXML Registry Relations

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

4.1 Representing RDF Schema Features in ebRIM

718

4.1.1 owl:Class → rim:ClassificationNode

719

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

721

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

724

Example owl:Class

725

```
726 <ClassificationScheme id=${GeographicalEntity}  
727 name="GeographicalEntity"/>  
728  
729 <rim:ClassificationNode id=${City} code='City'  
730 parent=${GeographicalEntity}>  
731 </rim:ClassificationNode>
```

732

Example Corresponding ebRIM construct ClassificationNode

733

734 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

735

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

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

740

```
741 <rdf:Property rdf:ID="hasAirport">  
742   <rdfs:domain rdf:resource="#City"/>  
743   <rdfs:range rdf:resource="#AirPort"/>  
744 </rdf:Property>
```

745

Example rdf:Property

746

```
747 <rim:Association id=${hasAirport}  
748 associationType='urn:oasis:names:tc:ebxml-  
749 regrep:profile:webontology:AssociationType:OWL:HasProperty'  
750 sourceObject= ${city}  
751 targetObject=${Airport} >  
752 </rim:Association>
```

753

Example: ebRIM construct Association corresponding to rdf:Property

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

756 4.1.3 rdfs:subPropertyOf → rim:Association Type SubPropertyOf

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

760

```
761 <rdf:Property rdf:ID="creditCardPayment">  
762   <rdfs:subPropertyOf rdf:Resource="#paymentMethods"/>  
763 </rdf:Property>
```

764

Example rdfs:subPropertyOf

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

767

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

771

```
772 <rim:Association id=${subPropertyOfID}  
773 associationType='urn:oasis:names:tc:ebxml-  
774 regrep:profile:webontology:AssociationType:OWL:SubPropertyOf '  
775 sourceObject= ${creditCardPayment} targetObject=${paymentMethods} >  
776 </rim:Association>
```

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

779 4.1.4 rdfs:subClassOf → rim:Association Type SubClassOf

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

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

```
790  
791 <owl:Class rdf:ID="AirReservationServices">  
792   <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-  
793     s/1.0/Profile.owl#Profile"/>  
794   <rdfs:subClassOf rdf:resource="#AirServices"/>  
795 </owl:Class>
```

796 Example rdfs:subClassOf

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

```
801  
802 <rim:Association id=${subClassOf}  
803 associationType='urn:oasis:names:tc:ebxml-  
804 regrep:profile:webontology:AssociationType:OWL:SubClassOf'  
805 sourceObject= ${AirReservationServices} targetObject=${OWL-S_Profile} >  
806 </rim:Association>  
807 <rim:Association id=${subClassOf2}  
808 associationType='urn:oasis:names:tc:ebxml-  
809 regrep:profile:webontology:AssociationType:OWL:SubClassOf'  
810 sourceObject= ${AirReservationServices} targetObject=${AirServices} >  
811 </rim:Association>
```

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

816 4.1.5 owl:Individual → rim:ExtrinsicObject

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

```
820  
821 <owl:Class rdf:ID="TravelWebService">  
822 </owl:Class>  
823  
824 <owl:ObjectProperty rdf:ID="paymentMethod">  
825   <rdfs:domain rdf:resource="#TravelWebService"/>  
826   <rdfs:range rdf:resource="#PossiblePaymentMethods"/>  
827 </owl:ObjectProperty >
```

828 Example owl:Class example

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

```
833  
834 <TravelWebService rdf:ID="MyTravelWebService">  
835   <paymentMethod> Cash </paymentMethod>  
836 </TravelWebService>
```

837 Example owl:Individual example

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

841 4.2 Representing OWL (In)Equality Constructs in ebXML RIM

842 4.2.1 owl:equivalentClass, owl:equivalentProperty → rim:Association Type 843 EquivalentTo

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

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

851 4.2.2 owl:sameAs → rim:Association Type SameAs

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

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

857

```
858 <rdf:Description rdf:about="#MyAirReservationService">  
859   <owl:sameAs rdf:resource="#THYAirReservationService"/>  
860 </rdf:Description>
```

861

Example owl:sameAs

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

```
864 <rim:Association id=${sameAs1} associationType='urn:oasis:names:tc:ebxml-  
865   regrep:profile:webontology:AssociationType:OWL:SameAs'  
866   sourceObject=${MyAirReservationService} targetObject=${THYAirReservationService}  
867   >  
868 </rim:Association>
```

869

Example Corresponding ebRIM construct Association

870

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

873 4.2.3 owl:differentFrom → rim:Association Type DifferentFrom

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

878

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

882

Example owl:differentFrom

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


```

886 <rim:Association id=${differentFrom1}
887 associationType='urn:oasis:names:tc:ebxml-
888 regrep:profile:webontology:AssociationType:OWL:DifferentFrom'
889   sourceObject= ${MyAirReservationService}
890   targetObject=${THYAirReservationService} >
891 </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:

```

901 <owl:AllDifferent>
902   <owl:distinctMembers rdf:parseType="Collection">
903     <WebService rdf:about="#MyCarService"/>
904     <WebService rdf:about="#MyFlightService"/>
905     <WebService rdf:about="#MyHotelService"/>
906   </owl:distinctMembers>
907 </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.

```

913
914 <rim:RegistryPackage id = ${Collection} >
915   <rim:Slot name=urn:oasis:names:tc:ebxml-
916   regrep:profile:webontology:slot:packagetype>
917     <rim:ValueList>
918       <rim:Value>allDifferent</rim:Value>
919     </rim:ValueList>
920   </rim:Slot>
921 </rim:RegistryPackage>
922 <rim:Association id = ${HasMemberRegistryPackageAssoc1}
923 associationType = "urn:oasis:names:tc:ebxml-
924 regrep:AssociationType:HasMember" sourceObject = ${Collection}
925 targetObject = ${MyCarService} />
926 <rim:Association id = ${HasMemberRegistryPackageAssoc2}
927 associationType = "urn:oasis:names:tc:ebxml-regrep:HasMember"
928 sourceObject = ${Collection}
929 targetObject = ${MyFlightService} />
930
931 <rim:Association id = ${HasMemberRegistryPackageAssoc3}
932 associationType = "urn:oasis:names:tc:ebxml-regrep:HasMember"
933 sourceObject = ${Collection}
934 targetObject = ${MyHotelService} />

```

Example Corresponding ebRIM Representation

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

937 4.3 Representing OWL Property Characteristics in ebRIM

938 4.3.1 owl:ObjectProperty → rim:Association Type objectProperty

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

942

```
943 <owl:ObjectProperty rdf:ID="hasAirport">  
944   <rdfs:domain rdf:resource="#City"/>  
945   <rdfs:range rdf:resource="#AirPort"/>  
946 </owl:ObjectProperty>
```

947

Example owl:ObjectProperty

948

```
949 <rim:Association id=${hasAirport} associationType='urn:oasis:names:tc:ebxml-  
950 regrep:profile:webontology:AssociationType:OWL:ObjectProperty'  
951   sourceObject= ${City} targetObject=${Airport} >  
952 </rim:Association>
```

953

Example Corresponding ebRIM construct Association

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

957 4.3.2 owl:DatatypeProperty → rim:Association Type DatatypeProperty

958 Similarly, to represent OWL DatatypeProperty in ebXML, a new Association Type called
959 "DatatypeProperty" MUST be defined. Consider the following example which defines an datatype property
960 "hasPrice" whose domain is the "AirReservationServices" and whose range is "XMLSchema
961 nonNegativeInteger". How OWL XML Schema types are handled in ebXML RIM is described in Section
962 4.9.

```
963 <owl:DatatypeProperty rdf:ID="hasPrice">  
964   <rdfs:domain rdf:resource="#AirReservationServices"/>  
965   <rdfs:range  
966   rdf:resource="http://www.w3.org/2001/XMLSchema/nonNegativeInteger"/>  
967 </owl:DatatypeProperty>
```

968

Example owl:DatatypeProperty

969

```
970 <rim:Association id=${hasPrice}  
971 associationType='urn:oasis:names:tc:ebxml-  
972 regrep:profile:webontology:AssociationType:OWL:DatatypeProperty'  
973   sourceObject= ${AirReservationServices}  
974   targetObject=urn:www.w3.org:2001/XMLSchema:nonNegativeInteger >  
975 </rim:Association>
```

976

Example Corresponding ebRIM construct Association

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

979 4.3.3 owl:TransitiveProperty → rim:Association Type TransitiveProperty

980 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)
981 [McGuinness, Harmelen]. Transitive property is a subproperty of ObjectProperty and MUST be defined as
982 a new Association Type called "TransitiveProperty" in ebRIM.

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

985

```

986 <owl:ObjectProperty rdf:ID="succeeds">
987   <rdf:type rdf:resource="#owl:TransitiveProperty" />
988   <rdfs:domain rdf:resource="#TravelWebService" />
989   <rdfs:range rdf:resource="#TravelWebService" />
990 </owl:ObjectProperty>

```

991 **Example owl:TransitiveProperty**

```

992
993 <rim:Association id=${succeeds}
994 associationType='urn:oasis:names:tc:ebxml-
995 regrep:profile:webontology:AssociationType:OWL:TransitiveProperty'
996 sourceObject= ${TravelWebService} targetObject=${TravelWebService} >
997 </rim:Association>

```

998 **Example Corresponding ebRIM construct Association**

999 Assume the following two definitions which declare three Web service instances from TravelWebService
1000 class where "MyHotelAvailabilityService" service succeeds "MyAirReservationService" and
1001 "MyInsuranceService" succeeds "MyHotelAvailabilityService". Since "succeeds" is a transitive property, it
1002 follows that "MyInsuranceService" succeeds "MyAirReservationService" although this fact is not explicitly
1003 stated.

1004

```

1005 <TravelWebService rdf:ID="MyHotelAvailabilityService">
1006   <succeeds rdf:resource="#MyAirReservationService" />
1007 </TravelWebService>
1008
1009 <TravelWebService rdf:ID="MyInsuranceService">
1010   <succeeds rdf:resource="#MyHotelAvailabilityService" />
1011 </TravelWebService>

```

1012 **Example owl:TransitiveProperty instances**

1013 To make any use of this transitive property in ebXML registries, coding is necessary to find out the implied
1014 information. The adhoc query presented in Section 6.16 MUST be available in the registry to handle this
1015 semantics.

1016 **4.3.4 owl:inverseOf → rim:Association Type InverseOf**

1017 In OWL, one property may be stated to be the inverse of another property. If the property P1 is stated to
1018 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
1019 P1 property [McGuinness, Harmelen].

1020 Consider, for example, the "succeeds" property defined in Section 4.3.3. To denote that a certain Web
1021 service instance precedes another during execution, we may define the "precedes" property as an inverse
1022 of the "succeeds" property as follows:

1023

```

1024 <owl:ObjectProperty rdf:ID="precedes">
1025   <owl:inverseOf rdf:resource="#succeeds" />
1026 </owl:ObjectProperty>

```

1027 **Example owl:inverseOf Property**

1028

```

1029 <rim:Association id=${inverseOf1}
1030 associationType='urn:oasis:names:tc:ebxml-
1031 regrep:profile:webontology:AssociationType:OWL:InverseOf'
1032 sourceObject= ${precedes} targetObject=${succeeds} >
1033 </rim:Association>

```

1034 **Example Corresponding ebRIM construct Association**

1035 Assume that we want to find all the Web services which can succeed a given Web service. In such a
1036 case, we need not only find all the Web services which succeeds this given Web service, that is the target
1037 objects of "succeeds" Association instance, but we also need to find all the sourceObjects of the
1038 "precedes" Association instance since "precedes" is declared to be the "inverseOf" succeeds Association

1039 instance. This can be achieved through the adhoc query presented in Section 6.19.

1040 Alternatively, one might use the additional semantics that this profile supports would be to cause inferred
1041 information to be produced and stored along with new data as that new data was inserted into the reg/rep.
1042 There is a trade off here: in this way, the extra work of inferring is only done at insertion/update time,
1043 instead of at query time. However, an insertion or an update will require all the inferred data to be inserted
1044 whether it will be used or not and hence will cause considerable maintenance overhead.

1045 4.3.5 owl:SymmetricProperty → rim:Association Type SymmetricProperty

1046 In OWL, if a property is symmetric, then if the pair (x,y) is an instance of the symmetric property P, then
1047 the pair (y,x) is also an instance of P [McGuinness, Harmelen]. Symmetric property is a subproperty of
1048 ObjectProperty in OWL. Consider the OWL class “WebService” and the “complements” symmetric
1049 property:

```
1050 <owl:Class rdf:ID="WebService">  
1051   <rdfs:subClassOf  
1052     rdf:resource="http://www.w3.org/2000/01/rdfschema#Resource"/>  
1053 </owl:Class>  
1054 <owl:SymmetricProperty rdf:ID="complements">  
1055   <rdfs:domain rdf:resource="#WebService"/>  
1056   <rdfs:range rdf:resource="#WebService"/>  
1057 </owl:SymmetricProperty>
```

1058 Example owl:SymmetricProperty

```
1059 <rim:Association id=${complements}  
1060 associationType='urn:oasis:names:tc:ebxml-  
1061 regrep:profile:webontology:AssociationType:OWL:SymetricProperty'  
1062 sourceObject= ${WebService} targetObject=${WebService} >  
1063 </rim:Association>
```

1064 Example Corresponding ebRIM construct Association

1065 Given that HotelReservationWebService complements AirReservationWebService, it is possible to
1066 deduce that AirReservationWebService complements HotelReservationWebService.

1067 owl:SymmetricProperty MUST be defined as a new type of Association in ebRIM called
1068 “SymmetricProperty”. Furthermore the adhoc query presented in Section 6.20 MUST be available in the
1069 Registry to retrieve symmetric Associations of a ClassificationNode.

1070 4.3.6 owl:FunctionalProperty → rim:Association Type FunctionalProperty

1071 In OWL, if a property is a FunctionalProperty, then it has no more than one value for each individual (it
1072 may have no values for an individual) [McGuinness, Harmelen]. The range of a FunctionalProperty can be
1073 either an Object or a datatype. Consider, for example, the “hasPrice” Functional property which has a
1074 unique price:

```
1075 <owl:DatatypeProperty rdf:ID="hasPrice">  
1076   <rdf:type rdf:resource="&owl;FunctionalProperty" />  
1077   <rdfs:domain rdf:resource="#AirReservationServices"/>  
1078   <rdfs:range  
1079     rdf:resource="http://www.w3.org/2001/XMLSchema/nonNegativeInteger"/>  
1080 </owl:DatatypeProperty>
```

1081 Example owl:FunctionalProperty

```
1082 <rim:Association id=${hasPrice}  
1083 associationType='urn:oasis:names:tc:ebxml-  
1084 regrep:profile:webontology:AssociationType:OWL:FunctionalProperty'  
1085 sourceObject= ${AirReservationServices}  
1086 targetObject=${uurn:www.w3.org:2001/XMLSchema:nonNegativeInteger} >  
1087 </rim:Association>
```

1088 Example Corresponding ebRIM construct Association

1089 ebXML RIM MUST contain a new Association Type called "FunctionalProperty" to express this semantics.
1090 Furthermore the he adhoc query presented in Section 6.21 MUST be available in the Registry to retrieve
1091 functional Associations of a ClassificationNode.

1092 4.3.7 owl:InverseFunctionalProperty → rim:Association Type 1093 InverseFunctionalProperty

1094 In OWL, if a property is inverse functional then the inverse of the property is functional. Thus the inverse
1095 of the property has at most one value for each individual [McGuinness, Harmelen]. InverseFunctional
1096 properties (IFPs) are like keys. An individual filling the range role in an inverseFunctional property
1097 instance identifies the individual in the domain role of that same property instance. In other words, if a
1098 semantic web tool encounters two individuals with the same value for an inverseFunctional property, it
1099 can be inferred that they are actually the same individual.

1100 As an example, the ObjectProperty "finalDestination" indicates that each flight arrives to only one airport
1101 as its final destination.

```
1102 <owl:ObjectProperty rdf:ID="finalDestination">  
1103   <rdf:type rdf:resource="&owl;InverseFunctionalProperty" />  
1104   <rdfs:domain rdf:resource="#Airport"/>  
1105   <rdfs:range rdf:resource="#Flight"/>  
1106 </owl:ObjectProperty>
```

1107 Example owl:InverseFunctionalProperty

```
1108 <rim:Association id=${finalDestination}  
1109 associationType='urn:oasis:names:tc:ebxml-  
1110 regrep:profile:webontology:AssociationType:OWL:InverseFunctionalProperty'  
1111   sourceObject= ${Airport} targetObject=${Flight} >  
1112 </rim:Association>
```

1113 Example Corresponding ebRIM construct Association

1114 ebRIM MUST contain a new Association Type called "InverseFunctionalProperty" to express this
1115 semantics. Furthermore the adhoc query presented in Section 6.22 MUST be available in the Registry to
1116 retrieve inverse functional Associations of a ClassificationNode.

1117 4.4 OWL Property Restrictions in ebXML RIM

1118 An important construct of OWL is "owl:Restriction". In RDF, a property has a global scope, that is, no
1119 matter what class the property is applied to, the range of the property is the same. "owl:Restriction", on the
1120 other hand, has a local scope; restriction is applied on the property within the scope of the class where it is
1121 defined. This makes property definitions more reusable by factoring out class specific characteristics of
1122 the property into the class description.

1123 For example, we may define a property "paymentMethod" for travel Web services in general and we may
1124 state that the range of this property is the class "PossiblePaymentMethods". Then, for
1125 "AirReservationServices", we may wish to restrict "paymentMethod" property to, say, "CreditCard" class as
1126 demonstrated in the following two examples:

```
1127  
1128 <owl:ObjectProperty rdf:ID="paymentMethod">  
1129   <rdfs:domain rdf:resource="#TravelWebService"/>  
1130   <rdfs:range rdf:resource="#PossiblePaymentMethods"/>  
1131 </owl:ObjectProperty >
```

1132 Example owl:ObjectProperty "paymentMethod"

```
1133  
1134 <owl:Class rdf:ID="AirReservationServices">  
1135   <rdfs:subClassOf>  
1136     <owlRestriction>  
1137       <owl:onProperty rdf:resource="#paymentMethod"/>  
1138       <owl:allValuesFrom rdf:resource= "#CreditCard"/>  
1139     </owl:Restriction>  
1140   </rdfs:subClassOf>
```

1141 </owl:Class>

1142 **Example owl:Restriction on ObjectProperty “paymentMethod”**

1143 A new Association Type of “restriction” SHOULD be defined to represent OWL restriction. A slot of this
1144 Association Type SHOULD indicate the whether the restriction is “allValuesFrom” or “someValuesFrom”.
1145 When such restriction is submitted to the system, the registry MUST create a new Association instance,
1146 say, “paymentMethod_1” of AssociationType “ObjectProperty” is created whose sourceObject is
1147 “AirReservationServices” and the targetObject is “CreditCard”. “paymentMethod_1” Association instance
1148 is related with the “paymentMethod” Association instance by using an instance of the Association Type
1149 “Restriction” as shown in the following example:

1150

```
1151 <rim:Association id = ${paymentMethod_1}
1152             associationType =
1153 "urn:oasis:names:tc:ebxml-
1154 regrep:profile:webontology:AssociationType:OWL:ObjectProperty"
1155             sourceObject = ${AirReservationServices}
1156             targetObject = ${CreditCard}>
1157 </rim:Association>
1158
1159 <rim:Association id = ${paymentMethodRestriction}
1160             associationType =
1161 "urn:oasis:names:tc:ebxml-
1162 regrep:profile:webontology:AssociationType:OWL:Restriction"
1163             sourceObject = ${paymentMethod}
1164             targetObject = ${paymentMethod_1}>
1165     <rim:Slot name="urn:oasis:names:tc:ebxml-
1166 regrep:profile:webontology:slot:restrictionType">
1167         <rim:ValueList>
1168             <rim:Value>allValuesFrom</rim:Value>
1169         </rim:ValueList>
1170     </rim:Slot>
1171 </rim:Association>
```

1172 **Example Handling owl:Restriction in ebXML Registry**

1173 Obviously, this serves the purpose of reusing the "paymentMethod" property. Otherwise, a new property
1174 "paymentMethodCC" can be defined between "AirReservationServices" and the "CreditCard" classes as
1175 shown in the following:

1176

```
1177 <owl:ObjectProperty rdf:ID="paymentMethodCC">
1178     <rdfs:domain rdf:resource="#AirReservationServices"/>
1179     <rdfs:range rdf:resource="#CreditCard"/>
1180 </owl:ObjectProperty >
```

1181 **Example owl:ObjectProperty “paymentMethodCC”**

1182 **4.5 Representing OWL Restricted Cardinality in ebXML RIM**

1183 **4.5.1 owl:minCardinality (only 0 or 1)**

1184 In OWL, cardinality is stated on a property with respect to a particular class. If a minCardinality of 1 is
1185 stated on a property with respect to a class, then any instance of the class will have at least one value for
1186 the restricted property. This restriction is another way of saying that the property is required to have a
1187 value for all instances of the class. In OWL Lite, the only minimum cardinalities allowed are 0 or 1. A
1188 minimum cardinality of zero on a property just states (in the absence of any more specific information)
1189 that the property is optional with respect to a class [McGuinness, Harmelen].

1190 Consider for example the following OWL code which states that each instance of a “WebService” class
1191 must have at least one price:

```
1192 <owl:Class rdf:ID="WebService">
1193     <rdfs:subClassOf>
1194         <owl:Restriction>
1195             <owl:onProperty rdf:resource="#hasPrice"/>
```



```

1196         <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">
1197 1 </owl:minCardinality>
1198     </owl:Restriction>
1199 </rdfs:subClassOf>
1200 </owl:Class>

```

1201 **Example owl:minCardinality**

1202 In ebXML RIM, cardinalities of Association Types MUST be defined by associating a minCardinality slot
1203 with the Association Types as shown in the following example:

```

1204
1205 <rim:Association id = ${hasPriceMinCardinalityRestriction}
1206 associationType = "urn:oasis:names:tc:ebxml-
1207 regrep:profile:webontology:AssociationType:OWL:ObjectProperty"
1208 sourceObject = ${WebService}
1209 targetObject = ${Price}>
1210     <rim:Name>
1211         <rim:LocalizedString value = 'hasPrice' />
1212     </rim:Name>
1213     <rim:Slot name="urn:oasis:names:tc:ebxml-
1214 regrep:profile:webontology:slot:minCardinality">
1215         <rim:ValueList>
1216             <rim:Value>1</rim:Value>
1217         </rim:ValueList>
1218     </rim:Slot>
1219 </rim:Association>

```

1220 **Example Representing owl:minCardinality in ebRIM**

1221 **4.5.2 owl:maxCardinality (only 0 or 1)**

1222 In OWL, cardinality is stated on a property with respect to a particular class. If a maxCardinality of 1 is
1223 stated on a property with respect to a class, then any instance of that class will be related to at most one
1224 individual by that property. A maxCardinality 1 restriction is sometimes called a functional or unique
1225 property. It may be useful to state that certain classes have no values for a particular property. This
1226 situation is represented by a maximum cardinality of zero on the property [McGuinness, Harmelen].

1227 Consider for example the following OWL code which states that each instance of a "WebService" class
1228 can have at most one price:

```

1229 <owl:Class rdf:ID="WebService">
1230     <rdfs:subClassOf>
1231         <owl:Restriction>
1232             <owl:onProperty rdf:resource="#hasPrice"/>
1233             <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
1234 1 </owl:maxCardinality>
1235         </owl:Restriction>
1236     </rdfs:subClassOf>
1237 </owl:Class>

```

1238 **Example owl:maxCardinality**

1239 In ebXML RIM, cardinalities of Association Types MUST be defined by associating a maxCardinality slot
1240 with the Association Types as shown in the following example:

```

1241
1242 <rim:Association id = ${hasPriceMaxCardinalityRestriction}
1243 associationType = "urn:oasis:names:tc:ebxml-
1244 regrep:profile:webontology:AssociationType:OWL:ObjectProperty"
1245 sourceObject = ${WebService}"
1246 targetObject = ${Price}>
1247     <rim:Name>
1248         <rim:LocalizedString value = 'hasPrice' />
1249     </rim:Name>
1250     <rim:Slot name="urn:oasis:names:tc:ebxml-
1251 regrep:profile:webontology:slot:maxCardinality">
1252         <rim:ValueList>

```

```

1253         <rim:Value>1</rim:Value>
1254     </rim:ValueList>
1255     </rim:Slot>
1256 </rim:Association>

```

1257 **Example Representing owl:maxCardinality in ebRIM**

1258 **4.5.3 owl:cardinality (only 0 or 1)**

1259 In OWL Lite, cardinality is provided as a convenience when it is useful to state that a property on a class
 1260 has both minCardinality 0 and maxCardinality 0 or both minCardinality 1 and maxCardinality 1
 1261 [McGuinness, Harmelen].

1262 Consider for example the following OWL code which states that each instance of a "WebService" class
 1263 must have exactly one price:

```

1264 <owl:Class rdf:ID="WebService">
1265   <rdfs:subClassOf>
1266     <owl:Restriction>
1267       <owl:onProperty rdf:resource="#hasPrice"/>
1268       <owl:Cardinality rdf:datatype="&xsd;nonNegativeInteger"> 1
1269     </owl:Cardinality>
1270   </owl:Restriction>
1271 </rdfs:subClassOf>
1272 </owl:Class>

```

1273 **Example owl:Cardinality**

1274 In ebXML RIM, cardinalities of Association Types MUST be defined by associating a Cardinality slot with
 1275 the Association Types as shown in the following example:

```

1276
1277 <rim:Association id = ${hasPriceCardinalityRestriction}
1278 associationType = "urn:oasis:names:tc:ebxml-
1279 regrep:profile:webontology:AssociationType:OWL:ObjectProperty"
1280 sourceObject = ${WebService}
1281 targetObject = ${Price}>
1282   <rim:Name>
1283     <rim:LocalizedString value = 'hasPrice' />
1284   </rim:Name>
1285   <rim:Slot name="urn:oasis:names:tc:ebxml-
1286 regrep:profile:webontology:slot:cardinality">
1287     <rim:ValueList>
1288       <rim:Value>1</rim:Value>
1289     </rim:ValueList>
1290   </rim:Slot>
1291 </rim:Association>

```

1292 **Example Representing owl:Cardinality in ebRIM**

1293 **4.6 Representing OWL Class Intersection in ebXML RIM**

1294 OWL provides the means to manipulate class extensions using basic set operators. In OWL lite, only
 1295 "owl:intersectionOf" is available which defines a class that consists of exactly all objects that belong to all
 1296 the classes specified in the intersection definition. In the following example, "AirReservationServices" is
 1297 defined as the intersection of "AirServices" and "ReservationServices":

```

1298
1299 <owl:Class rdf:ID="AirReservationServices">
1300   <owl:intersectionOf rdf:parseType="Collection">
1301     <owl:Class rdf:about="#AirServices" />
1302     <owl:Class rdf:about="#ReservationServices" />
1303   </owl:intersectionOf>
1304 </owl:Class>

```

1305 **Example owl:intersectionOf**

1306 In ebXML RIM "owl:intersectionOf" set operator MUST be represented as follows:

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

1311

```

1312 <rim:ClassificationNode id = ${AirReservationServices} code =
1313 "AirReservationServices">
1314   <rim:Slot name=urn:oasis:names:tc:ebxml-
1315     regrep:profile:webontology:slot:intersectionOf>
1316     <rim:ValueList>
1317       <rim:Value>${AirServices}</rim:Value>
1318       <rim:Value>${ReservationServices}</rim:Value>
1319     </rim:ValueList>
1320   </rim:Slot>
1321 </rim:ClassificationNode>
1322

```

1323 Example Defining Intersection of ClassificationNodes in ebRIM

1324 When such a representation is used to create a complex class (a new ClassificationNode) in RIM, it
 1325 becomes possible to infer that the objects (instances) classified by all of the classes
 1326 (ClassificationNodes) specified in the Intersection definition, are also classified by this complex class. The
 1327 adhoc query presented in Section 6.23 MUST be available in the ebXML Registry to retrieve the direct
 1328 instances of the complex class and also the instances of the intersection of the classes.

1329

1330 4.7 Representing OWL Versioning in ebXML RIM

1331 4.7.1 owl:versionInfo, owl:priorVersion

1332 An owl:versionInfo statement generally has as its object a string giving information about this version, for
 1333 example RCS/ CVS keywords. This statement does not contribute to the logical meaning of the ontology
 1334 other than that given by the RDF(S) model theory [McGuinness, Harmelen].

1335 An owl:priorVersion statement contains a reference to another ontology. This identifies the specified
 1336 ontology as a prior version of the containing ontology [McGuinness, Harmelen].

1337 In ebXML, since a RegistryObject MAY have several versions, a logical id (called lid) is also defined which
 1338 is unique for different logical objects. However the lid attribute value MUST be the same for all versions of
 1339 the same logical object. Therefore, almost for all the RegistryObjects the version information is kept
 1340 through "versionName" and "comment" attributes of the "VersionInfo" ebRIM Class.

1341 "owl:version" information MUST be stored in the "versionName" and "comment" attributes of the
 1342 VersionInfo ebRIM class.

1343 It should be noted that in freebXML implementation the versionInfo is flattened and the "versionName"
 1344 and "comment_" are provided as direct attributes of database tables.

```

1345 <owl:Ontology rdf:about="">
1346   <owl:versionInfo>v 1.17 2003/02/26 12:56:51 </owl:versionInfo>
1347 </owl:Ontology>

```

1348 Example owl:versionInfo

```

1349 <rim:ClassificationScheme
1350 lid= ${exampleOntology}
1351 id=${exampleOntology} isInternal="true"
1352 nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode">
1353 <rim:versionInfo>
1354 <rim:versionName>
1355   <rim:LocalizedString charset="UTF-8" value="v 1.17 2003/02/26
1356   12:56:51"/>
1357 </rim:versionName>
1358 </rim:versionInfo>
1359 </rim:ClassificationScheme>

```

1360 **Example rim:versionName**

1361 4.8 Representing OWL Annotation Properties in ebXML RIM

1362 4.8.1 rdfs:label

1363 rdfs:label is an instance of rdf:Property that may be used to provide a human-readable version of a
1364 resource's name [Brickley, Guha].

1365 In ebXML RIM, human readable names of resources are provided through rim:Name. rdfs:label MUST be
1366 expressed through rim:Name.

1367

```
1368 <owl:Class rdf:ID="AirReservationServices">  
1369   <rdfs:label>Air Reservation Services</rdfs:label>  
1370 </owl:Class>
```

1371 Example rdfs:label

1372

```
1373 <rim:ClassificationNode id = ${AirReservationServices} code =  
1374 'AirReservationServices'  
1375   <rim:Name>  
1376     <rim:LocalizedString value = 'Air Reservation Services' />  
1377   </rim:Name>  
1378 </rim:ClassificationNode>
```

1379 Example rim:Name

1380 4.8.2 rdfs:comment

1381 rdfs:comment is an instance of rdf:Property that may be used to provide a human-readable description of
1382 a resource [Brickley, Guha].

1383 In ebXML RIM, this construct MUST be expressed through rim:Description.

1384

```
1385 <owl:Class rdf:ID="AirReservationServices">  
1386   <rdfs:comment>Open Travel Alliance Air Reservation Services  
1387 </rdfs:comment>  
1388 </owl:Class>
```

1389 Example rdfs:comment

1390

```
1391 <rim:ClassificationNode id = ${AirReservationServices} code =  
1392 'AirReservationServices'  
1393   <rim:Description>  
1394     <rim:LocalizedString value = 'Open Travel Alliance Air  
1395 Reservation Services' />  
1396   </rim:Description>  
1397 </rim:ClassificationNode>
```

1398 Example: rim:Description

1399 4.8.3 rdfs:seeAlso

1400 rdfs:seeAlso is an instance of rdf:Property that is used to indicate a resource that might provide additional
1401 information about the subject resource [Brickley, Guha].

1402 This construct MUST be expressed in ebXML RIM by defining a new Association Type called "SeeAlso" to
1403 express this semantics.

1404

```
1405 <owl:Class rdf:ID="AirReservationServices">  
1406   <rdfs:seeAlso rdf:resource="http://www.opentravel.org" />
```

1407 </owl:Class>

1408 **Example rdfs:seeAlso**

```
1409 <rim:ClassificationNode id = ${AirReservationServices} code =  
1410 'AirReservationServices'>  
1411 </rim:ClassificationNode>  
1412  
1413 <rim:ExternalLink id = ${exampleExternalLink}  
1414 externalURI= "http://www.opentravel.org" >  
1415 </rim:ExternalLink>  
1416  
1417 <rim:Association id = ${seeAlsoID}  
1418 associationType = 'urn:oasis:names:tc:ebxml-  
1419 regrep:profile:webontology:AssociationType:OWL:SeeAlso'  
1420 sourceObject = ${AirReservationServices}  
1421 targetObject = ${exampleExternalLink} />
```

1422 **Example rim:seeAlsoExternalLink**

1423 **4.9 OWL Datatypes in ebXML RIM**

1424 OWL allows the use of XML Schema datatypes to describe part of the datatype domain by simply
1425 including their URIs within an OWL ontology [McGuinness, Harmelen]. In ebXML, XML Schema datatypes
1426 MAY be used by providing an external link from the registry.

1427 The following example demonstrates how XML Schema datatype “integer” can be referenced through an
1428 ExternalLink whose id is 'urn:www.w3.org:2001/XMLSchema:integer' and how to define a
1429 DatatypeProperty, namely, “hasPrice”, whose target object is the defined to be ExternalLink 'integer':

```
1430 <rim:ExternalLink id = urn:www.w3.org:2001/XMLSchema:integer  
1431 externalURI="http://www.w3.org/2001/XMLSchema#integer" >  
1432 <rim:Name> <rim:LocalizedString value = "XML Schema integer"/>  
1433 </rim:Name>  
1434 </rim:ExternalLink>  
1435 <rim:Association id = ${hasPrice} associationType =  
1436 'urn:oasis:names:tc:ebxml-  
1437 regrep:AssociationType:DatatypeProperty'  
1438 sourceObject = ${AirReservationServices}  
1439 targetObject = urn:www.w3.org:2001/XMLSchema:integer >  
1440 <rim:Name> <rim:LocalizedString value ="hasPrice"/></rim:Name>  
1441 </rim:Association>
```

1443 **Example Corresponding ebRIM construct Association**

1444 5 Cataloging Service Profile

1445 The ebXML Registry provides the ability for a content cataloging service to be configured for any type of
1446 content. The cataloging service serves the following purposes:

- 1447 • Automates the mapping from the source information model (in this case OWL) to ebRIM. This
1448 hides the complexity of the mapping from the OWL publisher and eliminates the need for any
1449 special UI tools to be provided by the registry implementor for publishing OWL documents.
- 1450 • Selectively converts content into ebRIM compatible metadata when the content is cataloged after
1451 being published. The generated metadata enables the selected content to be used as
1452 parameter(s) in content specific parameterized queries.

1453 This section describes the cataloging service for cataloging OWL content.

1454 An OWL document, when published to an ebXML Registry implementing the OWL Profile, **MUST** be
1455 cataloged as specified in this section using a OWL Content Cataloging Service as defined by [ebRS].

1456 5.1 Invocation Control File

1457 The OWL cataloging service **MAY** optionally support an invocation control file that declaratively specifies
1458 the transforms necessary to catalog published OWL documents.

1459 5.2 Input Metadata

1460 The OWL cataloging service **MUST** be pre-configured to be automatically invoked when the following
1461 types of metadata are published, as defined by the [ebRS] specifications.

1462 These are the only types of metadata that **MAY** describe a OWL document being published:

- 1463 • An `ExtrinsicObject` whose `ObjectType` references the canonical OWL `ClassificationNode`
1464 specified in Section 7. The `ExtrinsicObject` **MUST** have an OWL document as its `RepositoryItem`.
- 1465 • An `ExternalLink` whose `ObjectType` references the canonical OWL `ClassificationNode` specified in
1466 Section 7. In case of `ExternalLink` the OWL document **MUST** be resolvable via a URL described
1467 by the value of the `externalURI` attribute of the `ExternalLink`. Recall that, in the `ExternalLink` case
1468 the OWL document is not stored in the repository.

```
1470 <rim:ExtrinsicObject id="urn:acmeinc:ebxml:registry:3.0:owl">  
1471 ...  
1472 </rim:ExtrinsicObject>
```

1473 Example of ExtrinsicObject Input Metadata

```
1475 <rim:ExternalLink  
1476   id="urn:acmeinc:ebxml:registry:3.0:owl"  
1477   externalURI="http://www.acme.com/owl/ebXMLRegistryService.owl"  
1478   >  
1479 ...  
1480 </rim:ExternalLink>
```

1481 Example of ExternalLink Input Metadata

1482 5.3 Input Content

1483 The OWL cataloging service expects an OWL document as its input content. The input content **MUST** be
1484 processed by the OWL cataloging service regardless of whether it is a `RepositoryItem` for an
1485 `ExtrinsicObject` or whether it is content external to repository that is referenced by an `ExternalLink`.

1486

1487 5.4 Output Metadata

1488 This section describes the metadata produced by the OWL cataloging service produces as output.

1489 5.4.1 owl:Class → rim:ClassificationNode

1490 The OWL Cataloging service MUST automatically produce a rim:ClassificationNode instance for each
1491 owl:class element within the input OWL or its imports, as specified in the owl:Class →
1492 rim:ClassificationNode mapping earlier in this document.

1493 5.4.2 rdf:Property → rim:Association Type HasProperty

1494 The OWL Cataloging service MUST automatically produce an rim:Association instance with
1495 associationType HasProperty for each rdf:Property element within the input OWL or its imports, as
1496 specified in the rdf:Property → rim:Association Type HasProperty mapping earlier in this document.

1497 5.4.3 rdfs:subPropertyOf → rim:Association Type SubPropertyOf

1498 The OWL Cataloging service MUST automatically produce an rim:Association instance with
1499 associationType SubPropertyOf for each rdfs:subPropertyOf element within the input OWL or its imports,
1500 as specified in the rdfs:subPropertyOf → rim:Association Type SubPropertyOf mapping earlier in this
1501 document.

1502 5.4.4 rdfs:subClassOf → rim:Association Type subClassOf

1503 The OWL Cataloging service MUST automatically produce an rim:Association instance with
1504 associationType subClassOf for each rdfs:subClassOf element within the input OWL or its imports, as
1505 specified in the rdfs:subClassOf → rim:Association Type subClassOf mapping earlier in this document.

1506 5.4.5 owl:Individual → rim:ExtrinsicObject

1507 The OWL Cataloging service MUST automatically produce rim:ExtrinsicObject instances for each
1508 owl:Individual element within the input OWL or its imports, as specified in the owl:Individual →
1509 rim:ExtrinsicObject mapping earlier in this document.

1510 5.4.6 owl:equivalentClass, owl:equivalentProperty → rim:Association Type 1511 EquivalentTo

1512 The OWL Cataloging service MUST automatically produce rim:Association instances with
1513 associationType EquivalentTo for each owl:equivalentClass or owl:equivalentProperty element within the
1514 input OWL or its imports, as specified in the owl:equivalentClass, owl:equivalentProperty →
1515 rim:Association Type EquivalentTo mapping earlier in this document.

1516 5.4.7 owl:sameAs → rim:Association Type SameAs

1517 The OWL Cataloging service MUST automatically produce rim:Association instances with
1518 associationType SameAs for each owl:sameAs element within the input OWL or its imports, as specified
1519 in the owl:sameAs → rim:Association Type SameAs mapping earlier in this document.

1520 5.4.8 owl:differentFrom → rim:Association Type DifferentFrom

1521 The OWL Cataloging service MUST automatically produce rim:Association instances with
1522 associationType DifferentFrom for each owl:differentFrom element within the input OWL or its imports, as
1523 specified in the owl:differentFrom → rim:Association Type DifferentFrom mapping earlier in this
1524 document.

1525 5.4.9 owl:AllDifferent → rim:RegistryPackage

1526 The OWL Cataloging service MUST automatically produce rim:RegistryPackage instances for each

1527 owl:AllDifferent element within the input OWL or its imports, as specified in the owl:AllDifferent →
1528 rim:RegistryPackage mapping earlier in this document.

1529 [5.4.10 owl:ObjectProperty → rim:Association Type ObjectProperty](#)

1530 The OWL Cataloging service MUST automatically produce rim:Association instances with
1531 associationType ObjectProperty for each owl:ObjectProperty element within the input OWL or its imports,
1532 as specified in the owl:ObjectProperty → rim:Association Type ObjectProperty mapping earlier in this
1533 document.

1534 [5.4.11 owl:DatatypeProperty → rim:Association Type DatatypeProperty](#)

1535 The OWL Cataloging service MUST automatically produce rim:Association instances with
1536 associationType datatypeProperty for each owl:DatatypeProperty element within the input OWL or its
1537 imports, as specified in the owl:DatatypeProperty → rim:Association Type datatypeProperty mapping
1538 earlier in this document.

1539 [5.4.12 owl:TransitiveProperty → rim:Association Type TransitiveProperty](#)

1540 The OWL Cataloging service MUST automatically produce rim:Association instances with
1541 associationType TransitiveProperty for each owl:TransitiveProperty element within the input OWL or its
1542 imports, as specified in the owl:TransitiveProperty → rim:Association Type TransitiveProperty mapping
1543 earlier in this document.

1544 [5.4.13 owl:inverseOf → rim:Association Type InverseOf](#)

1545 The OWL Cataloging service MUST automatically produce rim:Association instances with
1546 associationType InverseOf for each owl:inverseOf element within the input OWL or its imports, as
1547 specified in the owl:inverseOf → rim:Association Type InverseOf mapping earlier in this document.

1548 [5.4.14 owl:SymmetricProperty → rim:Association Type SymetricProperty](#)

1549 The OWL Cataloging service MUST automatically produce rim:Association instances with
1550 associationType SymetricProperty for each owl:SymetricProperty element within the input OWL or its
1551 imports, as specified in the owl:SymetricProperty → rim:Association Type SymetricProperty mapping
1552 earlier in this document.

1553 [5.4.15 owl:FunctionalProperty → rim:Association Type FunctionalProperty](#)

1554 The OWL Cataloging service MUST automatically produce rim:Association instances with
1555 associationType FunctionalProperty for each owl:FunctionalProperty element within the input OWL or its
1556 imports, as specified in the owl:FunctionalProperty → rim:Association Type FunctionalProperty mapping
1557 earlier in this document.

1558 [5.4.16 owl:InverseFunctionalProperty → rim:Association Type 1559 InverseFunctionalProperty](#)

1560 The OWL Cataloging service MUST automatically produce rim:Association instances with
1561 associationType InverseFunctionalProperty for each owl:InverseFunctionalProperty element within the
1562 input OWL or its imports, as specified in the owl:InverseFunctionalProperty → rim:Association Type
1563 InverseFunctionalProperty mapping earlier in this document.

1564 [5.4.17 owl:minCardinality \(only 0 or 1\)](#)

1565 The OWL Cataloging service MUST automatically add a slot with name minCardinality to the relevant
1566 rim:Association instances for each owl:minCardinality element within the input OWL or its imports, as
1567 specified in section 4.5.1 where how to represent owl:minCardinality is described.

1568 **5.4.18 owl:maxCardinality (only 0 or 1)**

1569 The OWL Cataloging service MUST automatically add a slot with name maxCardinality to the relevant
1570 rim:Association instances for each owl:maxCardinality element within the input OWL or its imports, as
1571 specified in section 4.5.2 where how to represent owl:maxCardinality is described.

1572 **5.4.19 owl:cardinality**

1573 The OWL Cataloging service MUST automatically add a slot with name cardinality to the relevant
1574 rim:Association instances for each owl:cardinality element within the input OWL or its imports, as
1575 specified in section 4.5.3 where how to represent owl:cardinality is described.

1576 **5.4.20 owl:intersectionOf**

1577 The OWL Cataloging service MUST automatically produce a rim:RegistryPackage and a rim:Association
1578 instances with type IntersectionOf for each owl:intersectionOf element within the input OWL or its imports,
1579 as specified in section 4.6 where how to represent owl:intersectionOf is described.

1580 **5.4.21 rdfs:label**

1581 The OWL Cataloging service MUST automatically produce a rim:Name instance for each rdfs:label
1582 element within the input OWL or its imports, as specified in section 4.8.1 where how to represent
1583 rdfs:label is described.

1584 **5.4.22 rdfs:comment**

1585 The OWL Cataloging service MUST automatically produce a rim:Description instance for each
1586 rdfs:comment element within the input OWL or its imports, as specified in section 4.8.2 where how to
1587 represent rdfs:comment is described.

1588 **5.4.23 rdfs:seeAlso**

1589 The OWL Cataloging service MUST automatically produce a rim:ExternalLink and a rim:Association with
1590 type SeeAlso instances for each rdfs:seeAlso element within the input OWL or its imports, as specified in
1591 section 4.8.3 where how to represent rdfs:seeAlso is described.

1592

6 Discovery Profile

1593 The ebXML Registry provides the ability for a user defined parameterized queries to be configured for
1594 each type of content. The queries may be as complex or simple as the discovery use case requires. The
1595 complexity of the parameterized queries may hidden from the registry client by storing them within the
1596 ebXML Registry as instances of the AdhocQuery class, and being invoked by simply providing their
1597 parameters. Query parameters are often pattern strings that may contain wildcard characters '%'
1598 (matches any number of characters) and '_' (matches exactly one character) as described by [ebRS].

1599 An ebXML Registry SHOULD provide a graphical user interface that displays any configured
1600 parameterized query as a form which contains an appropriate field for entering each query parameter.

1601 This chapter defines the queries that MUST be supported by an ebXML Registry implementing the OWL
1602 Profile for processing the semantics provided in the OWL content. An implementation MAY also support
1603 additional discovery queries for OWL content, some of which have already identified in this section.

1604 The queries defined in this chapter are parameterized queries stored in the Registry as instances of the
1605 AdhocQuery type, in the same manner as any other RegistryObject.

1606 In the subsequent section each query is described simply in terms of its supported parameters that serve
1607 as its search criteria. The actual AdhocQuery instances are much more complex in comparison but they
1608 are not exposed to the client making the query. Details on these queries are specified canonically in
1609 section 7.3 .

1610 Some of the queries that are necessary to process the semantics involved in OWL documents requires
1611 SQL recursion mechanism which is available through SQL 99 Standard. Since SQL 92, does not support
1612 recursion mechanism, those queries are stated to be implemented optionally. Additionally for these types
1613 of discovery queries, references to the "stored procedures" are presented in Section 7.3 for the interested
1614 users.

1615 6.1 All SuperProperties Discovery Query

1616 As presented in Section 4.1.3, a new ebXML RIM Association Type called "SubPropertyOf" MUST be
1617 defined to represent rdfs:subPropertyOf in ebRIM. Such a semantic enhancement brings the following
1618 processing need: given a property, it should be possible to retrieve all of its super properties. This requires
1619 a recursion mechanism in SQL queries.

1620 The AllSuperProperties discovery query MAY be implemented by an ebXML Registry implementing this
1621 profile. It allows the discovery of all super properties of a given property instance (Association instance in
1622 ebXML terminology) recursively in a property hierarchy (hierarchy of Association Types) in an ebXML
1623 Registry implementation supporting recursion. The canonical query corresponding to this discovery query
1624 is presented in Section 7.3.1.

1625 6.1.1 Parameter \$propertyName

1626 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1627 value of Associations that have associationType of Property.

1628 6.1.2 Example of All SuperProperties Discovery Query

1629 The following example illustrates how to find all the super properties of a given property having a name
1630 containing "creditCardPayment" if the query is implemented as an AdHoc Query.

1631

```
1632 <rs:RequestSlotList>  
1633   <rim:Slot  
1634     name="urn:oasis:names:tc:ebxml-  
1635   regrep:3.0:rs:AdhocQueryRequest:queryId">  
1636     <rim:ValueList>  
1637       <rim:Value>urn:oasis:names:tc:ebxml-  
1638   regrep:profile:webontology:query:FindAllSuperProperties</rim:Value>  
1639     </rim:ValueList>  
1640   </rim:Slot>
```



```

1641     <rim:Slot name="urn:oasis:names:tc:ebxml-
1642 regrep:rs:AdhocQueryRequest:queryId">
1643         <rim:ValueList>
1644             <rim:Value>urn:oasis:names:tc:ebxml-
1645 regrep:profile:webontology:query:FindAllSuperProperties</rim:Value>
1646         </rim:ValueList>
1647     </rim:Slot>
1648     <rim:Slot name="$propertyName">
1649         <rim:ValueList>
1650             <rim:Value>%creditCardPayment%</rim:Value>
1651         </rim:ValueList>
1652     </rim:Slot>
1653 </rs:RequestSlotList>
1654
1655 <query:ResponseOption returnComposedObjects="true"
1656     returnType="LeafClassWithRepositoryItem"/>
1657
1658 <rim:AdhocQuery id="temporaryId">
1659     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1660 regrep:QueryLanguage:SQL-92">
1661     </rim:QueryExpression>
1662 </rim:AdhocQuery>

```

1663 Example of All SuperProperties Discovery Query

1664 6.2 Immediate SuperClass Discovery Query

1665 The Immediate SuperClass discovery query MUST be implemented by an ebXML Registry implementing
1666 this profile. It allows the discovery of all of the immediate super classes of a given class. The canonical
1667 query corresponding to this discovery query is presented in Section 7.3.2.

1668 6.2.1 Parameter \$className

1669 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1670 value of ClassificationNodes.

1671 6.2.2 Example of Immediate SuperClass Discovery Query

1672 The following example illustrates how to find all the immediate super classes of a given class that have a
1673 name containing the string "AirReservationServices".

```

1674 <rs:RequestSlotList>
1675     <rim:Slot
1676         name="urn:oasis:names:tc:ebxml-
1677 regrep:3.0:rs:AdhocQueryRequest:queryId">
1678         <rim:ValueList>
1679             <rim:Value>urn:oasis:names:tc:ebxml-
1680 regrep:profile:webontology:query:FindImmediateSuperClasses</rim:Value>
1681         </rim:ValueList>
1682     </rim:Slot>
1683     <rim:Slot name="urn:oasis:names:tc:ebxml-
1684 regrep:rs:AdhocQueryRequest:queryId">
1685         <rim:ValueList>
1686             <rim:Value>urn:oasis:names:tc:ebxml-
1687 regrep:profile:webontology:query:FindImmediateSuperClasses</rim:Value>
1688         </rim:ValueList>
1689     </rim:Slot>
1690     <rim:Slot name="$className">
1691         <rim:ValueList>
1692             <rim:Value>%AirReservationServices%</rim:Value>
1693         </rim:ValueList>
1694     </rim:Slot>
1695 </rs:RequestSlotList>
1696
1697 <query:ResponseOption returnComposedObjects="true"
1698     returnType="LeafClassWithRepositoryItem"/>

```

```

1699
1700 <rim:AdhocQuery id="temporaryId">
1701   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1702   regrep:QueryLanguage:SQL-92">
1703     </rim:QueryExpression>
1704 </rim:AdhocQuery>

```

1705 Example of Immediate SuperClass Discovery Query

1706 6.3 Immediate SubClass Discovery Query

1707 The Immediate SubClass discovery query MUST be implemented by an ebXML Registry implementing
1708 this profile. It allows the discovery of all of the immediate subclasses of a given class. The canonical
1709 query corresponding to this discovery query is presented in Section 7.3.3.

1710 6.3.1 Parameter \$className

1711 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1712 value of ClassificationNode.

1713 6.3.2 Example of Immediate SubClass Discovery Query

1714 The following example illustrates how to find all the immediate subclasses of a given class that have a
1715 name containing the string "AirServices" .

```

1716 <rs:RequestSlotList>
1717   <rim:Slot
1718     name="urn:oasis:names:tc:ebxml-
1719   regrep:3.0:rs:AdhocQueryRequest:queryId">
1720     <rim:ValueList>
1721       <rim:Value>urn:oasis:names:tc:ebxml-
1722   regrep:profile:webontology:query:FindImmediateSubClasses</rim:Value>
1723     </rim:ValueList>
1724   </rim:Slot>
1725   <rim:Slot name="urn:oasis:names:tc:ebxml-
1726   regrep:rs:AdhocQueryRequest:queryId">
1727     <rim:ValueList>
1728       <rim:Value>urn:oasis:names:tc:ebxml-
1729   regrep:profile:webontology:query:FindImmediateSubClasses</rim:Value>
1730     </rim:ValueList>
1731   </rim:Slot>
1732   <rim:Slot name="$className">
1733     <rim:ValueList>
1734       <rim:Value>%AirServices%</rim:Value>
1735     </rim:ValueList>
1736   </rim:Slot>
1737 </rs:RequestSlotList>
1738
1739 <query:ResponseOption returnComposedObjects="true"
1740   returnType="LeafClassWithRepositoryItem"/>
1741
1742 <rim:AdhocQuery id="temporaryId">
1743   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1744   regrep:QueryLanguage:SQL-92">
1745     </rim:QueryExpression>
1746 </rim:AdhocQuery>

```

1747 Example of Immediate SubClass Discovery Query

1748 6.4 All SuperClasses Discovery Query

1749 It should be noted that, given a class, finding its immediate subclasses, super classes is necessary but
1750 not sufficient. Given a class, it should be possible to retrieve all of its subclasses, and all of its super
1751 classes. This requires a recursion mechanism in SQL queries.

1752 The All SuperClasses discovery query MAY be implemented by an ebXML Registry implementing this
1753 profile. It allows the discovery of all super classes of a given ClassificationNode recursively in an ebXML
1754 Registry implementation supporting recursion. The canonical query corresponding to this discovery query
1755 is presented in Section 7.3.4.

1756 6.4.1 Parameter \$className

1757 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1758 value of ClassificationNode.

1759 6.4.2 Example of All SuperClasses Discovery Query

1760 The following example illustrates how to find all the super classes of a given class recursively that have a
1761 name containing the string "AirReservationServices" if the query is implemented as an Adhoc Query .

```
1762 <rs:RequestSlotList>  
1763   <rim:Slot  
1764     name="urn:oasis:names:tc:ebxml-  
1765   regrep:3.0:rs:AdhocQueryRequest:queryId">  
1766     <rim:ValueList>  
1767       <rim:Value>urn:oasis:names:tc:ebxml-  
1768   regrep:profile:webontology:query:FindAllSuperClasses</rim:Value>  
1769     </rim:ValueList>  
1770   </rim:Slot>  
1771   <rim:Slot name="urn:oasis:names:tc:ebxml-  
1772   regrep:rs:AdhocQueryRequest:queryId">  
1773     <rim:ValueList>  
1774       <rim:Value>urn:oasis:names:tc:ebxml-  
1775   regrep:profile:webontology:query:FindAllSuperClasses</rim:Value>  
1776     </rim:ValueList>  
1777   </rim:Slot>  
1778   <rim:Slot name="$className">  
1779     <rim:ValueList>  
1780       <rim:Value>%AirReservationServices%</rim:Value>  
1781     </rim:ValueList>  
1782   </rim:Slot>  
1783 </rs:RequestSlotList>  
1784  
1785 <query:ResponseOption returnComposedObjects="true"  
1786   returnType="LeafClassWithRepositoryItem"/>  
1787  
1788 <rim:AdhocQuery id="temporaryId">  
1789   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
1790   regrep:QueryLanguage:SQL-92">  
1791     </rim:QueryExpression>  
1792 </rim:AdhocQuery>
```

1793 Example of All SuperClasses Discovery Query

1794 6.5 All SubClasses Discovery Query

1795 The All SubClasses discovery query MAY be implemented by an ebXML Registry implementing this
1796 profile. It allows the discovery of all subclasses of a given ClassificationNode recursively in an ebXML
1797 Registry implementation supporting recursion. The canonical query corresponding to this discovery query
1798 is presented in Section 7.3.5.

1799 6.5.1 Parameter \$className

1800 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1801 value of ClassificationNode.

1802 6.5.2 Example of All SubClasses Discovery Query

1803 The following example illustrates how to find all the subclasses of a given class recursively that have a

1804 name containing the string "AirServices" , if the query is implemented as an Adhoc Query.

```
1805 <rs:RequestSlotList>
1806   <rim:Slot
1807     name="urn:oasis:names:tc:ebxml-
1808   regrep:3.0:rs:AdhocQueryRequest:queryId">
1809     <rim:ValueList>
1810       <rim:Value>urn:oasis:names:tc:ebxml-
1811   regrep:profile:webontology:query:FindAllSubClasses</rim:Value>
1812     </rim:ValueList>
1813   </rim:Slot>
1814   <rim:Slot name="urn:oasis:names:tc:ebxml-
1815   regrep:rs:AdhocQueryRequest:queryId">
1816     <rim:ValueList>
1817       <rim:Value>urn:oasis:names:tc:ebxml-
1818   regrep:profile:webontology:query:FindAllSubClasses</rim:Value>
1819     </rim:ValueList>
1820   </rim:Slot>
1821   <rim:Slot name="$className">
1822     <rim:ValueList>
1823       <rim:Value>%AirServices%</rim:Value>
1824     </rim:ValueList>
1825   </rim:Slot>
1826 </rs:RequestSlotList>
1827
1828 <query:ResponseOption returnComposedObjects="true"
1829   returnType="LeafClassWithRepositoryItem"/>
1830
1831 <rim:AdhocQuery id="temporaryId">
1832   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1833   regrep:QueryLanguage:SQL-92">
1834     </rim:QueryExpression>
1835 </rim:AdhocQuery>
```

1836 Example of All SubClasses Discovery Query

1837 6.6 EquivalentClasses Discovery Query

1838 The EquivalentClasses discovery query MUST be implemented by an ebXML Registry implementing this
1839 profile. It allows the discovery of all the equivalent classes of a given ClassificationNode.

1840 6.6.1 Parameter \$className

1841 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1842 value of ClassificationNodes.

1843 6.6.2 Example of EquivalentClasses Discovery Query

1844 The following example illustrates how to find all the equivalent classes of a given class that have a name
1845 containing the string "AirServices" .

```
1846 <rs:RequestSlotList>
1847   <rim:Slot
1848     name="urn:oasis:names:tc:ebxml-
1849   regrep:3.0:rs:AdhocQueryRequest:queryId">
1850     <rim:ValueList>
1851       <rim:Value>urn:oasis:names:tc:ebxml-
1852   regrep:profile:webontology:query:FindEquivalentClasses</rim:Value>
1853     </rim:ValueList>
1854   </rim:Slot>
1855   <rim:Slot name="urn:oasis:names:tc:ebxml-
1856   regrep:rs:AdhocQueryRequest:queryId">
1857     <rim:ValueList>
1858       <rim:Value>urn:oasis:names:tc:ebxml-
1859   regrep:profile:webontology:query:FindEquivalentClasses</rim:Value>
1860     </rim:ValueList>
```

```

1861     </rim:Slot>
1862     <rim:Slot name="$className">
1863         <rim:ValueList>
1864             <rim:Value>%AirServices%</rim:Value>
1865         </rim:ValueList>
1866     </rim:Slot>
1867 </rs:RequestSlotList>
1868
1869 <query:ResponseOption returnComposedObjects="true"
1870     returnType="LeafClassWithRepositoryItem"/>
1871
1872 <rim:AdhocQuery id="temporaryId">
1873     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1874     regrep:QueryLanguage:SQL-92">
1875     </rim:QueryExpression>
1876 </rim:AdhocQuery>

```

1877 Example of Equivalent Classes Discovery Query

1878 6.7 EquivalentProperties Discovery Query

1879 The EquivalentProperties discovery query MUST be implemented by an ebXML Registry implementing
1880 this profile. It allows the discovery of all the equivalent properties of a given Association that have
1881 associationType of Property. The canonical query corresponding to this discovery query is presented in
1882 Section 7.3.7.

1883 6.7.1 Parameter \$propertyName

1884 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1885 value of Associations that have associationType of Property

1886 6.7.2 Example of EquivalentProperties Discovery Query

1887 The following example illustrates how to find all the equivalent properties(Association Type) of a given
1888 property (Association Type) that have a name containing the string "paymentMethods" .

```

1889 <rs:RequestSlotList>
1890     <rim:Slot
1891         name="urn:oasis:names:tc:ebxml-
1892     regrep:3.0:rs:AdhocQueryRequest:queryId">
1893         <rim:ValueList>
1894             <rim:Value>urn:oasis:names:tc:ebxml-
1895     regrep:profile:webontology:query:FindEquivalentProperties</rim:Value>
1896         </rim:ValueList>
1897     </rim:Slot>
1898     <rim:Slot name="urn:oasis:names:tc:ebxml-
1899     regrep:rs:AdhocQueryRequest:queryId">
1900         <rim:ValueList>
1901             <rim:Value>urn:oasis:names:tc:ebxml-
1902     regrep:profile:webontology:query:FindEquivalentProperties</rim:Value>
1903         </rim:ValueList>
1904     </rim:Slot>
1905     <rim:Slot name="$propertyName">
1906         <rim:ValueList>
1907             <rim:Value>%paymentMethods%</rim:Value>
1908         </rim:ValueList>
1909     </rim:Slot>
1910 </rs:RequestSlotList>
1911
1912 <query:ResponseOption returnComposedObjects="true"
1913     returnType="LeafClassWithRepositoryItem"/>
1914
1915 <rim:AdhocQuery id="temporaryId">
1916     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1917     regrep:QueryLanguage:SQL-92">

```

1918 </rim:QueryExpression>
1919 </rim:AdhocQuery>

1920 Example of Equivalent Properties Discovery Query

1921 6.8 SameExtrinsicObjects Discovery Query

1922 The SameExtrinsicObjects discovery query MUST be implemented by an ebXML Registry implementing
1923 this profile. It allows the discovery of all the "ExtrinsicObjects" defined to be the same with a given
1924 ExtrinsicObject. The canonical query corresponding to this discovery query is presented in Section 7.3.8.

1925 6.8.1 Parameter \$extrinsicObjectName

1926 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1927 value of ExtrinsicObjects.

1928 6.8.2 Example of SameExtrinsicObjects Discovery Query

1929 The following example illustrates how to find all the ExtrinsicObjects that are defined to be the same as
1930 the ExtrinsicObject that have a name containing the string "MyDocument".

```
1931 <rs:RequestSlotList>  
1932   <rim:Slot  
1933     name="urn:oasis:names:tc:ebxml-  
1934     regrep:3.0:rs:AdhocQueryRequest:queryId">  
1935     <rim:ValueList>  
1936       <rim:Value>urn:oasis:names:tc:ebxml-  
1937       regrep:profile:webontology:query:FindTheSameExtrinsicObjects</rim:Value>  
1938     </rim:ValueList>  
1939   </rim:Slot>  
1940   <rim:Slot name="urn:oasis:names:tc:ebxml-  
1941   regrep:rs:AdhocQueryRequest:queryId">  
1942     <rim:ValueList>  
1943       <rim:Value>urn:oasis:names:tc:ebxml-  
1944       regrep:profile:webontology:query:FindTheSameExtrinsicObjects</rim:Value>  
1945     </rim:ValueList>  
1946   </rim:Slot>  
1947   <rim:Slot name="$extrinsicObjectName">  
1948     <rim:ValueList>  
1949       <rim:Value>%MyDocument%</rim:Value>  
1950     </rim:ValueList>  
1951   </rim:Slot>  
1952 </rs:RequestSlotList>  
1953  
1954 <query:ResponseOption returnComposedObjects="true"  
1955   returnType="LeafClassWithRepositoryItem"/>  
1956  
1957 <rim:AdhocQuery id="temporaryId">  
1958   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
1959   regrep:QueryLanguage:SQL-92">  
1960   </rim:QueryExpression>  
1961 </rim:AdhocQuery>  
1962
```

1963 Example of SameExtrinsicObjects Discovery Query

1964 6.9 DifferentExtrinsicObjects Discovery Query

1965 The DifferentExtrinsicObjects discovery query MUST be implemented by an ebXML Registry
1966 implementing this profile. It allows the discovery of all the "ExtrinsicObjects" defined to be the different
1967 from a given ExtrinsicObject. The canonical query corresponding to this discovery query is presented in
1968 Section 7.3.9.

1969 6.9.1 Parameter \$extrinsicObjectName

1970 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1971 value of ExtrinsicObjects.

1972 6.9.2 Example of DifferentExtrinsicObjects Discovery Query

1973 The following example illustrates how to find all the ExtrinsicObjects that are defined to be different from
1974 the ExtrinsicObject that have a name containing the string "MyDocument" .

```
1975 <rs:RequestSlotList>  
1976   <rim:Slot  
1977     name="urn:oasis:names:tc:ebxml-  
1978   regrep:3.0:rs:AdhocQueryRequest:queryId">  
1979     <rim:ValueList>  
1980       <rim:Value>urn:oasis:names:tc:ebxml-  
1981   regrep:profile:webontology:query:FindDifferentExtrinsicObjects</rim:Value  
1982   >  
1983     </rim:ValueList>  
1984   </rim:Slot>  
1985   <rim:Slot name="urn:oasis:names:tc:ebxml-  
1986   regrep:rs:AdhocQueryRequest:queryId">  
1987     <rim:ValueList>  
1988       <rim:Value>urn:oasis:names:tc:ebxml-  
1989   regrep:profile:webontology:query:FindDifferentExtrinsicObjects</rim:Value  
1990   >  
1991     </rim:ValueList>  
1992   </rim:Slot>  
1993   <rim:Slot name="$extrinsicObjectName">  
1994     <rim:ValueList>  
1995       <rim:Value>%MyDocument%</rim:Value>  
1996     </rim:ValueList>  
1997   </rim:Slot>  
1998 </rs:RequestSlotList>  
1999  
2000 <query:ResponseOption returnComposedObjects="true"  
2001   returnType="LeafClassWithRepositoryItem"/>  
2002  
2003 <rim:AdhocQuery id="temporaryId">  
2004   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
2005   regrep:QueryLanguage:SQL-92">  
2006     </rim:QueryExpression>  
2007 </rim:AdhocQuery>  
2008
```

2009 Example of DifferentExtrinsicObjects Discovery Query

2010 6.10 AllDifferentRegistryObject Discovery Query

2011 The AllDifferentRegistryObjects discovery query MUST be implemented by an ebXML Registry
2012 implementing this profile. Given a RegistryObject, it allows the discovery of all the other member
2013 "RegistryObjects" of a Registry package that are defined to be the different from each other through a
2014 allDifferent slot. The canonical query corresponding to this discovery query is presented in Section
2015 7.3.10.

2016 6.10.1 Parameter \$registryObjectName

2017 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2018 value of RegistryObjects.

2019 6.10.2 Example of AllDifferentRegistryObjects Discovery Query

2020 The following example illustrates how to find all the RegistryObjects that are defined to be different from
2021 the RegistryObject that have a name containing the string "MyDocument" .


```

2022
2023 <rs:RequestSlotList>
2024   <rim:Slot
2025     name="urn:oasis:names:tc:ebxml-
2026 regrep:3.0:rs:AdhocQueryRequest:queryId">
2027     <rim:ValueList>
2028       <rim:Value>urn:oasis:names:tc:ebxml-
2029 regrep:profile:webontology:query:FindAllDifferent</rim:Value>
2030     </rim:ValueList>
2031   </rim:Slot>
2032   <rim:Slot name="urn:oasis:names:tc:ebxml-
2033 regrep:rs:AdhocQueryRequest:queryId">
2034     <rim:ValueList>
2035       <rim:Value>urn:oasis:names:tc:ebxml-
2036 regrep:profile:webontology:query:FindAllDifferent</rim:Value>
2037     </rim:ValueList>
2038   </rim:Slot>
2039   <rim:Slot name="$registryObjectName">
2040     <rim:ValueList>
2041       <rim:Value>%MyDocument%</rim:Value>
2042     </rim:ValueList>
2043   </rim:Slot>
2044 </rs:RequestSlotList>
2045
2046 <query:ResponseOption returnComposedObjects="true"
2047   returnType="LeafClassWithRepositoryItem"/>
2048
2049 <rim:AdhocQuery id="temporaryId">
2050   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2051 regrep:QueryLanguage:SQL-92">
2052   </rim:QueryExpression>
2053 </rim:AdhocQuery>

```

2054 Example of AllDifferentRegistryObjects Discovery Query

2055 6.11 ObjectProperties Discovery Query

2056 The ObjectProperties discovery query MUST be implemented by an ebXML Registry implementing this
2057 profile. It allows the discovery of all of the objectProperties of a given classification node. The canonical
2058 query corresponding to this discovery query is presented in Section 7.3.11.

2059 6.11.1 Parameter \$className

2060 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2061 value of ClassificationNodes.

2062 6.11.2 Example of ObjectProperties Discovery Query

2063 The following example illustrates how to find all the object properties of a given classification node having
2064 a name containing "AirServices" .

```

2065
2066 <rs:RequestSlotList>
2067   <rim:Slot
2068     name="urn:oasis:names:tc:ebxml-
2069 regrep:3.0:rs:AdhocQueryRequest:queryId">
2070     <rim:ValueList>
2071       <rim:Value>urn:oasis:names:tc:ebxml-
2072 regrep:profile:webontology:query:FindObjectProperties</rim:Value>
2073     </rim:ValueList>
2074   </rim:Slot>
2075   <rim:Slot name="urn:oasis:names:tc:ebxml-
2076 regrep:rs:AdhocQueryRequest:queryId">
2077     <rim:ValueList>

```



```

2078         <rim:Value>urn:oasis:names:tc:ebxml-
2079 regrep:profile:webontology:query:FindObjectProperties</rim:Value>
2080     </rim:ValueList>
2081 </rim:Slot>
2082 <rim:Slot name="$className">
2083     <rim:ValueList>
2084         <rim:Value>%AirServices%</rim:Value>
2085     </rim:ValueList>
2086 </rim:Slot>
2087 </rs:RequestSlotList>
2088
2089 <query:ResponseOption returnComposedObjects="true"
2090     returnType="LeafClassWithRepositoryItem"/>
2091
2092 <rim:AdhocQuery id="temporaryId">
2093     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2094 regrep:QueryLanguage:SQL-92">
2095     </rim:QueryExpression>
2096 </rim:AdhocQuery>

```

2097 Example of ObjectProperties Discovery Query

2098 6.12 ImmediateInheritedObjectProperties Discovery Query

2099 The ImmediateInheritedObjectProperties discovery query MUST be implemented by an ebXML Registry
2100 implementing this profile. It allows the discovery of all of the objectProperties of a given classification node
2101 including the ones inherited from its immediate super classes. The canonical query corresponding to this
2102 discovery query is presented in Section 7.3.12.

2103 6.12.1 Parameter \$className

2104 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2105 value of ClassificationNodes.

2106 6.12.2 Example of ImmediateInheritedObjectProperties Discovery Query

2107 The following example illustrates how to find all the object properties of a given classification node having
2108 a name containing "AirServices" including the ones inherited from its immediate super classes.

```

2109
2110 <rs:RequestSlotList>
2111     <rim:Slot
2112         name="urn:oasis:names:tc:ebxml-
2113 regrep:3.0:rs:AdhocQueryRequest:queryId">
2114         <rim:ValueList>
2115             <rim:Value>urn:oasis:names:tc:ebxml-
2116 regrep:profile:webontology:query:FindImmediateInheritedObjectProperties</
2117 rim:Value>
2118         </rim:ValueList>
2119     </rim:Slot>
2120     <rim:Slot name="urn:oasis:names:tc:ebxml-
2121 regrep:rs:AdhocQueryRequest:queryId">
2122         <rim:ValueList>
2123             <rim:Value>urn:oasis:names:tc:ebxml-
2124 regrep:profile:webontology:query:FindImmediateInheritedObjectProperties</
2125 rim:Value>
2126         </rim:ValueList>
2127     </rim:Slot>
2128     <rim:Slot name="$className">
2129         <rim:ValueList>
2130             <rim:Value>%AirServices%</rim:Value>
2131         </rim:ValueList>
2132     </rim:Slot>
2133 </rs:RequestSlotList>
2134

```

```

2135 <query:ResponseOption returnComposedObjects="true"
2136     returnType="LeafClassWithRepositoryItem"/>
2137
2138 <rim:AdhocQuery id="temporaryId">
2139   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2140   regrep:QueryLanguage:SQL-92">
2141     </rim:QueryExpression>
2142   </rim:AdhocQuery>

```

2143 Example of ImmediateInheritedObjectProperties Discovery Query

2144 6.13 AllInheritedObjectProperties Discovery Query

2145 It should be noted that, given a class, finding the object properties inherited from immediate super classes
 2146 is necessary but not sufficient. Given a class, it should be possible to retrieve all of the object properties
 2147 inherited from its super classes. This requires a recursion mechanism in SQL queries.

2148 The AllInheritedObjectProperties discovery query MAY be implemented by an ebXML Registry
 2149 implementing this profile. It allows the discovery of all inherited ObjectProperties recursively of a given
 2150 ClassificationNode in a ClassificationScheme in an ebXML Registry implementation supporting recursion.

2151 The canonical query corresponding to this discovery query is presented in Section 7.3.13.

2152 6.13.1 Parameter \$className

2153 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
 2154 value of ClassificationNodes.

2155 6.13.2 Example of AllInheritedObjectProperties Discovery Query

2156 The following example illustrates how to find all the object properties of a given classification node having
 2157 a name containing "AirReservationServices" including the ones inherited from all of its super classes
 2158 recursively, if the query is implemented as an Adhoc Query.

```

2159
2160 <rs:RequestSlotList>
2161   <rim:Slot
2162     name="urn:oasis:names:tc:ebxml-
2163     regrep:3.0:rs:AdhocQueryRequest:queryId">
2164     <rim:ValueList>
2165       <rim:Value>urn:oasis:names:tc:ebxml-
2166       regrep:profile:webontology:query:FindAllInheritedObjectProperties</rim:Va
2167       lue>
2168     </rim:ValueList>
2169   </rim:Slot>
2170   <rim:Slot name="urn:oasis:names:tc:ebxml-
2171   regrep:rs:AdhocQueryRequest:queryId">
2172     <rim:ValueList>
2173       <rim:Value>urn:oasis:names:tc:ebxml-
2174       regrep:profile:webontology:query:FindAll
2175       InheritedObjectProperties</rim:Value>
2176     </rim:ValueList>
2177   </rim:Slot>
2178   <rim:Slot name="$className">
2179     <rim:ValueList>
2180       <rim:Value>%AirReservationServices%</rim:Value>
2181     </rim:ValueList>
2182   </rim:Slot>
2183 </rs:RequestSlotList>
2184
2185 <query:ResponseOption returnComposedObjects="true"
2186     returnType="LeafClassWithRepositoryItem"/>
2187
2188 <rim:AdhocQuery id="temporaryId">
2189   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2190   regrep:QueryLanguage:SQL-92">

```

```
2191 </rim:QueryExpression>
2192 </rim:AdhocQuery>
```

2193 Example of AllInheritedObjectProperties Discovery Query

2194 6.14 DatatypeProperties Discovery Query

2195 The DatatypeProperties discovery query MUST be implemented by an ebXML Registry implementing this
2196 profile. It allows the discovery of all of the datatypeProperties of a given classification node. The
2197 canonical query corresponding to this discovery query is presented in Section 7.3.14.

2198 6.14.1 Parameter \$className

2199 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2200 value of ClassificationNodes.

2201 6.14.2 Example of DatatypeProperties Discovery Query

2202 The following example illustrates how to find all the datatype properties of a given classification node
2203 having a name containing "AirReservationServices" .

2204

```
2205 <rs:RequestSlotList>
2206   <rim:Slot
2207     name="urn:oasis:names:tc:ebxml-
2208     regrep:3.0:rs:AdhocQueryRequest:queryId">
2209     <rim:ValueList>
2210       <rim:Value>urn:oasis:names:tc:ebxml-
2211       regrep:profile:webontology:query:FindDatatypeProperties</rim:Value>
2212     </rim:ValueList>
2213   </rim:Slot>
2214   <rim:Slot name="urn:oasis:names:tc:ebxml-
2215   regrep:rs:AdhocQueryRequest:queryId">
2216     <rim:ValueList>
2217       <rim:Value>urn:oasis:names:tc:ebxml-
2218       regrep:profile:webontology:query:FindDatatypeProperties</rim:Value>
2219     </rim:ValueList>
2220   </rim:Slot>
2221   <rim:Slot name="$className">
2222     <rim:ValueList>
2223       <rim:Value>%AirReservationServices%</rim:Value>
2224     </rim:ValueList>
2225   </rim:Slot>
2226 </rs:RequestSlotList>
2227
2228 <query:ResponseOption returnComposedObjects="true"
2229   returnType="LeafClassWithRepositoryItem"/>
2230
2231 <rim:AdhocQuery id="temporaryId">
2232   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2233   regrep:QueryLanguage:SQL-92">
2234     </rim:QueryExpression>
2235 </rim:AdhocQuery>
```

2236 Example of DatatypeProperties Discovery Query

2237 6.15 AllInheritedDatatypeProperties Discovery Query

2238 It should be noted that, given a class, finding the datatype properties inherited from immediate super
2239 classes is necessary but not sufficient. Given a class, it should be possible to retrieve all of the datatype
2240 properties inherited from its super classes. This requires a recursion mechanism in SQL queries.

2241 The AllInheritedDatatypeProperties discovery query MAY be implemented by an ebXML Registry
2242 implementing this profile. It allows the discovery of all inherited DatatypeProperties recursively of a given
2243 ClassificationNode in a ClassificationScheme in an ebXML Registry implementation supporting recursion.

2244 The canonical query corresponding to this discovery query is presented in Section 7.3.15.

2245 6.15.1 Parameter \$className

2246 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2247 value of ClassificationNodes.

2248 6.15.2 Example of AllInheritedDatatypeProperties Discovery Query

2249 The following example illustrates how to find all the datatype properties of a given classification node
2250 having a name containing "AirReservationServices" including the ones inherited from all of its super
2251 classes recursively, if the query is implemented as an Adhoc Query.

2252

```
2253 <rs:RequestSlotList>
2254   <rim:Slot
2255     name="urn:oasis:names:tc:ebxml-
2256   regrep:3.0:rs:AdhocQueryRequest:queryId">
2257     <rim:ValueList>
2258       <rim:Value>urn:oasis:names:tc:ebxml-
2259   regrep:profile:webontology:query:FindAllInheritedDatatypeProperties</rim:
2260   Value>
2261     </rim:ValueList>
2262   </rim:Slot>
2263   <rim:Slot name="urn:oasis:names:tc:ebxml-
2264   regrep:rs:AdhocQueryRequest:queryId">
2265     <rim:ValueList>
2266       <rim:Value>urn:oasis:names:tc:ebxml-
2267   regrep:profile:webontology:query:FindAllInheritedDatatypeProperties</rim:
2268   Value>
2269     </rim:ValueList>
2270   </rim:Slot>
2271   <rim:Slot name="$className">
2272     <rim:ValueList>
2273       <rim:Value>%AirReservationServices %</rim:Value>
2274     </rim:ValueList>
2275   </rim:Slot>
2276 </rs:RequestSlotList>
2277
2278 <query:ResponseOption returnComposedObjects="true"
2279   returnType="LeafClassWithRepositoryItem"/>
2280
2281 <rim:AdhocQuery id="temporaryId">
2282   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2283   regrep:QueryLanguage:SQL-92">
2284     </rim:QueryExpression>
2285 </rim:AdhocQuery>
```

2286 Example of AllInheritedDatatypeProperties Discovery Query

2287 6.16 TransitiveRelationships Discovery Query

2288 To make any use of the transitive property in ebXML registries, coding is necessary to find out the implied
2289 information. The TransitiveRelationships discovery query MUST be implemented by an ebXML Registry
2290 implementing this profile to handle this semantics.

2291 Given a class which is a source of a transitive property, this discovery query retrieves not only the target
2292 objects of a given transitive property, but if the target objects have the same property, it retrieves their
2293 target objects too. The canonical query corresponding to this discovery query is presented in Section
2294 7.3.16.

2295 6.16.1 Parameter \$className

2296 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2297 value of ClassificationNodes.

2298 6.16.2 Parameter \$propertyName

2299 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2300 value of Associations that have associationType of Property

2301 6.16.3 Example of TransitiveRelationships Discovery Query

2302 The following example illustrates how to retrieve all the target objects of the "succeeds" property of the
2303 "AirReservationServices" including the target objects implied by a transitive property relationship.

2304

```
2305 <rs:RequestSlotList>
2306   <rim:Slot
2307     name="urn:oasis:names:tc:ebxml-
2308   regrep:3.0:rs:AdhocQueryRequest:queryId">
2309     <rim:ValueList>
2310       <rim:Value>urn:oasis:names:tc:ebxml-
2311   regrep:profile:webontology:query:FindTransitiveRelationships</rim:Value>
2312     </rim:ValueList>
2313   </rim:Slot>
2314   <rim:Slot name="urn:oasis:names:tc:ebxml-
2315   regrep:rs:AdhocQueryRequest:queryId">
2316     <rim:ValueList>
2317       <rim:Value>urn:oasis:names:tc:ebxml-
2318   regrep:profile:webontology:query:FindTransitiveRelationships</rim:Value>
2319     </rim:ValueList>
2320   </rim:Slot>
2321   <rim:Slot name="$className">
2322     <rim:ValueList>
2323       <rim:Value>%AirReservationServices%</rim:Value>
2324     </rim:ValueList>
2325   </rim:Slot>
2326   <rim:Slot name="$propertyName">
2327     <rim:ValueList>
2328       <rim:Value>%succeeds%</rim:Value>
2329     </rim:ValueList>
2330   </rim:Slot>
2331 </rs:RequestSlotList>
2332
2333 <query:ResponseOption returnComposedObjects="true"
2334   returnType="LeafClassWithRepositoryItem"/>
2335
2336 <rim:AdhocQuery id="temporaryId">
2337   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2338   regrep:QueryLanguage:SQL-92">
2339     </rim:QueryExpression>
2340 </rim:AdhocQuery>
```

2341 Example of TransitiveRelationships Discovery Query

2342 6.17 TargetObjects Discovery Query

2343 The TargetObjects discovery query MUST be implemented by an ebXML Registry implementing this
2344 profile. It allows the discovery of the targetObjects from the Registry, given a Classification Node
2345 (sourceObject) and a property name (Association Type). The canonical query corresponding to this
2346 discovery query is presented in Section 7.3.17.

2347 6.17.1 Parameter \$className

2348 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2349 value of ClassificationNodes.

2350 6.17.2 Parameter \$propertyName

2351 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2352 value of Associations that have associationType of Property.

2353 6.17.3 Example of TargetObjects Discovery Query

2354 The following example illustrates how to retrieve all the target objects of the "paymentMethod" property of
2355 the "AirReservationServices".

2356

```
2357 <rs:RequestSlotList>
2358   <rim:Slot
2359     name="urn:oasis:names:tc:ebxml-
2360   regrep:3.0:rs:AdhocQueryRequest:queryId">
2361     <rim:ValueList>
2362       <rim:Value>urn:oasis:names:tc:ebxml-
2363   regrep:profile:webontology:query:FindTargetObjects</rim:Value>
2364     </rim:ValueList>
2365   </rim:Slot>
2366   <rim:Slot name="urn:oasis:names:tc:ebxml-
2367   regrep:rs:AdhocQueryRequest:queryId">
2368     <rim:ValueList>
2369       <rim:Value>urn:oasis:names:tc:ebxml-
2370   regrep:profile:webontology:query:FindTargetObjects</rim:Value>
2371     </rim:ValueList>
2372   </rim:Slot>
2373   <rim:Slot name="$className">
2374     <rim:ValueList>
2375       <rim:Value>%AirReservationServices%</rim:Value>
2376     </rim:ValueList>
2377   </rim:Slot>
2378   <rim:Slot name="$propertyName">
2379     <rim:ValueList>
2380       <rim:Value>%paymentMethod%</rim:Value>
2381     </rim:ValueList>
2382   </rim:Slot>
2383 </rs:RequestSlotList>
2384
2385 <query:ResponseOption returnComposedObjects="true"
2386   returnType="LeafClassWithRepositoryItem"/>
2387
2388 <rim:AdhocQuery id="temporaryId">
2389   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2390   regrep:QueryLanguage:SQL-92">
2391     </rim:QueryExpression>
2392 </rim:AdhocQuery>
```

2393 Example of TargetObjects Discovery Query

2394

2395 6.18 TargetObjectsInverseOf Discovery Query

2396 The TargetObjectsInverseOf discovery query MUST be implemented by an ebXML Registry implementing
2397 this profile. Given a Classification Node (sourceObject) and a property name (Association Type), this
2398 query retrieves the source objects of the properties which are stated to be inverseOf the property name
2399 given as a parameter, and considering the Classification Node name as the targetObject of these
2400 properties. The canonical query corresponding to this discovery query is presented in Section 7.3.18.

2401 6.18.1 Parameter \$className

2402 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2403 value of ClassificationNodes.

2404 6.18.2 Parameter \$propertyName

2405 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2406 value of Associations that have associationType of Property.

2407 6.18.3 Example of TargetObjectsInverseOf Discovery Query

2408 The following example illustrates how to retrieve all the source objects of the properties which are stated
2409 to the the inverseOf the property "succeeds", considering the "AirReservationServices" as the target object
2410 of these properties.

2411

```
2412 <rs:RequestSlotList>  
2413   <rim:Slot  
2414     name="urn:oasis:names:tc:ebxml-  
2415   regrep:3.0:rs:AdhocQueryRequest:queryId">  
2416     <rim:ValueList>  
2417       <rim:Value>urn:oasis:names:tc:ebxml-  
2418   regrep:profile:webontology:query:FindTOinverseOf</rim:Value>  
2419     </rim:ValueList>  
2420   </rim:Slot>  
2421   <rim:Slot name="urn:oasis:names:tc:ebxml-  
2422   regrep:rs:AdhocQueryRequest:queryId">  
2423     <rim:ValueList>  
2424       <rim:Value>urn:oasis:names:tc:ebxml-  
2425   regrep:profile:webontology:query:FindTOinverseOf</rim:Value>  
2426     </rim:ValueList>  
2427   </rim:Slot>  
2428   <rim:Slot name="$className">  
2429     <rim:ValueList>  
2430       <rim:Value>%AirReservationServices%</rim:Value>  
2431     </rim:ValueList>  
2432   </rim:Slot>  
2433   <rim:Slot name="$propertyName">  
2434     <rim:ValueList>  
2435       <rim:Value>%succeeds%</rim:Value>  
2436     </rim:ValueList>  
2437   </rim:Slot>  
2438 </rs:RequestSlotList>  
2439  
2440 <query:ResponseOption returnComposedObjects="true"  
2441   returnType="LeafClassWithRepositoryItem"/>  
2442  
2443 <rim:AdhocQuery id="temporaryId">  
2444   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
2445   regrep:QueryLanguage:SQL-92">  
2446     </rim:QueryExpression>  
2447 </rim:AdhocQuery>
```

2448 Example of TargetObjectsInverseOf Discovery Query

2449

2450 6.19 InverseRanges Discovery Query

2451 The InverseRanges discovery query MUST be implemented by an ebXML Registry implementing this
2452 profile to handle this semantics. Given a Classification Node (sourceObject) and a property name
2453 (Association Type), this query retrieves not only the target objects of this property, but also the source
2454 objects of the properties which are stated to be inverseOf the property name given as a parameter, and
2455 considering the Classification Node name as the targetObject of these properties. This query can be

2456 thought as the union of the queries presented in Sections 6.17 and 6.18. The canonical query
2457 corresponding to this discovery query is presented in Section 7.3.19.

2458 6.19.1 Parameter \$className

2459 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2460 value of ClassificationNodes.

2461 6.19.2 Parameter \$propertyName

2462 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2463 value of Associations that have associationType of Property

2464 6.19.3 Example of InverseRanges Discovery Query

2465 Consider, for example, the "succeeds" property defined in Section 4.3.3. To denote that a certain Web
2466 service instance precedes another during execution, we may define the "precedes" property as an inverse
2467 of the "succeeds" property as follows:

2468

```
2469 <owl:ObjectProperty rdf:ID="precedes">  
2470   <owl:inverseOf rdf:resource="#succeeds" />  
2471 </owl:ObjectProperty>
```

2472 Example owl:inverseOf Property

2473 Assume that we want to find all the Web services which can succeed a given Web service. In such a
2474 case, we need not only find all the Web services which succeeds this given Web service, that is the target
2475 objects of "succeeds" Association instance, but we also need to find all the sourceObjects of the
2476 "precedes" Association instance since "precedes" is declared to be the "inverseOf" succeeds Association
2477 instance.

2478 The following example illustrates how to retrieve all the services that "succeeds" "AirReservationServices"
2479 by also making use of its "precedes" property.

2480

```
2481 <rs:RequestSlotList>  
2482   <rim:Slot  
2483     name="urn:oasis:names:tc:ebxml-  
2484     regrep:3.0:rs:AdhocQueryRequest:queryId">  
2485     <rim:ValueList>  
2486       <rim:Value>urn:oasis:names:tc:ebxml-  
2487       regrep:profile:webontology:query:FindInverseRanges</rim:Value>  
2488     </rim:ValueList>  
2489   </rim:Slot>  
2490   <rim:Slot name="urn:oasis:names:tc:ebxml-  
2491   regrep:rs:AdhocQueryRequest:queryId">  
2492     <rim:ValueList>  
2493       <rim:Value>urn:oasis:names:tc:ebxml-  
2494       regrep:profile:webontology:query:FindInverseRanges</rim:Value>  
2495     </rim:ValueList>  
2496   </rim:Slot>  
2497   <rim:Slot name="$className">  
2498     <rim:ValueList>  
2499       <rim:Value>%AirReservationServices%</rim:Value>  
2500     </rim:ValueList>  
2501   </rim:Slot>  
2502   <rim:Slot name="$propertyName">  
2503     <rim:ValueList>  
2504       <rim:Value>%succeeds%</rim:Value>  
2505     </rim:ValueList>  
2506   </rim:Slot>  
2507 </rs:RequestSlotList>  
2508  
2509 <query:ResponseOption returnComposedObjects="true"
```



```

2510     returnType="LeafClassWithRepositoryItem"/>
2511
2512 <rim:AdhocQuery id="temporaryId">
2513   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2514   regrep:QueryLanguage:SQL-92">
2515     </rim:QueryExpression>
2516   </rim:AdhocQuery>

```

2517 Example of InverseRanges Discovery Query

2518 6.20 SymmetricProperties Discovery Query

2519 The SymmetricProperties discovery query MUST be implemented by an ebXML Registry implementing
2520 this profile. It allows the discovery of all of the Symmetric Properties of a given classification node. The
2521 canonical query corresponding to this discovery query is presented in Section 7.3.20.

2522 6.20.1 Parameter \$className

2523 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2524 value of ClassificationNodes.

2525 6.20.2 Example of SymmetricProperties Discovery Query

2526 The following example illustrates how to find all the symmetric properties of a given classification node
2527 having a name containing "AirReservationServices" .

```

2528
2529 <rs:RequestSlotList>
2530   <rim:Slot
2531     name="urn:oasis:names:tc:ebxml-
2532     regrep:3.0:rs:AdhocQueryRequest:queryId">
2533     <rim:ValueList>
2534       <rim:Value>urn:oasis:names:tc:ebxml-
2535       regrep:profile:webontology:query:FindSymmetricProperties</rim:Value>
2536     </rim:ValueList>
2537   </rim:Slot>
2538   <rim:Slot name="urn:oasis:names:tc:ebxml-
2539   regrep:rs:AdhocQueryRequest:queryId">
2540     <rim:ValueList>
2541       <rim:Value>urn:oasis:names:tc:ebxml-
2542       regrep:profile:webontology:query:FindSymmetricProperties</rim:Value>
2543     </rim:ValueList>
2544   </rim:Slot>
2545   <rim:Slot name="$className">
2546     <rim:ValueList>
2547       <rim:Value>%AirReservationServices%</rim:Value>
2548     </rim:ValueList>
2549   </rim:Slot>
2550 </rs:RequestSlotList>
2551
2552 <query:ResponseOption returnComposedObjects="true"
2553   returnType="LeafClassWithRepositoryItem"/>
2554
2555 <rim:AdhocQuery id="temporaryId">
2556   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2557   regrep:QueryLanguage:SQL-92">
2558     </rim:QueryExpression>
2559   </rim:AdhocQuery>

```

2560 Example of SymmetricProperties Discovery Query

2561 6.21 FunctionalProperties Discovery Query

2562 The FunctionalProperties discovery query MUST be implemented by an ebXML Registry implementing
2563 this profile. It allows the discovery of all of the Functional Properties of a given classification node. The

2564 canonical query corresponding to this discovery query is presented in Section 7.3.21.

2565 6.21.1 Parameter \$className

2566 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2567 value of ClassificationNodes.

2568 6.21.2 Example of FunctionalProperties Discovery Query

2569 The following example illustrates how to find all the functional properties of a given classification node
2570 having a name containing "AirReservationServices" .

2571

```
2572 <rs:RequestSlotList>
2573   <rim:Slot
2574     name="urn:oasis:names:tc:ebxml-
2575   regrep:3.0:rs:AdhocQueryRequest:queryId">
2576     <rim:ValueList>
2577       <rim:Value>urn:oasis:names:tc:ebxml-
2578   regrep:profile:webontology:query:FindFunctionalProperties</rim:Value>
2579     </rim:ValueList>
2580   </rim:Slot>
2581   <rim:Slot name="urn:oasis:names:tc:ebxml-
2582   regrep:rs:AdhocQueryRequest:queryId">
2583     <rim:ValueList>
2584       <rim:Value>urn:oasis:names:tc:ebxml-
2585   regrep:profile:webontology:query:FindFunctionalProperties</rim:Value>
2586     </rim:ValueList>
2587   </rim:Slot>
2588   <rim:Slot name="$className">
2589     <rim:ValueList>
2590       <rim:Value>%AirReservationServices%</rim:Value>
2591     </rim:ValueList>
2592   </rim:Slot>
2593 </rs:RequestSlotList>
2594
2595 <query:ResponseOption returnComposedObjects="true"
2596   returnType="LeafClassWithRepositoryItem"/>
2597
2598 <rim:AdhocQuery id="temporaryId">
2599   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2600   regrep:QueryLanguage:SQL-92">
2601     </rim:QueryExpression>
2602 </rim:AdhocQuery>
```

2603

Example of Functional Properties Discovery Query

2604 6.22 InverseFunctionalProperties Discovery Query

2605 The InverseFunctionalProperties discovery query MUST be implemented by an ebXML Registry
2606 implementing this profile. It allows the discovery of all of the Inverse Functional Properties of a given
2607 classification node. The canonical query corresponding to this discovery query is presented in Section
2608 7.3.22.

2609 6.22.1 Parameter \$className

2610 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2611 value of ClassificationNodes.

2612 6.22.2 Example of InverseFunctionalProperties Discovery Query

2613 The following example illustrates how to find all the inverse functional properties of a given classification
2614 node having a name containing "AirReservationServices" .

2615

```
2616 <rs:RequestSlotList>
2617   <rim:Slot
2618     name="urn:oasis:names:tc:ebxml-
2619   regrep:3.0:rs:AdhocQueryRequest:queryId">
2620     <rim:ValueList>
2621       <rim:Value>urn:oasis:names:tc:ebxml-
2622   regrep:profile:webontology:query:FindInverseFunctionalProperties</rim:Val
2623   ue>
2624     </rim:ValueList>
2625   </rim:Slot>
2626   <rim:Slot name="urn:oasis:names:tc:ebxml-
2627   regrep:rs:AdhocQueryRequest:queryId">
2628     <rim:ValueList>
2629       <rim:Value>urn:oasis:names:tc:ebxml-
2630   regrep:profile:webontology:query:FindInverseFunctionalProperties</rim:Val
2631   ue>
2632     </rim:ValueList>
2633   </rim:Slot>
2634   <rim:Slot name="$className">
2635     <rim:ValueList>
2636       <rim:Value>%AirReservationServices%</rim:Value>
2637     </rim:ValueList>
2638   </rim:Slot>
2639 </rs:RequestSlotList>
2640
2641 <query:ResponseOption returnComposedObjects="true"
2642   returnType="LeafClassWithRepositoryItem"/>
2643
2644 <rim:AdhocQuery id="temporaryId">
2645   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2646   regrep:QueryLanguage:SQL-92">
2647     </rim:QueryExpression>
2648 </rim:AdhocQuery>
```

2649

Example of InverseFunctional Properties Discovery Query

2650 6.23 Instances Discovery Query

2651 When an intersection definition is used to create a complex class (a new ClassificationNode) in RIM as
2652 described in Section 4.6, it becomes possible to infer that the objects (instances) classified by all of the
2653 classes (ClassificationNodes) specified in the Intersection definition, are also classified by this complex
2654 class.

2655 The Instances discovery query MUST be implemented by an ebXML Registry implementing this profile. It
2656 allows the discovery of all of the direct instances of a given classification node and if it is a complex class
2657 which is an intersection two classes, it also allows to retrieve the intersection of the instances of both of
2658 the classes involved in the intersection definition. The canonical query corresponding to this discovery
2659 query is presented in Section 7.3.23.

2660 6.23.1 Parameter \$className

2661 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2662 value of ClassificationNodes.

2663 6.23.2 Example of Instances Discovery Query

2664 Consider the "AirReservationServices" definition presented in Section 4.6. The following example
2665 illustrates how to find all the direct instances of the "AirReservationServices" and also the instances
2666 classified by both "AirServices" and also the "ReservationServices".

2667

```
2668 <rs:RequestSlotList>
2669   <rim:Slot
```

```

2670     name="urn:oasis:names:tc:ebxml-
2671 regrep:3.0:rs:AdhocQueryRequest:queryId">
2672     <rim:ValueList>
2673         <rim:Value>urn:oasis:names:tc:ebxml-
2674 regrep:profile:webontology:query:FindInstances</rim:Value>
2675     </rim:ValueList>
2676 </rim:Slot>
2677 <rim:Slot name="urn:oasis:names:tc:ebxml-
2678 regrep:rs:AdhocQueryRequest:queryId">
2679     <rim:ValueList>
2680         <rim:Value>urn:oasis:names:tc:ebxml-
2681 regrep:profile:webontology:query:FindInstances</rim:Value>
2682     </rim:ValueList>
2683 </rim:Slot>
2684 <rim:Slot name="$className">
2685     <rim:ValueList>
2686         <rim:Value>%AirReservationServices%</rim:Value>
2687     </rim:ValueList>
2688 </rim:Slot>
2689 </rs:RequestSlotList>
2690
2691 <query:ResponseOption returnComposedObjects="true"
2692     returnType="LeafClassWithRepositoryItem"/>
2693
2694 <rim:AdhocQuery id="temporaryId">
2695     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2696 regrep:QueryLanguage:SQL-92">
2697     </rim:QueryExpression>
2698 </rim:AdhocQuery>

```

Example of Instances Discovery Query

7 Canonical Metadata Definitions

2700

2701 This chapter specifies the canonical metadata defined by this profile.

7.1 ObjectType Extensions

2702

2703 The following new extensions to the canonical ObjectType ClassificationScheme are described by this
2704 profile:

2705

```
2706 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2707 regrep:ObjectType:RegistryObject:ExtrinsicObject"  
2708 lid="urn:oasis:names:tc:ebxml-  
2709 regrep:profile:webontology:ObjectType:RegistryObject:ExtrinsicObject:OWL"  
2710 code="OWL" id="urn:oasis:names:tc:ebxml-  
2711 regrep:profile:webontology:ObjectType:RegistryObject:ExtrinsicObject:OWL"  
2712 >  
2713 <rim:Name>  
2714 <rim:LocalizedString charset="UTF-8" value="OWL"/>  
2715 </rim:Name>  
2716 </rim:ClassificationNode>
```

7.2 AssociationType Extensions

2717

2718 The following new extensions to the AssociationType ClassificationScheme are described by this profile:

2719

```
2720 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2721 regrep:classificationScheme:AssociationType"  
2722 lid="urn:oasis:names:tc:ebxml-  
2723 regrep:profile:webontology:AssociationType:OWL" code="OWL"  
2724 id="urn:oasis:names:tc:ebxml-  
2725 regrep:profile:webontology:AssociationType:OWL">  
2726 <rim:Name>  
2727 <rim:LocalizedString charset="UTF-8" value="OWL"/>  
2728 </rim:Name>  
2729 </rim:ClassificationNode>  
2730 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2731 regrep:profile:webontology:AssociationType:OWL"  
2732 lid="urn:oasis:names:tc:ebxml-  
2733 regrep:profile:webontology:AssociationType:OWL:ObjectProperty"  
2734 code="ObjectProperty" id="urn:oasis:names:tc:ebxml-  
2735 regrep:profile:webontology:AssociationType:OWL:ObjectProperty">  
2736 <rim:Name>  
2737 <rim:LocalizedString charset="UTF-8"  
2738 value="ObjectProperty"/>  
2739 </rim:Name>  
2740 </rim:ClassificationNode>  
2741 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2742 regrep:profile:webontology:AssociationType:OWL"  
2743 lid="urn:oasis:names:tc:ebxml-  
2744 regrep:profile:webontology:AssociationType:OWL:HasProperty"  
2745 code="Property" id="urn:oasis:names:tc:ebxml-  
2746 regrep:profile:webontology:AssociationType:OWL:HasProperty">  
2747 <rim:Name>  
2748 <rim:LocalizedString charset="UTF-8" value="Property"/>  
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:SubPropertyOf"  
2755 code="SubPropertyOf" id="urn:oasis:names:tc:ebxml-  
2756 regrep:profile:webontology:AssociationType:OWL:SubPropertyOf">  
2757 <rim:Name>
```

```

2758         <rim:LocalizedString charset="UTF-8" value="SubPropertyOf"/>
2759         </rim:Name>
2760     </rim:ClassificationNode>
2761     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2762     regrep:profile:webontology:AssociationType:OWL"
2763     lid="urn:oasis:names:tc:ebxml-
2764     regrep:profile:webontology:AssociationType:OWL:SubClassOf"
2765     code="SubClassOf" id="urn:oasis:names:tc:ebxml-
2766     regrep:profile:webontology:AssociationType:OWL:SubClassOf">
2767         <rim:Name>
2768             <rim:LocalizedString charset="UTF-8" value="SubClassOf"/>
2769         </rim:Name>
2770     </rim:ClassificationNode>
2771
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:IntersectionOf"
2776     code="IntersectionOf" id="urn:oasis:names:tc:ebxml-
2777     regrep:profile:webontology:AssociationType:OWL:IntersectionOf">
2778         <rim:Name>
2779             <rim:LocalizedString charset="UTF-8"
2780     value="IntersectionOf"/>
2781         </rim:Name>
2782     </rim:ClassificationNode>
2783
2784     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2785     regrep:profile:webontology:AssociationType:OWL"
2786     lid="urn:oasis:names:tc:ebxml-
2787     regrep:profile:webontology:AssociationType:OWL:SameAs" code="SameAs"
2788     id="urn:oasis:names:tc:ebxml-
2789     regrep:profile:webontology:AssociationType:OWL:SameAs">
2790         <rim:Name>
2791             <rim:LocalizedString charset="UTF-8" value="SameAs"/>
2792         </rim:Name>
2793     </rim:ClassificationNode>
2794
2795     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2796     regrep:profile:webontology:AssociationType:OWL "
2797     lid="urn:oasis:names:tc:ebxml-
2798     regrep:profile:webontology:AssociationType:OWL:Restriction"
2799     code="restriction" id="urn:oasis:names:tc:ebxml-
2800     regrep:profile:webontology:AssociationType:OWL:Restriction">
2801         <rim:Name>
2802             <rim:LocalizedString charset="UTF-8" value="Restriction"/>
2803         </rim:Name>
2804     </rim:ClassificationNode>
2805
2806     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2807     regrep:profile:webontology:AssociationType:OWL "
2808     lid="urn:oasis:names:tc:ebxml-
2809     regrep:profile:webontology:AssociationType:OWL:DifferentFrom"
2810     code="DifferentFrom" id="urn:oasis:names:tc:ebxml-
2811     regrep:profile:webontology:AssociationType:OWL:DifferentFrom">
2812         <rim:Name>
2813             <rim:LocalizedString charset="UTF-8" value="DifferentFrom"/>
2814         </rim:Name>
2815     </rim:ClassificationNode>
2816
2817     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2818     regrep:profile:webontology:AssociationType:OWL "
2819     lid="urn:oasis:names:tc:ebxml-
2820     regrep:profile:webontology:AssociationType:OWL:DatatypeProperty"
2821     code="DatatypeProperty" id="urn:oasis:names:tc:ebxml-
2822     regrep:profile:webontology:AssociationType:OWL:DatatypeProperty">
2823         <rim:Name>

```

```

2824         <rim:LocalizedString charset="UTF-8"
2825 value="DatatypeProperty"/>
2826         </rim:Name>
2827 </rim:ClassificationNode>
2828
2829 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2830 regrep:profile:webontology:AssociationType:OWL "
2831 lid="urn:oasis:names:tc:ebxml-
2832 regrep:profile:webontology:AssociationType:OWL:TransitiveProperty"
2833 code="TransitiveProperty" id="urn:oasis:names:tc:ebxml-
2834 regrep:profile:webontology:AssociationType:OWL:TransitiveProperty">
2835     <rim:Name>
2836         <rim:LocalizedString charset="UTF-8"
2837 value="TransitiveProperty"/>
2838     </rim:Name>
2839 </rim:ClassificationNode>
2840
2841 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2842 regrep:profile:webontology:AssociationType:OWL "
2843 lid="urn:oasis:names:tc:ebxml-
2844 regrep:profile:webontology:AssociationType:OWL:InverseOf"
2845 code="InverseOf" id="urn:oasis:names:tc:ebxml-
2846 regrep:profile:webontology:AssociationType:OWL:InverseOf">
2847     <rim:Name>
2848         <rim:LocalizedString charset="UTF-8" value="InverseOf"/>
2849     </rim:Name>
2850 </rim:ClassificationNode>
2851
2852 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2853 regrep:profile:webontology:AssociationType:OWL "
2854 lid="urn:oasis:names:tc:ebxml-
2855 regrep:profile:webontology:AssociationType:OWL:SymmetricProperty"
2856 code="SymmetricProperty" id="urn:oasis:names:tc:ebxml-
2857 regrep:profile:webontology:AssociationType:OWL:SymmetricProperty">
2858     <rim:Name>
2859         <rim:LocalizedString charset="UTF-8"
2860 value="SymmetricProperty"/>
2861     </rim:Name>
2862 </rim:ClassificationNode>
2863
2864 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2865 regrep:profile:webontology:AssociationType:OWL "
2866 lid="urn:oasis:names:tc:ebxml-
2867 regrep:profile:webontology:AssociationType:OWL:FunctionalProperty"
2868 code="FunctionalProperty" id="urn:oasis:names:tc:ebxml-
2869 regrep:profile:webontology:AssociationType:OWL:FunctionalProperty">
2870     <rim:Name>
2871         <rim:LocalizedString charset="UTF-8"
2872 value="FunctionalProperty"/>
2873     </rim:Name>
2874 </rim:ClassificationNode>
2875
2876 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2877 regrep:profile:webontology:AssociationType:OWL "
2878 lid="urn:oasis:names:tc:ebxml-
2879 regrep:profile:webontology:AssociationType:OWL:InverseFunctionalProperty"
2880 code="InverseFunctionalProperty" id="urn:oasis:names:tc:ebxml-
2881 regrep:profile:webontology:AssociationType:OWL:InverseFunctionalProperty"
2882 >
2883     <rim:Name>
2884         <rim:LocalizedString charset="UTF-8"
2885 value="InverseFunctionalProperty"/>
2886     </rim:Name>
2887 </rim:ClassificationNode>

```



```

2888 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2889 regrep:profile:webontology:AssociationType:OWL "
2890 lid="urn:oasis:names:tc:ebxml-
2891 regrep:profile:webontology:AssociationType:OWL:SeeAlso" code="SeeAlso"
2892 id="urn:oasis:names:tc:ebxml-
2893 regrep:profile:webontology:AssociationType:OWL:SeeAlso">
2894   <rim:Name>
2895     <rim:LocalizedString charset="UTF-8" value="SeeAlso"/>
2896   </rim:Name>
2897 </rim:ClassificationNode>

```

Extensions to the AssociationType ClassificationScheme

2899 7.3 Canonical Queries

2900 The following new canonical queries are described by this profile. Note that while these queries are
 2901 complex, the complexity is hidden from clients by exposing only the query parameters to them.

2902 7.3.1 All SuperProperties Discovery Query

2903 Recursion is not supported by SQL-92, for this reason the stored procedure for this query coded in SQL
 2904 99 Standard is available from:

2905 <http://www.srdc.metu.edu.tr/ebxml/ebXMLRegistryProfileForOWL/StoredProceduresSupportingebXMLRegistryProfileforOWL.htm>.

2907

2908 7.3.2 Immediate SuperClass Discovery Query

```

2909 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2910 regrep:profile:webontology:query:FindImmediateSuperClasses"
2911 id="urn:oasis:names:tc:ebxml-
2912 regrep:profile:webontology:query:FindImmediateSuperClasses">
2913   <rim:Name>
2914     <rim:LocalizedString
2915 value="label.FindImmediateSuperClasses"/>
2916   </rim:Name>
2917   <rim:Description>
2918     <rim:LocalizedString
2919 value="label.FindImmediateSuperClasses.desc"/>
2920   </rim:Description>
2921   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2922 regrep:QueryLanguage:SQL-92">
2923     SELECT C2.*
2924     FROM ClassificationNode C2, Association A, Name_ N,
2925 ClassificationNode C1
2926     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2927 regrep:profile:webontology:AssociationType:OWL:SubClassOf' AND
2928     C1.id = N.parent AND
2929     N.value LIKE '$className' AND
2930     A.sourceObject = C1.id AND
2931     A.targetObject = C2.id
2932   </rim:QueryExpression>
2933 </rim:AdhocQuery>
2934

```

2935 **The Adhoc Query retrieving immediate super classes of a given classification node**

2936 7.3.3 Immediate SubClass Discovery Query

```

2937 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2938 regrep:profile:webontology:query:FindImmediateSubClasses"
2939 id="urn:oasis:names:tc:ebxml-
2940 regrep:profile:webontology:query:FindImmediateSubClasses">
2941   <rim:Name>

```

```

2942         <rim:LocalizedString value="label.FindImmediateSubClasses"/>
2943     </rim:Name>
2944     <rim:Description>
2945         <rim:LocalizedString
2946 value="label.FindImmediateSubClasses.desc"/>
2947     </rim:Description>
2948     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2949 regrep:QueryLanguage:SQL-92">
2950         SELECT C2.*
2951         FROM ClassificationNode C2, Association A, Name_ N,
2952 ClassificationNode C1
2953         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2954 regrep:profile:webontology:AssociationType:OWL:SubClassOf' AND
2955         C1.id = N.parent AND
2956         N.value LIKE '$className' AND
2957         A.sourceObject = C2.id AND
2958         A.targetObject = C1.id
2959     </rim:QueryExpression>
2960 </rim:AdhocQuery>

```

2961 **The Adhoc Query retrieving immediate subclasses of a given classification node**

2962 7.3.4 All SuperClasses Discovery Query

2963 Recursion is not supported by SQL-92, for this reason the stored procedure for this query coded in SQL
2964 99 Standard is available from:

2965 <http://www.srdc.metu.edu.tr/ebxml/ebXMLRegistryProfileForOWL/StoredProceduresSupportingebXMLRegistryProfileforOWL.htm>.

2967

2968 7.3.5 All SubClasses Discovery Query

2969 Recursion is not supported by SQL-92, for this reason the stored procedure for this query coded in SQL
2970 99 Standard is available from:

2971 <http://www.srdc.metu.edu.tr/ebxml/ebXMLRegistryProfileForOWL/StoredProceduresSupportingebXMLRegistryProfileforOWL.htm>.

2972

2973 7.3.6 EquivalentClasses Discovery Query

```

2974 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2975 regrep:profile:webontology:query:FindEquivalentClasses"
2976 id="urn:oasis:names:tc:ebxml-
2977 regrep:profile:webontology:query:FindEquivalentClasses">
2978     <rim:Name>
2979         <rim:LocalizedString value="label.FindEquivalentClasses"/>
2980     </rim:Name>
2981     <rim:Description>
2982         <rim:LocalizedString
2983 value="label.FindEquivalentClasses.desc"/>
2984     </rim:Description>
2985     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2986 regrep:QueryLanguage:SQL-92">
2987         SELECT C2.*
2988         FROM ClassificationNode C2, Association A, Name_ N,
2989 ClassificationNode C
2990         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2991 regrep:profile:webontology:AssociationType:OWL:EquivalentTo' AND
2992         C.id = N.parent AND
2993         N.value LIKE '$className' AND
2994         A.sourceObject = C.id AND
2995         A.targetObject = C2.id
2996     </rim:QueryExpression>
2997 </rim:AdhocQuery>

```

2998

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

2999 7.3.7 EquivalentProperties Discovery Query

```
3000 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3001 regrep:profile:webontology:query:FindEquivalentProperties"
3002 id="urn:oasis:names:tc:ebxml-
3003 regrep:profile:webontology:query:FindEquivalentProperties">
3004   <rim:Name>
3005     <rim:LocalizedString
3006 value="label.FindEquivalentProperties"/>
3007   </rim:Name>
3008   <rim:Description>
3009     <rim:LocalizedString
3010 value="label.FindEquivalentProperties.desc"/>
3011   </rim:Description>
3012   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3013 regrep:QueryLanguage:SQL-92">
3014     SELECT A3.*
3015     FROM Association A3, Association A1, Name_ N, Association
3016 A2
3017     WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
3018 regrep:profile:webontology:AssociationType:OWL:EquivalentTo' AND
3019     A2.id = N.parent AND
3020     N.value LIKE '$propertyName' AND
3021     A1.sourceObject = A2.id AND
3022     A1.targetObject = A3.id
3023   </rim:QueryExpression>
3024 </rim:AdhocQuery>
```

3025 **Adhoc Query retrieving all the equivalent Association Type of a given Association Type**

3026 7.3.8 SameExtrinsicObjects Discovery Query

```
3027 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3028 regrep:profile:webontology:query:FindTheSameExtrinsicObjects"
3029 id="urn:oasis:names:tc:ebxml-
3030 regrep:profile:webontology:query:FindTheSameExtrinsicObjects">
3031   <rim:Name>
3032     <rim:LocalizedString
3033 value="label.FindTheSameExtrinsicObjects"/>
3034   </rim:Name>
3035   <rim:Description>
3036     <rim:LocalizedString
3037 value="label.FindTheSameExtrinsicObjects.desc"/>
3038   </rim:Description>
3039   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3040 regrep:QueryLanguage:SQL-92">
3041     SELECT E2.*
3042     FROM ExtrinsicObject E2, Association A, Name_ N,
3043 ExtrinsicObject E
3044     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3045 regrep:profile:webontology:AssociationType:OWL:SameAs' AND
3046     E.id = N.parent AND
3047     N.value LIKE '$extrinsicObjectName' AND
3048     A.sourceObject = E.id AND
3049     A.targetObject = E2.id
3050   </rim:QueryExpression>
3051 </rim:AdhocQuery>
```

3052 **Adhoc Query retrieving all the "ExtrinsicObjects" defined to be the same with a given**
3053 **ExtrinsicObject**

3054 7.3.9 DifferentExtrinsicObjects Discovery Query

```
3055 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3056 regrep:profile:webontology:query:FindDifferentExtrinsicObjects"
3057 id="urn:oasis:names:tc:ebxml-
3058 regrep:profile:webontology:query:FindDifferentExtrinsicObjects">
3059   <rim:Name>
3060     <rim:LocalizedString
3061 value="label.FindDifferentExtrinsicObjects"/>
3062   </rim:Name>
3063   <rim:Description>
3064     <rim:LocalizedString
3065 value="label.FindDifferentExtrinsicObjects.desc"/>
3066   </rim:Description>
3067   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3068 regrep:QueryLanguage:SQL-92">
3069     SELECT E2.*
3070     FROM ExtrinsicObject E2, Association A, Name_ N,
3071 ExtrinsicObject E
3072     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3073 regrep:profile:webontology:AssociationType:OWL:DifferentFrom' AND
3074     E.id = N.parent AND
3075     N.value LIKE '$extrinsicObjectName' AND
3076     A.sourceObject = E.id AND
3077     A.targetObject = E2.id
3078   </rim:QueryExpression>
3079 </rim:AdhocQuery>
```

3080 **Adhoc Query retrieving all the "ExtrinsicObjects" defined to be different from a given**
3081 **ExtrinsicObject**

3082 7.3.10 AllDifferentRegistryObject Discovery Query

```
3083 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3084 regrep:profile:webontology:query:FindAllDifferent"
3085 id="urn:oasis:names:tc:ebxml-
3086 regrep:profile:webontology:query:FindAllDifferent">
3087   <rim:Name>
3088     <rim:LocalizedString value="label.FindAllDifferent"/>
3089   </rim:Name>
3090   <rim:Description>
3091     <rim:LocalizedString value="label.FindAllDifferent.desc"/>
3092   </rim:Description>
3093   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3094 regrep:QueryLanguage:SQL-92">
3095     SELECT RO2.*
3096     FROM RegistryObject RO2, Association A1, Association A2,
3097 Name_ N, RegistryObject RO,
3098 RegistryPackage RP<!--, Slot S-->
3099     WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
3100 regrep:profile:webontology:AssociationType:OWL:HasMember' AND
3101     RO.id = N.parent AND
3102     N.value LIKE '$registryObjectName' AND
3103     A1.sourceObject = RP.id AND
3104     <!-- S.parent = RP.id AND
3105     S.name_ LIKE 'packageType' AND S.value LIKE
3106 'allDifferent' AND -->
3107     A1.targetObject = RO.id AND
3108     A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3109 regrep:profile:webontology:AssociationType:OWL:HasMember' AND
3110     A2.sourceObject = RP.id AND
3111     A2.targetObject != RO.id AND
3112     A2.targetObject = RO2.id
3113   </rim:QueryExpression>
3114 </rim:AdhocQuery>
```

3115 **Adhoc Query retrieving all the "RegistryObjects" defined to be different from a given**

3116 RegistryObject through a "allDifferent" construct

3117 7.3.11 ObjectProperties Discovery Query

```
3118 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3119 regrep:profile:webontology:query:FindObjectProperties"
3120 id="urn:oasis:names:tc:ebxml-
3121 regrep:profile:webontology:query:FindObjectProperties">
3122   <rim:Name>
3123     <rim:LocalizedString value="label.FindObjectProperties"/>
3124   </rim:Name>
3125   <rim:Description>
3126     <rim:LocalizedString
3127 value="label.FindObjectProperties.desc"/>
3128   </rim:Description>
3129   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3130 regrep:QueryLanguage:SQL-92">
3131     SELECT A.*
3132     FROM Association A, Name_N, ClassificationNode C
3133     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3134 regrep:profile:webontology:AssociationType:OWL:ObjectProperty' AND
3135     C.id = N.parent AND
3136     N.value LIKE '$className' AND
3137     A.sourceObject = C.id
3138   </rim:QueryExpression>
3139 </rim:AdhocQuery>
```

3140 **Adhoc Query retrieving all the object properties of a given classification node**

3141 7.3.12 ImmediateInheritedObjectProperties Discovery Query

```
3142 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3143 regrep:profile:webontology:query:FindImmediateInheritedObjectProperties"
3144 id="urn:oasis:names:tc:ebxml-
3145 regrep:profile:webontology:query:FindImmediateInheritedObjectProperties">
3146   <rim:Name>
3147     <rim:LocalizedString
3148 value="label.FindImmediateInheritedObjectProperties"/>
3149   </rim:Name>
3150   <rim:Description>
3151     <rim:LocalizedString
3152 value="label.FindImmediateInheritedObjectProperties.desc"/>
3153   </rim:Description>
3154   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3155 regrep:QueryLanguage:SQL-92">
3156     SELECT A2.*
3157     FROM Association A, Name_N, ClassificationNode C1,
3158 ClassificationNode C2, Association A2
3159     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3160 regrep:profile:webontology:AssociationType:OWL:SubClassOf' AND
3161     C1.id = N.parent AND
3162     N.value LIKE '$className' AND
3163     A.sourceObject = C1.id AND
3164     A.targetObject = C2.id AND
3165     A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3166 regrep:profile:webontology:AssociationType:OWL:ObjectProperty' AND
3167     A2.sourceObject=C2.id
3168   </rim:QueryExpression>
3169 </rim:AdhocQuery>
```

3170 **Adhoc Query retrieving all of the properties of a given classification node including the ones**
3171 **inherited from its immediate super classes**

3172 7.3.13 AllInheritedObjectProperties Discovery Query

3173 Recursion is not supported by SQL-92, for this reason the stored procedure for this query coded in SQL
3174 99 Standard is available from:

3175 <http://www.srdc.metu.edu.tr/ebxml/ebXMLRegistryProfileForOWL/StoredProceduresSupportingebXMLRegistryProfileforOWL.htm>.
3176

3177 7.3.14 DatatypeProperties Discovery Query

```
3178 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-  
3179 regrep:profile:webontology:query:FindDatatypeProperties"  
3180 id="urn:oasis:names:tc:ebxml-  
3181 regrep:profile:webontology:query:FindDatatypeProperties">  
3182   <rim:Name>  
3183     <rim:LocalizedString value="label.FindDatatypeProperties"/>  
3184   </rim:Name>  
3185   <rim:Description>  
3186     <rim:LocalizedString  
3187 value="label.FindDatatypeProperties.desc"/>  
3188   </rim:Description>  
3189   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
3190 regrep:QueryLanguage:SQL-92">  
3191     SELECT A.*  
3192     FROM Association A, Name_ N, ClassificationNode C  
3193     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-  
3194 regrep:profile:webontology:AssociationType:OWL:DatatypeProperty' AND  
3195     C.id = N.parent AND  
3196     N.value LIKE '$className' AND  
3197     A.sourceObject = C.id  
3198   </rim:QueryExpression>  
3199 </rim:AdhocQuery>
```

3200 **Adhoc Query retrieving all the datatype properties of a given classification node**

3201 7.3.15 AllInheritedDatatypeProperties Discovery Query

3202 Recursion is not supported by SQL-92, for this reason the stored procedure for this query coded in SQL
3203 99 Standard is available from:

3204 <http://www.srdc.metu.edu.tr/ebxml/ebXMLRegistryProfileForOWL/StoredProceduresSupportingebXMLRegistryProfileforOWL.htm>.
3205

3206 7.3.16 TransitiveRelationships Discovery Query

```
3207 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-  
3208 regrep:profile:webontology:query:FindTransitiveRelationships"  
3209 id="urn:oasis:names:tc:ebxml-  
3210 regrep:profile:webontology:query:FindTransitiveRelationships">  
3211   <rim:Name>  
3212     <rim:LocalizedString  
3213 value="label.FindTransitiveRelationships"/>  
3214   </rim:Name>  
3215   <rim:Description>  
3216     <rim:LocalizedString  
3217 value="label.FindTransitiveRelationships.desc"/>  
3218   </rim:Description>  
3219   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
3220 regrep:QueryLanguage:SQL-92">  
3221     SELECT C.*  
3222     FROM ClassificationNode C, Association A1, Association A2,  
3223     Name_ N1, Name_ N2, Name_ N3  
3224     WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-  
3225 regrep:profile:webontology:AssociationType:OWL:TransitiveProperty' AND  
3226     A1.id = N1.parent AND  
3227     N1.value LIKE '$propertyName' AND
```

```

3228         A1.sourceObject = N3.parent AND
3229         N3.value LIKE '$className' AND
3230         A2.sourceObject = A1.targetObject AND
3231         A2.id = N2.parent AND
3232         N2.value LIKE '$propertyName' AND
3233         A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3234 regrep:profile:webontology:AssociationType:OWL:TransitiveProperty' AND
3235         A2.targetObject = C.id
3236         <!-- UNION
3237         SELECT C.*
3238         FROM ClassificationNode C, Association A1, Name_ N1, Name_
3239 N3
3240         WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
3241 regrep:profile:webontology:AssociationType:OWL:TransitiveProperty' AND
3242         A1.id = N1.parent AND
3243         N1.value LIKE '$propertyName' AND
3244         A1.sourceObject = N3.parent AND
3245         N3.value LIKE '$className' AND
3246         A1.targetObject = C.id -->
3247     </rim:QueryExpression>
3248 </rim:AdhocQuery>

```

3249 **Adhoc Query retrieving the objects in transitive relationship with a given object**

3250 7.3.17 TargetObjects Discovery Query

```

3251 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3252 regrep:profile:webontology:query:FindTargetObjects"
3253 id="urn:oasis:names:tc:ebxml-
3254 regrep:profile:webontology:query:FindTargetObjects">
3255     <rim:Name>
3256         <rim:LocalizedString value="label.FindTargetObjects"/>
3257     </rim:Name>
3258     <rim:Description>
3259         <rim:LocalizedString value="label.FindTargetObjects.desc"/>
3260     </rim:Description>
3261     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3262 regrep:QueryLanguage:SQL-92">
3263         SELECT C2.*
3264         FROM ClassificationNode C2, Association A, Name_ N, Name_
3265 N2, ClassificationNode C1
3266         WHERE A.id=N2.parent AND
3267         N2.value LIKE '$propertyName' AND
3268         C1.id = N.parent AND
3269         N.value LIKE '$className' AND
3270         A.sourceObject = C1.id AND
3271         A.targetObject = C2.id
3272     </rim:QueryExpression>
3273 </rim:AdhocQuery>

```

3274 **Adhoc Query retrieving the Target Objects from the Registry, given a Source Object and an**
3275 **Association**

3276 7.3.18 TargetObjectsInverseOf Discovery Query

```

3277 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3278 regrep:profile:webontology:query:FindTOinverseOf"
3279 id="urn:oasis:names:tc:ebxml-
3280 regrep:profile:webontology:query:FindTOinverseOf">
3281     <rim:Name>
3282         <rim:LocalizedString value="label.FindTOinverseOf"/>
3283     </rim:Name>
3284     <rim:Description>
3285         <rim:LocalizedString value="label.FindTOinverseOf.desc"/>
3286     </rim:Description>
3287     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3288 regrep:QueryLanguage:SQL-92">

```



```

3289         SELECT C2.*
3290         FROM ClassificationNode C2, Association A1, Association A2,
3291 Association A3, Name_ N, Name_ N2, ClassificationNode C1
3292         WHERE A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3293 regrep:profile:webontology:AssociationType:OWL:InverseOf' AND
3294         A1.id = N.parent AND
3295         N.value LIKE '$propertyName' AND
3296         A2.sourceObject = A1.id AND
3297         A2.targetObject = A3.id AND
3298         C1.id = N2.parent AND
3299         N2.value LIKE '$className' AND
3300         A3.targetObject = C1.id AND
3301         A3.sourceObject = C2.id
3302     </rim:QueryExpression>
3303 </rim:AdhocQuery>

```

3304 **Adhoc query retrieving the Source Objects of an Association which is in "inverseOf"**
3305 **relationship to this Association**

3306 7.3.19 InverseRanges Discovery Query

```

3307 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3308 regrep:profile:webontology:query:FindInverseRanges"
3309 id="urn:oasis:names:tc:ebxml-
3310 regrep:profile:webontology:query:FindInverseRanges">
3311     <rim:Name>
3312         <rim:LocalizedString value="label.FindInverseRanges"/>
3313     </rim:Name>
3314     <rim:Description>
3315         <rim:LocalizedString value="label.FindInverseRanges.desc"/>
3316     </rim:Description>
3317     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3318 regrep:QueryLanguage:SQL-92">
3319         <!-- SELECT C2.*
3320         FROM Association A, Name_ N, Name_ N2, ClassificationNode
3321 C1, ClassificationNode C2
3322         WHERE A.id=N2.parent AND
3323         N2.value LIKE '$propertyName' AND
3324         C1.id = N.parent AND
3325         N.value LIKE '$className' AND
3326         A.sourceObject = C1.id AND
3327         A.targetObject = C2.id
3328         UNION -->
3329         SELECT C2.*
3330         FROM ClassificationNode C2, Association A1, Association A2,
3331 Association A3, Name_ N, NAME_ N2, ClassificationNode C1
3332         WHERE A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3333 regrep:profile:webontology:AssociationType:OWL:InverseOf' AND
3334         A1.id = N.parent AND
3335         N.value LIKE '$propertyName' AND
3336         A2.sourceObject = A1.id AND
3337         A2.targetObject = A3.id AND
3338         C1.id = N2.parent AND
3339         N2.value LIKE '$className' AND
3340         A1.sourceObject = C1.id AND
3341         A3.sourceObject = C2.id
3342     </rim:QueryExpression>
3343 </rim:AdhocQuery>

```

3344 **Adhoc Query Retrieving both the Target Objects of a given Association and the Source**
3345 **Objects of an Association which is in "inverseOf" relationship to this Association**

3346 7.3.20 SymmetricProperties Discovery Query

```
3347 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3348 regrep:profile:webontology:query:FindSymmetricProperties"
3349 id="urn:oasis:names:tc:ebxml-
3350 regrep:profile:webontology:query:FindSymmetricProperties">
3351   <rim:Name>
3352     <rim:LocalizedString value="label.FindSymmetricProperties"/>
3353   </rim:Name>
3354   <rim:Description>
3355     <rim:LocalizedString
3356 value="label.FindSymmetricProperties.desc"/>
3357   </rim:Description>
3358   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3359 regrep:QueryLanguage:SQL-92">
3360     SELECT A.*
3361     FROM Association A, Name_N, ClassificationNode C
3362     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3363 regrep:profile:webontology:AssociationType:OWL:SymmetricProperty' AND
3364     C.id = N.parent AND
3365     N.value LIKE '$className' AND
3366     A.sourceObject = C.id
3367   </rim:QueryExpression>
3368 </rim:AdhocQuery>
```

3369 **Adhoc Query retrieving all the Symmetric properties of a given classification node**

3370 7.3.21 FunctionalProperties Discovery Query

```
3371 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3372 regrep:profile:webontology:query:FindFunctionalProperties"
3373 id="urn:oasis:names:tc:ebxml-
3374 regrep:profile:webontology:query:FindFunctionalProperties">
3375   <rim:Name>
3376     <rim:LocalizedString
3377 value="label.FindFunctionalProperties"/>
3378   </rim:Name>
3379   <rim:Description>
3380     <rim:LocalizedString
3381 value="label.FindFunctionalProperties.desc"/>
3382   </rim:Description>
3383   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3384 regrep:QueryLanguage:SQL-92">
3385     SELECT A.*
3386     FROM Association A, Name_N, ClassificationNode C
3387     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3388 regrep:profile:webontology:AssociationType:OWL:FunctionalProperty' AND
3389     C.id = N.parent AND
3390     N.value LIKE '$className' AND
3391     A.sourceObject = C.id
3392   </rim:QueryExpression>
3393 </rim:AdhocQuery>
```

3394 **Adhoc Query retrieving all the Functional properties of a given classification node**

3395 7.3.22 InverseFunctionalProperties Discovery Query

```
3396 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3397 regrep:profile:webontology:query:FindInverseFunctionalProperties"
3398 id="urn:oasis:names:tc:ebxml-
3399 regrep:profile:webontology:query:FindInverseFunctionalProperties">
3400   <rim:Name>
3401     <rim:LocalizedString
3402 value="label.FindInverseFunctionalProperties"/>
3403   </rim:Name>
3404   <rim:Description>
```

```

3405         <rim:LocalizedString
3406 value="label.FindInverseFunctionalProperties.desc"/>
3407     </rim:Description>
3408     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3409 regrep:QueryLanguage:SQL-92">
3410         SELECT A.*
3411         FROM Association A, Name_N, ClassificationNode C
3412         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3413 regrep:profile:webontology:AssociationType:OWL:InverseFunctionalProperty'
3414         ' AND
3415             C.id = N.parent AND
3416             N.value LIKE '$className' AND
3417             A.sourceObject = C.id
3418     </rim:QueryExpression>
3419 </rim:AdhocQuery>

```

3420 **Adhoc Query retrieving all the Inverse Functional properties of a given classification node**

3421 7.3.23 Instances Discovery Query Discovery Query

```

3422 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3423 regrep:profile:webontology:query:FindInstances"
3424 id="urn:oasis:names:tc:ebxml-
3425 regrep:profile:webontology:query:FindInstances">
3426     <rim:Name>
3427         <rim:LocalizedString value="label.FindInstances"/>
3428     </rim:Name>
3429     <rim:Description>
3430         <rim:LocalizedString value="label.FindInstances.desc"/>
3431     </rim:Description>
3432     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3433 regrep:QueryLanguage:SQL-92">
3434         <!-- SELECT S.* FROM Service S, (
3435         SELECT S1.value AS id
3436         FROM Slot S1, Name_N, ClassificationNode C
3437         WHERE S1.parent = C.id AND
3438
3439             C.id = N.parent AND
3440             N.value LIKE '$className' AND
3441             S1.name_ LIKE 'urn:oasis:names:tc:ebxml-
3442 regrep:profile:webontology:slot:intersectionOf '
3443         '
3444         )
3445         AS T1, (
3446         SELECT S1.value AS id
3447         FROM Slot S1, Name_N, ClassificationNode C
3448         WHERE S1.parent = C.id AND
3449             C.id = N.parent AND
3450             N.value LIKE '$className' AND
3451             S1.name_ LIKE 'urn:oasis:names:tc:ebxml-
3452 regrep:profile:webontology:slot:intersectionOf '
3453         '
3454         ) AS T2
3455         WHERE S.id IN (
3456         SELECT classifiedObject
3457         FROM Classification
3458         WHERE classificationNode=T1.id
3459         INTERSECT
3460         SELECT classifiedObject
3461         FROM Classification
3462         WHERE classificationNode=T2.id
3463         ) AND T1.id!=T2.id
3464         UNION -->
3465         SELECT S.*
3466         FROM Service S, Classification C, ClassificationNode CN,
3467         Name_N
3468         WHERE S.id = C.classifiedObject AND
3469             C.classificationNode = CN.id AND

```

3469
3470
3471
3472
3473

```
N.value LIKE '$className' AND  
N.parent = CN.id  
</rim:QueryExpression>  
</rim:AdhocQuery>
```

Adhoc Query Retrieving the instances of intersected classes

3474 8 OWL Profile References

3475 8.1 Normative References

- 3476 [Bechhofer, Harmelen, Hendler, Horrocks, McGuinness, Patel-Schneider, Stein]
3477 Bechhofer, S., Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., Stein, L.
3478 A., OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004
3479 <http://www.w3.org/TR/2004/REC-owl-ref-20040210/>
3480
- 3481 [Brickley, Guha] Brickley, D., Guha, R.V., RDF Vocabulary Description Language 1.0: RDF Schema
3482 W3C Recommendation 10 February 2004
3483 <http://www.w3.org/TR/rdf-schema/>
3484
- 3485 [DAML+OIL] <http://www.daml.org/>
3486 [ebRIM] ebXML Registry Information Model version 3.0
3487 <http://docs.oasis-open.org/regrep/regrep-rim/v3.0/regrep-rim-3.0-os.pdf>
3488
- 3489 [ebRS] ebXML Registry Services Specification version 3.0
3490 <http://docs.oasis-open.org/regrep/regrep-rs/v3.0/regrep-rs-3.0-os.pdf>
- 3491 [ebRR-DPT] Deployment Profile Template For ebXML Registry 3.0 OASIS Specifications V_0.1.2
- 3492 [ebMS-DPT] Deployment Profile Template For OASIS Specification ebXML Message Service 2.0
- 3493 [McGuinness, Harmelen] McGuinness, D. L., Harmelen, F., OWL Web Ontology Language Overview,
3494 W3C Recommendation 10 February 2004, <http://www.w3.org/TR/owl-features/>
- 3495 [OWL] Web Ontology Language (OWL), <http://www.w3.org/2004/OWL/>
- 3496 [RDF] Resource Description Framework, <http://www.w3.org/TR/rdf-concepts/>
- 3497 [RDFS] RDF Vocabulary Description Language 1.0: RDF Schema <http://www.w3.org/TR/rdf-schema/>
- 3498 [Smith, Welty, McGuinness] Smith, M. K., Welty, C., McGuinness, D. L.,
3499 OWL Web Ontology Language Guide, W3C Recommendation 10 February 2004,
3500 <http://www.w3.org/TR/owl-guide/>
- 3501 [SQL 92] SQL ISO/IEC 9075:1992 Information technology - Database languages - SQL.
- 3502 [SQL 99] ISO/IEC 9075:1999(E) Information technology - Database languages – SQL.
- 3503 [UML] Unified Modeling Language version 1.5
3504 <http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.pdf>
- 3505 [WSDL] WSDL Specification
3506 <http://www.w3.org/TR/wsdl>

3507 **8.2 Informative References**

3508 [Dogac, et. al. 2005] Dogac A., Kabak Y., Laleci G. C. Mattocks, F. Najmi, J. Pollock
3509 Enhancing ebXML Registries to Make them OWL Aware
3510 Distributed and Parallel Databases Journal, Springer-Verlag, Vol. 18, No. 1, July 2005, pp. 9-36.

3511
3512 [Dogac et. al. 2006] Dogac A., Laleci G., Kabak Y., Unal S., Beale T., Heard S., Elkin P., Najmi F.,
3513 Mattocks C., Webber D., Kernberg M.
3514 Exploiting ebXML Registry Semantic Constructs for Handling Archetype Metadata in Healthcare
3515 Informatics
3516 International Journal of Metadata, Semantics and Ontologies, Volume 1, No. 1, 2006.

3517
3518 [IMPL] ebXML Registry 3.0 Implementations
3519 freebXML Registry: A royalty free, open source ebXML Registry Implementation
3520 <http://ebxmlrr.sourceforge.net>

3521
3522 [LeeHendler]
3523 Berners-Lee, T., Hendler, J., Lassila, O., "The Semantic Web", Scientific American, May 2001.

3524
3525 [StaabStuder] Staab, S., Studer, R., Handbook on Ontologies, Springer, 2004.

3526

3527 **Appendix A**

3528 **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	-

3529