



ebXML Registry Profile for Web Ontology

Language (OWL)

Version 1.5

Draft OASIS Profile, July 04, 2006

Document identifier:

regrep-owl-profile-1.5-July4

Location:

<http://www.oasis-open.org/committees/regrep-semantic/documents/profile/regrep-owl-profile-1.5.pdf>

Authors:

Name	Affiliation
Asuman Dogac	Middle East Technical University, Software R&D Center, Turkey
Yildiray Kabak	Middle East Technical University, Software R&D Center, Turkey
Gokce B. Laleci	Middle East Technical University, Software R&D Center, Turkey

Contributors:

Name	Affiliation
Farrukh Najmi	Sun Micro Systems, USA
Carl Mattocks	ITIL Application Knowledge Management, USA
Jeff Pollock	Network Inference, USA
Evan Wallace	NIST, USA
Dave RR Webber	XML Consultant, USA
Nikola Stojanovic	RosettaNet
Ivan Bedini	France Telecom, France

Abstract:

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

18

19 **Status:**

20 This document is an OASIS ebXML Registry Semantic Content Management Committee Working
21 Draft Profile and the work by the Editors is realized within the scope of the IST 2104 SATINE
22 Project sponsored by the European Commission, DG Information Society and Media, eBusiness
23 Unit.

24 Committee members should send comments on this specification to the regrep-semantic@lists.oasis-open.org list. Others should subscribe to and send comments to the regrep-comment@lists.oasis-open.org list. To subscribe, send an email message to regrep-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

28 For information on whether any patents have been disclosed that may be essential to
29 implementing this specification, and any offers of patent licensing terms, please refer to the
30 Intellectual Property Rights section of the OASIS ebXML Registry TC web page (<http://www.oasis-open.org/committees/regrep/>).
31

1 Table of Contents

32		
33	1 Table of Contents.....	3
34	1 Introduction.....	10
35	1.1 Terminology.....	11
36	1.2 Conventions.....	11
37	1.3 Recommended Enhancements.....	11
38	2 OWL Overview.....	12
39	2.1 Semantic Web Languages upon which OWL is Layered.....	12
40	2.2 OWL Lite Constructs.....	13
41	2.2.1 RDF Schema Features.....	13
42	2.2.2 (In)Equality.....	13
43	2.2.3 Property Characteristics	13
44	2.2.4 Property Restrictions.....	13
45	2.2.5 Restricted Cardinality.....	13
46	2.2.6 Class Intersection.....	13
47	2.2.7 Versioning.....	14
48	2.2.8 Annotation Properties	14
49	2.2.9 Datatypes	14
50	2.3 OWL DL Constructs.....	14
51	2.3.1 Class Axioms.....	14
52	2.3.2 Boolean Combinations of Class Expressions	14
53	2.3.3 Arbitrary Cardinality	14
54	2.3.4 Filler Information.....	14
55	3 ebXML Registry Overview.....	15
56	3.1 Overview of [ebRIM].....	15
57	3.1.1 RegistryObject.....	16
58	3.1.2 Object Identification.....	16
59	3.1.3 Object Naming and Description.....	17
60	3.1.4 Object Attributes.....	17
61	3.1.4.1 Slot Attributes.....	17
62	3.1.5 Object Classification.....	18
63	3.1.6 Object Association.....	18
64	3.1.7 Object References To Web Content.....	19
65	3.1.8 Object Packaging.....	19
66	3.1.9 ExtrinsicObject	20
67	3.1.10 Service Description.....	20
68	3.2 Overview of [ebRS].....	20
69	4 Representing OWL Lite Constructs in ebRIM	21
70	4.1 Representing RDF Schema Features in ebRIM.....	21
71	4.1.1 owl:Class → rim:ClassificationNode.....	21
72	4.1.2 rdf:Property → rim:Association Type HasProperty.....	21
73	4.1.3 rdfs:subPropertyOf → rim:Association Type SubPropertyOf.....	22
74	4.1.4 rdfs:subClassOf → rim:Association Type SubClassOf.....	22
75	4.1.5 owl:Individual → rim:ExtrinsicObject.....	23
76	4.2 Representing OWL (In)Equality Constructs in ebXML RIM.....	24

77	4.2.1 owl:equivalentClass, owl:equivalentProperty → rim:Association Type EquivalentTo	24
78	4.2.2 owl:sameAs → rim:Association Type SameAs.....	24
79	4.2.3 owl:differentFrom → rim:Association Type DifferentFrom.....	24
80	4.2.4 owl:AllDifferent.....	25
81	4.3 Representing OWL Property Characteristics in ebRIM.....	26
82	4.3.1 owl:ObjectProperty → rim:Association Type objectProperty.....	26
83	4.3.2 owl:DatatypeProperty → rim:Association Type DatatypeProperty.....	26
84	4.3.3 owl:TransitiveProperty → rim:Association Type TransitiveProperty.....	26
85	4.3.4 owl:inverseOf → rim:Association Type InverseOf.....	27
86	4.3.5 owl:SymmetricProperty→ rim:Association Type SymmetricProperty.....	28
87	4.3.6 owl:FunctionalProperty→ rim:Association Type FunctionalProperty.....	28
88	4.3.7 owl:InverseFunctionalProperty→ rim:Association Type InverseFunctionalProperty.....	29
89	4.4 OWL Property Restrictions in ebXML RIM.....	29
90	4.5 Representing OWL Restricted Cardinality in ebXML RIM.....	30
91	4.5.1 owl:minCardinality (only 0 or 1).....	30
92	4.5.2 owl:maxCardinality (only 0 or 1).....	31
93	4.5.3 owl:cardinality (only 0 or 1).....	32
94	4.6 Representing OWL Class Intersection in ebXML RIM.....	32
95	4.7 Representing OWL Versioning in ebXML RIM.....	33
96	4.7.1 owl:versionInfo, owl:priorVersion.....	33
97	4.8 Representing OWL Annotation Properties in ebXML RIM.....	34
98	4.8.1 rdfs:label.....	34
99	4.8.2 rdfs:comment.....	34
100	4.8.3 rdfs:seeAlso.....	34
101	4.9 OWL Datatypes in ebXML RIM.....	35
102	5 Cataloging Service Profile.....	36
103	5.1 Invocation Control File.....	36
104	5.2 Input Metadata.....	36
105	5.3 Input Content.....	36
106	5.4 Output Metadata.....	37
107	5.4.1 owl:Class → rim:ClassificationNode.....	37
108	5.4.2 rdf:Property → rim:Association Type HasProperty.....	37
109	5.4.3 rdfs:subPropertyOf → rim:Association Type SubPropertyOf.....	37
110	5.4.4 rdfs:subClassOf → rim:Association Type subClassOf.....	37
111	5.4.5 owl:Individual → rim:ExtrinsicObject.....	37
112	5.4.6 owl:equivalentClass, owl:equivalentProperty → rim:Association Type EquivalentTo	37
113	5.4.7 owl:sameAs → rim:Association Type SameAs	37
114	5.4.8 owl:differentFrom → rim:Association Type DifferentFrom.....	37
115	5.4.9 owl:AllDifferent → rim:RegistryPackage.....	37
116	5.4.10 owl:ObjectProperty → rim:Association Type ObjectProperty.....	38
117	5.4.11 owl:DatatypeProperty → rim:Association Type DatatypeProperty.....	38
118	5.4.12 owl:TransitiveProperty → rim:Association Type TransitiveProperty.....	38
119	5.4.13 owl:inverseOf → rim:Association Type InverseOf.....	38
120	5.4.14 owl:SymmetricProperty→ rim:Association Type SymetricProperty.....	38
121	5.4.15 owl:FunctionalProperty→ rim:Association Type FunctionalProperty.....	38
122	5.4.16 owl:InverseFunctionalProperty→ rim:Association Type InverseFunctionalProperty.....	38

123	5.4.17 owl:minCardinality (only 0 or 1).....	38
124	5.4.18 owl:maxCardinality (only 0 or 1).....	39
125	5.4.19 owl:cardinality.....	39
126	5.4.20 owl:intersectionOf.....	39
127	5.4.21 rdfs:label.....	39
128	5.4.22 rdfs:comment.....	39
129	5.4.23 rdfs:seeAlso.....	39
130	6 Discovery Profile.....	40
131	6.1 All SuperProperties Discovery Query.....	40
132	6.1.1 Parameter \$propertyName.....	40
133	6.1.2 Example of All SuperProperties Discovery Query.....	40
134	6.2 Immediate SuperClass Discovery Query.....	41
135	6.2.1 Parameter \$className.....	41
136	6.2.2 Example of Immediate SuperClass Discovery Query.....	41
137	6.3 Immediate SubClass Discovery Query.....	42
138	6.3.1 Parameter \$className.....	42
139	6.3.2 Example of Immediate SubClasses Discovery Query.....	42
140	6.4 All SuperClasses Discovery Query.....	42
141	6.4.1 Parameter \$className.....	43
142	6.4.2 Example of All SuperClasses Discovery Query.....	43
143	6.5 All SubClasses Discovery Query.....	43
144	6.5.1 Parameter \$className.....	43
145	6.5.2 Example of All SubClasses Discovery Query.....	43
146	6.6 EquivalentClasses Discovery Query.....	44
147	6.6.1 Parameter \$className.....	44
148	6.6.2 Example of EquivalentClasses Discovery Query.....	44
149	6.7 EquivalentProperties Discovery Query.....	45
150	6.7.1 Parameter \$propertyName.....	45
151	6.7.2 Example of EquivalentProperties Discovery Query.....	45
152	6.8 SameExtrinsicObjects Discovery Query.....	46
153	6.8.1 Parameter \$extrinsicObjectName.....	46
154	6.8.2 Example of SameExtrinsicObjects Discovery Query.....	46
155	6.9 DifferentExtrinsicObjects Discovery Query.....	46
156	6.9.1 Parameter \$extrinsicObjectName.....	47
157	6.9.2 Example of DifferentExtrinsicObjects Discovery Query.....	47
158	6.10 AllDifferentRegistryObject Discovery Query.....	47
159	6.10.1 Parameter \$registryObjectName.....	47
160	6.10.2 Example of AllDifferentRegistryObjects Discovery Query.....	47
161	6.11 ObjectProperties Discovery Query.....	48
162	6.11.1 Parameter \$className.....	48
163	6.11.2 Example of ObjectProperties Discovery Query.....	48
164	6.12 ImmediateInheritedObjectProperties Discovery Query.....	49
165	6.12.1 Parameter \$className.....	49
166	6.12.2 Example of ImmediateInheritedObjectProperties Discovery Query.....	49
167	6.13 AllInheritedObjectProperties Discovery Query.....	50
168	6.13.1 Parameter \$className.....	50

169	6.13.2 Example of AllInheritedObjectProperties Discovery Query.....	50
170	6.14 DatatypeProperties Discovery Query.....	51
171	6.14.1 Parameter \$className.....	51
172	6.14.2 Example of DatatypeProperties Discovery Query.....	51
173	6.15 AllInheritedDatatypeProperties Discovery Query.....	51
174	6.15.1 Parameter \$className.....	52
175	6.15.2 Example of AllInheritedDatatypeProperties Discovery Query.....	52
176	6.16 TransitiveRelationships Discovery Query.....	52
177	6.16.1 Parameter \$className.....	52
178	6.16.2 Parameter \$propertyName.....	53
179	6.16.3 Example of TransitiveRelationships Discovery Query.....	53
180	6.17 TargetObjects Discovery Query.....	53
181	6.17.1 Parameter \$className.....	53
182	6.17.2 Parameter \$propertyName.....	54
183	6.17.3 Example of TargetObjects Discovery Query.....	54
184	6.18 TargetObjectsInverseOf Discovery Query.....	54
185	6.18.1 Parameter \$className.....	54
186	6.18.2 Parameter \$propertyName.....	55
187	6.18.3 Example of TargetObjectsInverseOf Discovery Query.....	55
188	6.19 InverseRanges Discovery Query.....	55
189	6.19.1 Parameter \$className.....	56
190	6.19.2 Parameter \$propertyName.....	56
191	6.19.3 Example of InverseRanges Discovery Query.....	56
192	6.20 SymmetricProperties Discovery Query.....	57
193	6.20.1 Parameter \$className.....	57
194	6.20.2 Example of SymmetricProperties Discovery Query.....	57
195	6.21 FunctionalProperties Discovery Query.....	57
196	6.21.1 Parameter \$className.....	58
197	6.21.2 Example of FunctionalProperties Discovery Query.....	58
198	6.22 InverseFunctionalProperties Discovery Query.....	58
199	6.22.1 Parameter \$className.....	58
200	6.22.2 Example of InverseFunctionalProperties Discovery Query.....	58
201	6.23 Instances Discovery Query.....	59
202	6.23.1 Parameter \$className.....	59
203	6.23.2 Example of Instances Discovery Query.....	59
204	7 Canonical Metadata Definitions.....	61
205	7.1 ObjectType Extensions.....	61
206	7.2 AssociationType Extensions.....	61
207	7.3 Canonical Queries.....	64
208	7.3.1 All SuperProperties Discovery Query.....	64
209	7.3.2 Immediate SuperClass Discovery Query.....	64
210	7.3.3 Immediate SubClass Discovery Query.....	64
211	7.3.4 All SuperClasses Discovery Query.....	65
212	7.3.5 All SubClasses Discovery Query.....	65
213	7.3.6 EquivalentClasses Discovery Query.....	65
214	7.3.7 EquivalentProperties Discovery Query.....	66

215	7.3.8 SameExtrinsicObjects Discovery Query.....	66
216	7.3.9 DifferentExtrinsicObjects Discovery Query.....	67
217	7.3.10 AllDifferentRegistryObject Discovery Query.....	67
218	7.3.11 ObjectProperties Discovery Query.....	68
219	7.3.12 ImmediateInheritedObjectProperties Discovery Query.....	68
220	7.3.13 AllInheritedObjectProperties Discovery Query.....	68
221	7.3.14 DatatypeProperties Discovery Query.....	69
222	7.3.15 AllInheritedDatatypeProperties Discovery Query.....	69
223	7.3.16 TransitiveRelationships Discovery Query.....	69
224	7.3.17 TargetObjects Discovery Query.....	70
225	7.3.18 TargetObjectsInverseOf Discovery Query.....	70
226	7.3.19 InverseRanges Discovery Query.....	71
227	7.3.20 SymmetricProperties Discovery Query.....	71
228	7.3.21 FunctionalProperties Discovery Query.....	72
229	7.3.22 InverseFunctionalProperties Discovery Query.....	72
230	7.3.23 Instances Discovery Query Discovery Query.....	73
231	8 OWL Profile References.....	74
232	8.1 Normative References.....	74
233	8.2 Informative References.....	75
234		

Illustration Index

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

235

Index of Tables

236

1 Introduction

237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284

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.

285 1.1 Terminology

286 The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT,
287 RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in IETF RFC
288 2119 [RFC2111].

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

292 1.2 Conventions

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

- 295 • UML Diagrams

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

- 298 • Identifier Placeholders

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

304

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

306 1.3 Recommended Enhancements

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

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

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

2 OWL Overview

316

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

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

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

331 Within the scope of this document, only OWL Lite constructs are considered and in the rest of the
332 document, "OWL" is used to mean "OWL Lite" unless otherwise stated.

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

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

2.1 Semantic Web Languages upon which OWL is Layered

335

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

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

346

347 Table 1: Semantic Web namespace table

348

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#

349

350

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

354

355

356 2.2 OWL Lite Constructs

357 2.2.1 RDF Schema Features

- 358 • Class (Thing, Nothing)
- 359 • rdfs:subClassOf
- 360 • rdf:Property
- 361 • rdfs:subPropertyOf
- 362 • rdfs:domain
- 363 • rdfs:range
- 364 • Individual

365 2.2.2 (In)Equality

- 366 • equivalentClass
- 367 • equivalentProperty
- 368 • sameAs
- 369 • differentFrom
- 370 • AllDifferent
- 371 • distinctMembers

372 2.2.3 Property Characteristics

- 373 • ObjectProperty
- 374 • DatatypeProperty
- 375 • inverseOf
- 376 • TransitiveProperty
- 377 • SymmetricProperty
- 378 • FunctionalProperty
- 379 • InverseFunctionalProperty

380 2.2.4 Property Restrictions

- 381 • Restriction
- 382 • onProperty
- 383 • allValuesFrom
- 384 • someValuesFrom

385 2.2.5 Restricted Cardinality

- 386 • minCardinality (only 0 or 1)
- 387 • maxCardinality (only 0 or 1)
- 388 • cardinality (only 0 or 1)

389 2.2.6 Class Intersection

- 390 • intersectionOf

391 2.2.7 Versioning

- 392 • versionInfo
- 393 • priorVersion
- 394 • backwardCompatibleWith
- 395 • incompatibleWith
- 396 • DeprecatedClass
- 397 • DeprecatedProperty

398 2.2.8 Annotation Properties

- 399 • rdfs:label
- 400 • rdfs:comment
- 401 • rdfs:seeAlso
- 402 • rdfs:isDefinedBy
- 403 • AnnotationProperty
- 404 • OntologyProperty

405 2.2.9 Datatypes

- 406 • xsd datatypes

407 2.3 OWL DL Constructs

408 2.3.1 Class Axioms

- 409 • oneOf, dataRange
- 410 • disjointWith
- 411 • equivalentClass (applied to class expressions)
- 412 • rdfs:subClassOf (applied to class expressions)

413 2.3.2 Boolean Combinations of Class Expressions

- 414 • unionOf
- 415 • complementOf
- 416 • intersectionOf

417 2.3.3 Arbitrary Cardinality

- 418 • minCardinality
- 419 • maxCardinality
- 420 • cardinality

421 2.3.4 Filler Information

- 422 • hasValue

423

3 ebXML Registry Overview

424

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

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

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

3.1 Overview of [ebRIM]

430

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

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

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

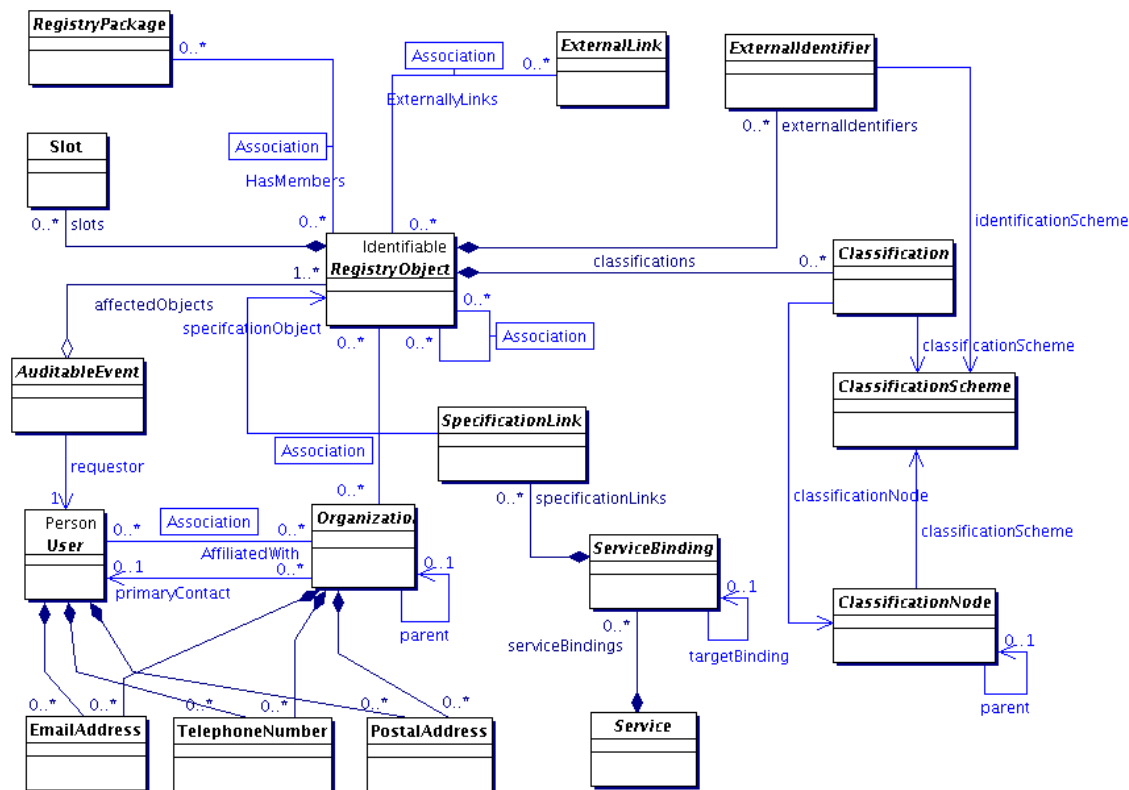


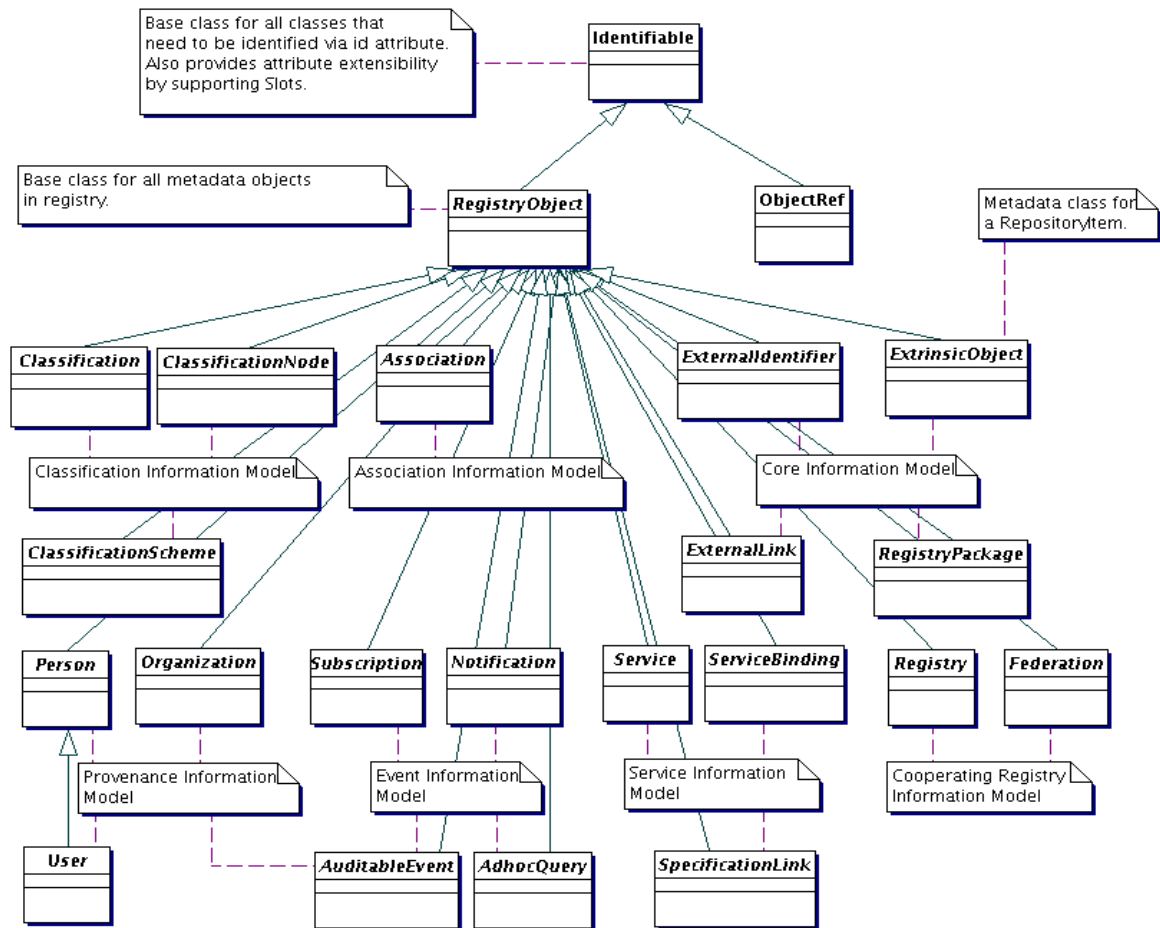
Figure 1: ebXML Registry Information Model, High Level Public View

438

439

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

444



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

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

448 3.1.1 RegistryObject

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

452 3.1.2 Object Identification

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

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

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

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

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

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

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

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

466

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

469

Listing 3: Example of lid Attribute

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

472

```
473 <rim:Organization id=${ACME_ORG_ID}  
474   lid="urn:acme:ACMEOrganization">  
475  
476   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}  
477     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}  
478     value="ACME"/>  
479   </rim:ExternalIdentifier>  
480  
481 </rim:Organization>
```

482

Listing 4: Example of ExternalIdentifier

483 3.1.3 Object Naming and Description

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

486

```
487 <rim:Organization id=${ACME_ORG_ID}  
488   lid="urn:acme:ACMEOrganization">  
489  
490   <rim:Name>  
491     <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>  
492   </rim:Name>  
493   <rim:Description>  
494     <rim:LocalizedString value="ACME is a provider of Java software."  
495       xml:lang="en-US"/>  
496   </rim:Description>  
497  
498   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}  
499     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}  
500     value="ACME"/>  
501   </rim:ExternalIdentifier>  
502 </rim:Organization>
```

503

Listing 5: Example of Name and Description

504

505 3.1.4 Object Attributes

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

508 3.1.4.1 Slot Attributes

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

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

515

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

```

518 <rim:Slot name="urn:acme:slot:NASDAQSymbol">
519   <rim:ValueList>
520     <rim:Value>ACME</rim:Value>
521   </rim:ValueList>
522 </rim:Slot>
523
524   <rim:Name>
525     <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
526   </rim:Name>
527   <rim:Description>
528     <rim:LocalizedString value="ACME makes Java. Provider of free Java
529 software."
530     xml:lang="en-US"/>
531   </rim:Description>
532   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
533     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
534     value="ACME"/>
535   </rim:ExternalIdentifier>
536 </rim:Organization>
537

```

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.

```

544
545 <rim:Organization id=${ACME_ORG_ID}
546   lid="urn:acme:ACMEOrganization">
547   <rim:Slot name="urn:acme:slot:NASDAQSymbol">
548     <rim:ValueList>
549       <rim:Value>ACME</rim:Value>
550     </rim:ValueList>
551   </rim:Slot>
552   <rim:Name>
553     <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
554   </rim:Name>
555   <rim:Description>
556     <rim:LocalizedString value="ACME makes Java. Provider of free Java
557 software." xml:lang="en-US"/>
558   </rim:Description>
559   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
560     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
561     value="ACME"/>
562   </rim:ExternalIdentifier>
563
564   <!--Classify Organization as a Software Publisher using NAICS Taxonomy-->
565   <rim:Classification id=${CLASSIFICATION_ID}
566     classificationNode=${NAICS_SOFTWARE_PUBLISHER_NODE_ID}
567     classifiedObject=${ACME_ORG_ID}>
568
569 </rim:Organization>

```

Listing 7: Example of Object Classification

3.1.6 Object Association

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

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

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

580 ml

581 [ebRIM] allows this scheme to be extensible.

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

585

```
586 <rim:Association  
587   id=${ASSOCIATION_ID}  
588   associationType=${CANONICAL_ASSOCIATION_TYPE_OFFERS_SERVICE_ID}  
589   sourceObject=${ACME_ORG_ID}  
590   targetObject=${ACME_SERVICE1_ID}/>
```

591

Listing 8: Example of Object Association

592 3.1.7 Object References To Web Content

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

598

```
599 <rim:ExternalLink externalURI="http://www.acme.com"  
600   id=${ACME_WEBSITE_EXTERNAL_ID}>  
601 <rim:Association  
602   id=${EXTERNALLYLINKS_ASSOCIATION_ID}  
603   associationType=${CANONICAL_ASSOCIATION_TYPE_EXTERNALLY_LINKS_ID}  
604   sourceObject=${ACME_WEBSITE_EXTERNAL_ID}  
605   targetObject=${ACME_ORG_ID}/>
```

606

Listing 9: Example of Reference to Web Content Using ExternalLink

607 3.1.8 Object Packaging

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

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

615

```
616 <rim:RegistryPackage  
617   id=${ACME_SERVICES_PACKAGE_ID}>  
618   <rim:RegistryObjectList>  
619     <rim:ObjectRef id=${ACME_SERVICE1_ID}>  
620     <rim:RegistryPackage  
621       id=${ACME_PURCHASING_SERVICES_PACKAGE_ID}>  
622       <rim:ObjectRef id=${ACME_PURCHASING_SERVICE1_ID}>  
623       <rim:ObjectRef id=${ACME_PURCHASING_SERVICE2_ID}>  
624     </rim:RegistryPackage>  
625     <rim:RegistryPackage  
626       id=${ACME_HR_SERVICES_PACKAGE_ID}>  
627       <rim:ObjectRef id=${ACME_HR_SERVICE1_ID}>  
628       <rim:ObjectRef id=${ACME_HR_SERVICE2_ID}>  
629     </rim:RegistryPackage>  
630   </rim:RegistryObjectList>  
631 </rim:RegistryPackage>
```

632

Listing 10: Example of Object Packaging Using RegistryPackages

633 3.1.9 ExtrinsicObject

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

638 3.1.10 Service Description

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

642 3.2 Overview of [ebRS]

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

4 Representing OWL Lite Constructs in ebRIM

645

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

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

654

```
655 ClassScheme (id, home, lid, objectType, status, versionName, comment_,...);  
656  
657 ClassificationNode(accessControlPolicy, id, lid, home, objectType, code, parent,  
658 path,versionName, comment_...)  
659  
660 Association(accessControlPolicy, id, lid, home, objectType, associationType,  
661 sourceObject, targetObject, isConfirmedBySourceOwner,versionName, comment_  
662 isConfirmedByTargetOwner,...)  
663  
664 Name_(charset, lang, value, parent,...)  
665  
666 Classification (id, objectType, lid, home, classificationNode, versionName,  
667 comment_, classificationScheme, classifiedObject, nodeRepresentation,...);  
668  
669 ExtrinsicObject (id, lid, home, objectType,...)
```

670

ebXML Registry Relations

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

4.1 Representing RDF Schema Features in ebRIM

674

4.1.1 owl:Class → rim:ClassificationNode

675

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

677

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

680

Example owl:Class

681

```
682 <ClassificationScheme id=${GeographicalEntity}  
683 name="GeographicalEntity"/>  
684  
685 <rim:ClassificationNode id=${City} code='City'  
686 parent=${GeographicalEntity}>  
687 </rim:ClassificationNode>
```

688

Example Corresponding ebRIM construct ClassificationNode

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

4.1.2 rdf:Property → rim:Association Type HasProperty

691

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

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

696

```
697 <rdf:Property rdf:ID="hasAirport">  
698   <rdfs:domain rdf:resource="#City"/>  
699   <rdfs:range rdf:resource="#AirPort"/>  
700 </rdf:Property>
```

701

Example rdf:Property

702

```
703 <rim:Association id=${hasAirport}  
704 associationType='urn:oasis:names:tc:ebxml-  
705 regrep:profile:webontology:AssociationType:OWL:HasProperty'  
706   sourceObject= ${city}  
707   targetObject=${Airport} >  
708 </rim:Association>
```

709

Example: ebRIM construct Association corresponding to rdf:Property

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

712 4.1.3 rdfs:subPropertyOf → rim:Association Type SubPropertyOf

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

716

```
717 <rdf:Property rdf:ID="creditCardPayment">  
718   <rdfs:subPropertyOf rdf:Resource="#paymentMethods"/>  
719 </rdf:Property>
```

720

Example rdfs:subPropertyOf

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

723

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

727

```
728 <rim:Association id=${subPropertyOfID}  
729 associationType='urn:oasis:names:tc:ebxml-  
730 regrep:profile:webontology:AssociationType:OWL:SubPropertyOf '  
731   sourceObject= ${creditCardPayment} targetObject=${paymentMethods} >  
732 </rim:Association>
```

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

735 4.1.4 rdfs:subClassOf → rim:Association Type SubClassOf

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

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

```
746  
747 <owl:Class rdf:ID="AirReservationServices">  
748   <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-  
749     s/1.0/Profile.owl#Profile"/>  
750   <rdfs:subClassOf rdf:resource="#AirServices"/>  
751 </owl:Class>
```

752 Example rdfs:subClassOf

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

```
757  
758 <rim:Association id=${subClassOf}  
759 associationType='urn:oasis:names:tc:ebxml-  
760 regrep:profile:webontology:AssociationType:OWL:SubClassOf'  
761 sourceObject= ${AirReservationServices} targetObject=${OWL-S_Profile} >  
762 </rim:Association>  
763 <rim:Association id=${subClassOf2}  
764 associationType='urn:oasis:names:tc:ebxml-  
765 regrep:profile:webontology:AssociationType:OWL:SubClassOf'  
766 sourceObject= ${AirReservationServices} targetObject=${AirServices} >  
767 </rim:Association>
```

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

772 4.1.5 owl:Individual → rim:ExtrinsicObject

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

```
776  
777 <owl:Class rdf:ID="TravelWebService">  
778 </owl:Class>  
779  
780 <owl:ObjectProperty rdf:ID="paymentMethod">  
781   <rdfs:domain rdf:resource="#TravelWebService"/>  
782   <rdfs:range rdf:resource="#PossiblePaymentMethods"/>  
783 </owl:ObjectProperty >
```

784 Example owl:Class example

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

```
789  
790 <TravelWebService rdf:ID="MyTravelWebService">  
791   <paymentMethod> Cash </paymentMethod>  
792 </TravelWebService>
```

793 Example owl:Individual example

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

797 4.2 Representing OWL (In)Equality Constructs in ebXML RIM

798 4.2.1 owl:equivalentClass, owl:equivalentProperty → rim:Association Type 799 EquivalentTo

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

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

807 4.2.2 owl:sameAs → rim:Association Type SameAs

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

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

813

```
814 <rdf:Description rdf:about="#MyAirReservationService">  
815   <owl:sameAs rdf:resource="#THYAirReservationService"/>  
816 </rdf:Description>
```

817

Example owl:sameAs

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

```
820 <rim:Association id=${sameAs1} associationType='urn:oasis:names:tc:ebxml-  
821   regrep:profile:webontology:AssociationType:OWL:SameAs'  
822   sourceObject=${MyAirReservationService} targetObject=${THYAirReservationService}  
823 >  
824 </rim:Association>
```

825

Example Corresponding ebRIM construct Association

826

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

829 4.2.3 owl:differentFrom → rim:Association Type DifferentFrom

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

834

```
835 <rdf:Description rdf:about="#MyAirReservationService">  
836   <owl:differentFrom rdf:resource="#THYAirReservationService"/>  
837 </rdf:Description>
```

838

Example owl:differentFrom

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


```

842 <rim:Association id=${differentFrom1}
843 associationType='urn:oasis:names:tc:ebxml-
844 regrep:profile:webontology:AssociationType:OWL:DifferentFrom'
845 sourceObject= ${MyAirReservationService}
846 targetObject=${THYAirReservationService} >
847 </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:

```

857 <owl:AllDifferent>
858   <owl:distinctMembers rdf:parseType="Collection">
859     <WebService rdf:about="#MyCarService"/>
860     <WebService rdf:about="#MyFlightService"/>
861     <WebService rdf:about="#MyHotelService"/>
862   </owl:distinctMembers>
863 </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.

```

869
870 <rim:RegistryPackage id = ${Collection} >
871   <rim:Slot name=urn:oasis:names:tc:ebxml-
872 regrep:profile:webontology:slot:packagetype>
873   <rim:ValueList>
874     <rim:Value>allDifferent</rim:Value>
875   </rim:ValueList>
876 </rim:Slot>
877 </rim:RegistryPackage>
878 <rim:Association id = ${HasMemberRegistryPackageAssoc1}
879 associationType = "urn:oasis:names:tc:ebxml-
880 regrep:AssociationType:HasMember" sourceObject = ${Collection}
881 targetObject = ${MyCarService} />
882 <rim:Association id = ${HasMemberRegistryPackageAssoc2}
883 associationType = "urn:oasis:names:tc:ebxml-regrep:HasMember"
884 sourceObject = ${Collection}
885 targetObject = ${MyFlightService} />
886
887 <rim:Association id = ${HasMemberRegistryPackageAssoc3}
888 associationType = "urn:oasis:names:tc:ebxml-regrep:HasMember"
889 sourceObject = ${Collection}
890 targetObject = ${MyHotelService} />

```

Example Corresponding ebRIM Representation

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

893 4.3 Representing OWL Property Characteristics in ebRIM

894 4.3.1 owl:ObjectProperty → rim:Association Type objectProperty

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

898

```
899 <owl:ObjectProperty rdf:ID="hasAirport">  
900   <rdfs:domain rdf:resource="#City"/>  
901   <rdfs:range rdf:resource="#AirPort"/>  
902 </owl:ObjectProperty>
```

903

Example owl:ObjectProperty

904

```
905 <rim:Association id=${hasAirport} associationType='urn:oasis:names:tc:ebxml-  
906   regrep:profile:webontology:AssociationType:OWL:ObjectProperty'  
907   sourceObject= ${City} targetObject=${Airport} >  
908 </rim:Association>
```

909

Example Corresponding ebRIM construct Association

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

913 4.3.2 owl:DatatypeProperty → rim:Association Type DatatypeProperty

914 Similarly, to represent OWL DatatypeProperty in ebXML, a new Association Type called
915 "DatatypeProperty" MUST be defined. Consider the following example which defines an datatype property
916 "hasPrice" whose domain is the "AirReservationServices" and whose range is "XMLSchema
917 nonNegativeInteger". How OWL XML Schema types are handled in ebXML RIM is described in Section
918 4.9.

```
919 <owl:DatatypeProperty rdf:ID="hasPrice">  
920   <rdfs:domain rdf:resource="#AirReservationServices"/>  
921   <rdfs:range  
922   rdf:resource="http://www.w3.org/2001/XMLSchema/nonNegativeInteger"/>  
923 </owl:DatatypeProperty>
```

924

Example owl:DatatypeProperty

925

```
926 <rim:Association id=${hasPrice}  
927   associationType='urn:oasis:names:tc:ebxml-  
928   regrep:profile:webontology:AssociationType:OWL:DatatypeProperty'  
929   sourceObject= ${AirReservationServices}  
930   targetObject=urn:www.w3.org:2001/XMLSchema:nonNegativeInteger >  
931 </rim:Association>
```

932

Example Corresponding ebRIM construct Association

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

935 4.3.3 owl:TransitiveProperty → rim:Association Type TransitiveProperty

936 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)
937 [McGuinness, Harmelen]. Transitive property is a subproperty of ObjectProperty and MUST be defined as
938 a new Association Type called "TransitiveProperty" in ebRIM.

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

941

```

942 <owl:ObjectProperty rdf:ID="succeeds">
943   <rdf:type rdf:resource="#owl:TransitiveProperty" />
944   <rdfs:domain rdf:resource="#TravelWebService" />
945   <rdfs:range rdf:resource="#TravelWebService" />
946 </owl:ObjectProperty>

```

947 **Example owl:TransitiveProperty**

```

948
949 <rim:Association id=${succeeds}
950 associationType='urn:oasis:names:tc:ebxml-
951 regrep:profile:webontology:AssociationType:OWL:TransitiveProperty'
952 sourceObject= ${TravelWebService} targetObject=${TravelWebService} >
953 </rim:Association>

```

954 **Example Corresponding ebRIM construct Association**

955 Assume the following two definitions which declare three Web service instances from TravelWebService
 956 class where "MyHotelAvailabilityService" service succeeds "MyAirReservationService" and
 957 "MyInsuranceService" succeeds "MyHotelAvailabilityService". Since "succeeds" is a transitive property, it
 958 follows that "MyInsuranceService" succeeds "MyAirReservationService" although this fact is not explicitly
 959 stated.

960

```

961 <TravelWebService rdf:ID="MyHotelAvailabilityService">
962   <succeeds rdf:resource="#MyAirReservationService" />
963 </TravelWebService>
964
965 <TravelWebService rdf:ID="MyInsuranceService">
966   <succeeds rdf:resource="#MyHotelAvailabilityService" />
967 </TravelWebService>

```

968 **Example owl:TransitiveProperty instances**

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

972 **4.3.4 owl:inverseOf → rim:Association Type InverseOf**

973 In OWL, one property may be stated to be the inverse of another property. If the property P1 is stated to
 974 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
 975 P1 property [McGuinness, Harmelen].

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

979

```

980 <owl:ObjectProperty rdf:ID="precedes">
981   <owl:inverseOf rdf:resource="#succeeds" />
982 </owl:ObjectProperty>

```

983 **Example owl:inverseOf Property**

984

```

985 <rim:Association id=${inverseOf1}
986 associationType='urn:oasis:names:tc:ebxml-
987 regrep:profile:webontology:AssociationType:OWL:InverseOf'
988 sourceObject= ${precedes} targetObject=${succeeds} >
989 </rim:Association>

```

990 **Example Corresponding ebRIM construct Association**

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

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

996 Alternatively, one might use the additional semantics that this profile supports would be to cause inferred
997 information to be produced and stored along with new data as that new data was inserted into the reg/rep.
998 There is a trade off here: in this way, the extra work of inferring is only done at insertion/update time,
999 instead of at query time. However, an insertion or an update will require all the inferred data to be inserted
1000 whether it will be used or not and hence will cause considerable maintenance overhead.

1001 4.3.5 owl:SymmetricProperty → rim:Association Type SymmetricProperty

1002 In OWL, if a property is symmetric, then if the pair (x,y) is an instance of the symmetric property P, then
1003 the pair (y,x) is also an instance of P [McGuinness, Harmelen]. Symmetric property is a subproperty of
1004 ObjectProperty in OWL. Consider the OWL class “WebService” and the “complements” symmetric
1005 property:

```
1006 <owl:Class rdf:ID="WebService">  
1007   <rdfs:subClassOf  
1008     rdf:resource="http://www.w3.org/2000/01/rdfschema#Resource"/>  
1009 </owl:Class>  
1010 <owl:SymmetricProperty rdf:ID="complements">  
1011   <rdfs:domain rdf:resource="#WebService"/>  
1012   <rdfs:range rdf:resource="#WebService"/>  
1013 </owl:SymmetricProperty>
```

1014 Example owl:SymmetricProperty

```
1015 <rim:Association id=${complements}  
1016 associationType='urn:oasis:names:tc:ebxml-  
1017 regrep:profile:webontology:AssociationType:OWL:SymetricProperty'  
1018 sourceObject= ${WebService} targetObject=${WebService} >  
1019 </rim:Association>
```

1020 Example Corresponding ebRIM construct Association

1021 Given that HotelReservationWebService complements AirReservationWebService, it is possible to
1022 deduce that AirReservationWebService complements HotelReservationWebService.

1023 owl:SymmetricProperty MUST be defined as a new type of Association in ebRIM called
1024 “SymmetricProperty”. Furthermore the adhoc query presented in Section 6.20 MUST be available in the
1025 Registry to retrieve symmetric Associations of a ClassificationNode.

1026 4.3.6 owl:FunctionalProperty → rim:Association Type FunctionalProperty

1027 In OWL, if a property is a FunctionalProperty, then it has no more than one value for each individual (it
1028 may have no values for an individual) [McGuinness, Harmelen]. The range of a FunctionalProperty can be
1029 either an Object or a datatype. Consider, for example, the “hasPrice” Functional property which has a
1030 unique price:

```
1031 <owl:DatatypeProperty rdf:ID="hasPrice">  
1032   <rdf:type rdf:resource="&owl;FunctionalProperty" />  
1033   <rdfs:domain rdf:resource="#AirReservationServices"/>  
1034   <rdfs:range  
1035     rdf:resource="http://www.w3.org/2001/XMLSchema/nonNegativeInteger"/>  
1036 </owl:DatatypeProperty>
```

1037 Example owl:FunctionalProperty

```
1038 <rim:Association id=${hasPrice}  
1039 associationType='urn:oasis:names:tc:ebxml-  
1040 regrep:profile:webontology:AssociationType:OWL:FunctionalProperty'  
1041 sourceObject= ${AirReservationServices}  
1042 targetObject=${uurn:www.w3.org:2001/XMLSchema:nonNegativeInteger} >  
1043 </rim:Association>
```

1044 Example Corresponding ebRIM construct Association

1045 ebXML RIM MUST contain a new Association Type called “FunctionalProperty” to express this semantics.
1046 Furthermore the he adhoc query presented in Section 6.21 MUST be available in the Registry to retrieve

1047 functional Associations of a ClassificationNode.

1048 4.3.7 owl:InverseFunctionalProperty → rim:Association Type 1049 InverseFunctionalProperty

1050 In OWL, if a property is inverse functional then the inverse of the property is functional. Thus the inverse
1051 of the property has at most one value for each individual [McGuinness, Harmelen]. InverseFunctional
1052 properties (IFPs) are like keys. An individual filling the range role in an inverseFunctional property
1053 instance identifies the individual in the domain role of that same property instance. In other words, if a
1054 semantic web tool encounters two individuals with the same value for an inverseFunctional property, it
1055 can be inferred that they are actually the same individual.

1056 As an example, the ObjectProperty “finalDestination” indicates that each flight arrives to only one airport
1057 as its final destination.

```
1058 <owl:ObjectProperty rdf:ID="finalDestination">  
1059   <rdf:type rdf:resource="&owl;InverseFunctionalProperty" />  
1060   <rdfs:domain rdf:resource="#Airport"/>  
1061   <rdfs:range rdf:resource="#Flight"/>  
1062 </owl:ObjectProperty>
```

1063 Example owl:InverseFunctionalProperty

```
1064 <rim:Association id=${finalDestination}  
1065 associationType='urn:oasis:names:tc:ebxml-  
1066 regrep:profile:webontology:AssociationType:OWL:InverseFunctionalProperty'  
1067   sourceObject= ${Airport} targetObject=${Flight} >  
1068 </rim:Association>
```

1069 Example Corresponding ebRIM construct Association

1070 ebRIM MUST contain a new Association Type called “InverseFunctionalProperty” to express this
1071 semantics. Furthermore the adhoc query presented in Section 6.22 MUST be available in the Registry to
1072 retrieve inverse functional Associations of a ClassificationNode.

1073 4.4 OWL Property Restrictions in ebXML RIM

1074 An important construct of OWL is "owl:Restriction". In RDF, a property has a global scope, that is, no
1075 matter what class the property is applied to, the range of the property is the same. "owl:Restriction", on the
1076 other hand, has a local scope; restriction is applied on the property within the scope of the class where it is
1077 defined. This makes property definitions more reusable by factoring out class specific characteristics of
1078 the property into the class description.

1079 For example, we may define a property "paymentMethod" for travel Web services in general and we may
1080 state that the range of this property is the class "PossiblePaymentMethods". Then, for
1081 "AirReservationServices", we may wish to restrict "paymentMethod" property to, say, "CreditCard" class as
1082 demonstrated in the following two examples:

```
1083  
1084 <owl:ObjectProperty rdf:ID="paymentMethod">  
1085   <rdfs:domain rdf:resource="#TravelWebService"/>  
1086   <rdfs:range rdf:resource="#PossiblePaymentMethods"/>  
1087 </owl:ObjectProperty >
```

1088 Example owl:ObjectProperty “paymentMethod”

```
1089  
1090 <owl:Class rdf:ID="AirReservationServices">  
1091   <rdfs:subClassOf>  
1092     <owl:Restriction>  
1093       <owl:onProperty rdf:resource="#paymentMethod"/>  
1094       <owl:allValuesFrom rdf:resource= "#CreditCard"/>  
1095     </owl:Restriction>  
1096   </rdfs:subClassOf>  
1097 </owl:Class>
```

1098 Example owl:Restriction on ObjectProperty “paymentMethod”

1099 A new Association Type of "restriction" SHOULD be defined to represent OWL restriction. A slot of this
1100 Association Type SHOULD indicate the whether the restriction is "allValuesFrom" or "someValuesFrom".
1101 When such restriction is submitted to the system, the registry MUST create a new Association instance,
1102 say, "paymentMethod_1" of AssociationType "ObjectProperty" is created whose sourceObject is
1103 "AirReservationServices" and the targetObject is "CreditCard". "paymentMethod_1" Association instance
1104 is related with the "paymentMethod" Association instance by using an instance of the Association Type
1105 "Restriction" as shown in the following example:

1106

```
1107 <rim:Association id = ${paymentMethod_1}  
1108             associationType =  
1109 "urn:oasis:names:tc:ebxml-  
1110 regrep:profile:webontology:AssociationType:OWL:ObjectProperty"  
1111             sourceObject = ${AirReservationServices}  
1112             targetObject = ${CreditCard}>  
1113 </rim:Association>  
1114  
1115 <rim:Association id = ${paymentMethodRestriction}  
1116             associationType =  
1117 "urn:oasis:names:tc:ebxml-  
1118 regrep:profile:webontology:AssociationType:OWL:Restriction"  
1119             sourceObject = ${paymentMethod}  
1120             targetObject = ${paymentMethod_1}>  
1121     <rim:Slot name="urn:oasis:names:tc:ebxml-  
1122 regrep:profile:webontology:slot:restrictionType">  
1123         <rim:ValueList>  
1124             <rim:Value>allValuesFrom</rim:Value>  
1125         </rim:ValueList>  
1126     </rim:Slot>  
1127 </rim:Association>
```

1128 Example Handling owl:Restriction in ebXML Registry

1129 Obviously, this serves the purpose of reusing the "paymentMethod" property. Otherwise, a new property
1130 "paymentMethodCC" can be defined between "AirReservationServices" and the "CreditCard" classes as
1131 shown in the following:

1132

```
1133 <owl:ObjectProperty rdf:ID="paymentMethodCC">  
1134   <rdfs:domain rdf:resource="#AirReservationServices"/>  
1135   <rdfs:range rdf:resource="#CreditCard"/>  
1136 </owl:ObjectProperty >
```

1137 Example owl:ObjectProperty "paymentMethodCC"

1138 4.5 Representing OWL Restricted Cardinality in ebXML RIM

1139 4.5.1 owl:minCardinality (only 0 or 1)

1140 In OWL, cardinality is stated on a property with respect to a particular class. If a minCardinality of 1 is
1141 stated on a property with respect to a class, then any instance of the class will have at least one value for
1142 the restricted property. This restriction is another way of saying that the property is required to have a
1143 value for all instances of the class. In OWL Lite, the only minimum cardinalities allowed are 0 or 1. A
1144 minimum cardinality of zero on a property just states (in the absence of any more specific information)
1145 that the property is optional with respect to a class [McGuinness, Harmelen].

1146 Consider for example the following OWL code which states that each instance of a "WebService" class
1147 must have at least one price:

```
1148 <owl:Class rdf:ID="WebService">  
1149   <rdfs:subClassOf>  
1150     <owl:Restriction>  
1151       <owl:onProperty rdf:resource="#hasPrice"/>  
1152       <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">  
1153 1 </owl:minCardinality>  
1154     </owl:Restriction>
```

```
1155     </rdfs:subClassOf>
1156 </owl:Class>
```

1157 **Example owl:minCardinality**

1158 In ebXML RIM, cardinalities of Association Types MUST be defined by associating a minCardinality slot
1159 with the Association Types as shown in the following example:

1160

```
1161 <rim:Association id = ${hasPriceMinCardinalityRestriction}
1162 associationType = "urn:oasis:names:tc:ebxml-
1163 regrep:profile:webontology:AssociationType:OWL:ObjectProperty"
1164 sourceObject = ${WebService}
1165 targetObject = ${Price}>
1166     <rim:Name>
1167         <rim:LocalizedString value = 'hasPrice' />
1168     </rim:Name>
1169     <rim:Slot name="urn:oasis:names:tc:ebxml-
1170 regrep:profile:webontology:slot:minCardinality">
1171         <rim:ValueList>
1172             <rim:Value>1</rim:Value>
1173         </rim:ValueList>
1174     </rim:Slot>
1175 </rim:Association>
```

1176 **Example Representing owl:minCardinality in ebRIM**

1177 **4.5.2 owl:maxCardinality (only 0 or 1)**

1178 In OWL, cardinality is stated on a property with respect to a particular class. If a maxCardinality of 1 is
1179 stated on a property with respect to a class, then any instance of that class will be related to at most one
1180 individual by that property. A maxCardinality 1 restriction is sometimes called a functional or unique
1181 property. It may be useful to state that certain classes have no values for a particular property. This
1182 situation is represented by a maximum cardinality of zero on the property [McGuinness, Harmelen].

1183 Consider for example the following OWL code which states that each instance of a "WebService" class
1184 can have at most one price:

```
1185 <owl:Class rdf:ID="WebService">
1186     <rdfs:subClassOf>
1187         <owl:Restriction>
1188             <owl:onProperty rdf:resource="#hasPrice"/>
1189             <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
1190 1 </owl:maxCardinality>
1191         </owl:Restriction>
1192     </rdfs:subClassOf>
1193 </owl:Class>
```

1194 **Example owl:maxCardinality**

1195 In ebXML RIM, cardinalities of Association Types MUST be defined by associating a maxCardinality slot
1196 with the Association Types as shown in the following example:

1197

```
1198 <rim:Association id = ${hasPriceMaxCardinalityRestriction}
1199 associationType = "urn:oasis:names:tc:ebxml-
1200 regrep:profile:webontology:AssociationType:OWL:ObjectProperty"
1201 sourceObject = ${WebService}"
1202 targetObject = ${Price}>
1203     <rim:Name>
1204         <rim:LocalizedString value = 'hasPrice' />
1205     </rim:Name>
1206     <rim:Slot name="urn:oasis:names:tc:ebxml-
1207 regrep:profile:webontology:slot:maxCardinality">
1208         <rim:ValueList>
1209             <rim:Value>1</rim:Value>
1210         </rim:ValueList>
1211     </rim:Slot>
```

1212 </rim:Association>

1213 Example Representing owl:maxCardinality in ebRIM

1214 4.5.3 owl:cardinality (only 0 or 1)

1215 In OWL Lite, cardinality is provided as a convenience when it is useful to state that a property on a class
1216 has both minCardinality 0 and maxCardinality 0 or both minCardinality 1 and maxCardinality 1
1217 [McGuinness, Harmelen].

1218 Consider for example the following OWL code which states that each instance of a “WebService” class
1219 must have exactly one price:

```
1220 <owl:Class rdf:ID="WebService">  
1221   <rdfs:subClassOf>  
1222     <owl:Restriction>  
1223       <owl:onProperty rdf:resource="#hasPrice"/>  
1224       <owl:Cardinality rdf:datatype="&xsd;nonNegativeInteger"> 1  
1225     </owl:Cardinality>  
1226   </owl:Restriction>  
1227 </rdfs:subClassOf>  
1228 </owl:Class>
```

1229 Example owl:Cardinality

1230 In ebXML RIM, cardinalities of Association Types MUST be defined by associating a Cardinality slot with
1231 the Association Types as shown in the following example:

```
1232  
1233 <rim:Association id = ${hasPriceCardinalityRestriction}  
1234 associationType = "urn:oasis:names:tc:ebxml-  
1235 regrep:profile:webontology:AssociationType:OWL:ObjectProperty"  
1236 sourceObject = ${WebService}  
1237 targetObject = ${Price}>  
1238   <rim:Name>  
1239     <rim:LocalizedString value = 'hasPrice' />  
1240   </rim:Name>  
1241   <rim:Slot name="urn:oasis:names:tc:ebxml-  
1242 regrep:profile:webontology:slot:cardinality">  
1243     <rim:ValueList>  
1244       <rim:Value>1</rim:Value>  
1245     </rim:ValueList>  
1246   </rim:Slot>  
1247 </rim:Association>
```

1248 Example Representing owl:Cardinality in ebRIM

1249 4.6 Representing OWL Class Intersection in ebXML RIM

1250 OWL provides the means to manipulate class extensions using basic set operators. In OWL lite, only
1251 “owl:intersectionOf” is available which defines a class that consists of exactly all objects that belong to all
1252 the classes specified in the intersection definition. In the following example, "AirReservationServices" is
1253 defined as the intersection of "AirServices" and "ReservationServices":

```
1254  
1255 <owl:Class rdf:ID="AirReservationServices">  
1256   <owl:intersectionOf rdf:parseType="Collection">  
1257     <owl:Class rdf:about="#AirServices" />  
1258     <owl:Class rdf:about="#ReservationServices" />  
1259   </owl:intersectionOf>  
1260 </owl:Class>
```

1261 Example owl:intersectionOf

1262 In ebXML RIM "owl:intersectionOf" set operator MUST be represented as follows:

- 1263
- 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
- 1264

1265 multi-valued slot named as: "urn:oasis:names:tc:ebxml-
1266 regrep:profile:webontology:slot:intersectionOf".

1267

```
1268 <rim:ClassificationNode id = ${AirReservationServices} code =  
1269 "AirReservationServices">  
1270   <rim:Slot name=urn:oasis:names:tc:ebxml-  
1271     regrep:profile:webontology:slot:intersectionOf>  
1272     <rim:ValueList>  
1273       <rim:Value>${AirServices}</rim:Value>  
1274       <rim:Value>${ReservationServices}</rim:Value>  
1275     </rim:ValueList>  
1276   </rim:Slot>  
1277 </rim:ClassificationNode>  
1278
```

1279 **Example Defining Intersection of ClassificationNodes in ebRIM**

1280 When such a representation is used to create a complex class (a new ClassificationNode) in RIM, it
1281 becomes possible to infer that the objects (instances) classified by all of the classes
1282 (ClassificationNodes) specified in the Intersection definition, are also classified by this complex class. The
1283 adhoc query presented in Section 6.23 MUST be available in the ebXML Registry to retrieve the direct
1284 instances of the complex class and also the instances of the intersection of the classes.

1285

1286 **4.7 Representing OWL Versioning in ebXML RIM**

1287 **4.7.1 owl:versionInfo, owl:priorVersion**

1288 An owl:versionInfo statement generally has as its object a string giving information about this version, for
1289 example RCS/ CVS keywords. This statement does not contribute to the logical meaning of the ontology
1290 other than that given by the RDF(S) model theory [McGuinness, Harmelen].

1291 An owl:priorVersion statement contains a reference to another ontology. This identifies the specified
1292 ontology as a prior version of the containing ontology [McGuinness, Harmelen].

1293 In ebXML, since a RegistryObject MAY have several versions, a logical id (called lid) is also defined which
1294 is unique for different logical objects. However the lid attribute value MUST be the same for all versions of
1295 the same logical object. Therefore, almost for all the RegistryObjects the version information is kept
1296 through "versionName" and "comment" attributes of the "VersionInfo" ebRIM Class.

1297 "owl:version" information MUST be stored in the "versionName" and "comment" attributes of the
1298 VersionInfo ebRIM class.

1299 It should be noted that in freebXML implementation the versionInfo is flattened and the "versionName"
1300 and "comment_" are provided as direct attributes of database tables.

```
1301 <owl:Ontology rdf:about="">  
1302   <owl:versionInfo>v 1.17 2003/02/26 12:56:51 </owl:versionInfo>  
1303 </owl:Ontology>
```

1304 **Example owl:versionInfo**

```
1305 <rim:ClassificationScheme  
1306 lid= ${exampleOntology}  
1307 id=${exampleOntology} isInternal="true"  
1308 nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode">  
1309   <rim:versionInfo>  
1310     <rim:versionName>  
1311       <rim:LocalizedString charset="UTF-8" value="v 1.17 2003/02/26  
1312         12:56:51"/>  
1313     </rim:versionName>  
1314   </rim:versionInfo>  
1315 </rim:ClassificationScheme>
```

1316 **Example rim:versionName**

1317 4.8 Representing OWL Annotation Properties in ebXML RIM

1318 4.8.1 rdfs:label

1319 rdfs:label is an instance of rdf:Property that may be used to provide a human-readable version of a
1320 resource's name [Brickley, Guha].

1321 In ebXML RIM, human readable names of resources are provided through rim:Name. rdfs:label MUST be
1322 expressed through rim:Name.

1323

```
1324 <owl:Class rdf:ID="AirReservationServices">  
1325   <rdfs:label>Air Reservation Services</rdfs:label>  
1326 </owl:Class>
```

1327 Example rdfs:label

1328

```
1329 <rim:ClassificationNode id = ${AirReservationServices} code =  
1330 'AirReservationServices'  
1331   <rim:Name>  
1332     <rim:LocalizedString value = 'Air Reservation Services' />  
1333   </rim:Name>  
1334 </rim:ClassificationNode>
```

1335 Example rim:Name

1336 4.8.2 rdfs:comment

1337 rdfs:comment is an instance of rdf:Property that may be used to provide a human-readable description of
1338 a resource [Brickley, Guha].

1339 In ebXML RIM, this construct MUST be expressed through rim:Description.

1340

```
1341 <owl:Class rdf:ID="AirReservationServices">  
1342   <rdfs:comment>Open Travel Alliance Air Reservation Services  
1343   </rdfs:comment>  
1344 </owl:Class>
```

1345 Example rdfs:comment

1346

```
1347 <rim:ClassificationNode id = ${AirReservationServices} code =  
1348 'AirReservationServices'  
1349   <rim:Description>  
1350     <rim:LocalizedString value = 'Open Travel Alliance Air  
1351 Reservation Services' />  
1352   </rim:Description>  
1353 </rim:ClassificationNode>
```

1354 Example: rim:Description

1355 4.8.3 rdfs:seeAlso

1356 rdfs:seeAlso is an instance of rdf:Property that is used to indicate a resource that might provide additional
1357 information about the subject resource [Brickley, Guha].

1358 This construct MUST be expressed in ebXML RIM by defining a new Association Type called "SeeAlso" to
1359 express this semantics.

1360

```
1361 <owl:Class rdf:ID="AirReservationServices">  
1362   <rdfs:seeAlso rdf:resource="http://www.opentravel.org" />  
1363 </owl:Class>
```

1364 Example rdfs:seeAlso

```

1365 <rim:ClassificationNode id = ${AirReservationServices} code =
1366 'AirReservationServices'>
1367 </rim:ClassificationNode>
1368
1369 <rim:ExternalLink id = ${exampleExternalLink}
1370     externalURI= "http://www.opentravel.org" >
1371 </rim:ExternalLink>
1372
1373 <rim:Association id = ${seeAlsoID}
1374     associationType = 'urn:oasis:names:tc:ebxml-
1375 regrep:profile:webontology:AssociationType:OWL:SeeAlso'
1376     sourceObject = ${AirReservationServices}
1377     targetObject = ${exampleExternalLink} />

```

1378 **Example rim:seeAlsoExternalLink**

1379 4.9 OWL Datatypes in ebXML RIM

1380 OWL allows the use of XML Schema datatypes to describe part of the datatype domain by simply
1381 including their URIs within an OWL ontology [McGuinness, Harmelen]. In ebXML, XML Schema datatypes
1382 MAY be used by providing an external link from the registry.

1383 The following example demonstrates how XML Schema datatype “integer” can be referenced through an
1384 ExternalLink whose id is 'urn:www.w3.org:2001/XMLSchema:integer' and how to define a
1385 DatatypeProperty, namely, “hasPrice”, whose target object is the defined to be ExternalLink 'integer':

```

1386 <rim:ExternalLink id = urn:www.w3.org:2001/XMLSchema:integer
1387     externalURI="http://www.w3.org/2001/XMLSchema#integer" >
1388     <rim:Name> <rim:LocalizedString value = "XML Schema integer"/>
1389     </rim:Name>
1390 </rim:ExternalLink>
1391 <rim:Association id = ${hasPrice} associationType =
1392 'urn:oasis:names:tc:ebxml-
1393 regrep:AssociationType:DatatypeProperty'
1394     sourceObject = ${AirReservationServices}
1395     targetObject = urn:www.w3.org:2001/XMLSchema:integer >
1396     <rim:Name> <rim:LocalizedString value ="hasPrice"/></rim:Name>
1397 </rim:Association>
1398

```

1399 **Example Corresponding ebRIM construct Association**

1400 5 Cataloging Service Profile

1401 The ebXML Registry provides the ability for a content cataloging service to be configured for any type of
1402 content. The cataloging service serves the following purposes:

- 1403 • Automates the mapping from the source information model (in this case OWL) to ebRIM. This
1404 hides the complexity of the mapping from the OWL publisher and eliminates the need for any
1405 special UI tools to be provided by the registry implementor for publishing OWL documents.
- 1406 • Selectively converts content into ebRIM compatible metadata when the content is cataloged after
1407 being published. The generated metadata enables the selected content to be used as
1408 parameter(s) in content specific parameterized queries.

1409 This section describes the cataloging service for cataloging OWL content.

1410 An OWL document, when published to an ebXML Registry implementing the OWL Profile, **MUST** be
1411 cataloged as specified in this section using a OWL Content Cataloging Service as defined by [ebRS].

1412 5.1 Invocation Control File

1413 The OWL cataloging service **MAY** optionally support an invocation control file that declaratively specifies
1414 the transforms necessary to catalog published OWL documents.

1415 5.2 Input Metadata

1416 The OWL cataloging service **MUST** be pre-configured to be automatically invoked when the following
1417 types of metadata are published, as defined by the [ebRS] specifications.

1418 These are the only types of metadata that **MAY** describe a OWL document being published:

- 1419 • An `ExtrinsicObject` whose `ObjectType` references the canonical OWL `ClassificationNode`
1420 specified in Section 7. The `ExtrinsicObject` **MUST** have an OWL document as its `RepositoryItem`.
- 1421 • An `ExternalLink` whose `ObjectType` references the canonical OWL `ClassificationNode` specified in
1422 Section 7. In case of `ExternalLink` the OWL document **MUST** be resolvable via a URL described
1423 by the value of the `externalURI` attribute of the `ExternalLink`. Recall that, in the `ExternalLink` case
1424 the OWL document is not stored in the repository.

1425

```
1426 <rim:ExtrinsicObject id="urn:acmeinc:ebxml:registry:3.0:owl">  
1427 ...  
1428 </rim:ExtrinsicObject>
```

1429 Example of ExtrinsicObject Input Metadata

1430

```
1431 <rim:ExternalLink  
1432 id="urn:acmeinc:ebxml:registry:3.0:owl"  
1433 externalURI="http://www.acme.com/owl/ebXMLRegistryService.owl"  
1434 >  
1435 ...  
1436 </rim:ExternalLink>
```

1437 Example of ExternalLink Input Metadata

1438 5.3 Input Content

1439 The OWL cataloging service expects an OWL document as its input content. The input content **MUST** be
1440 processed by the OWL cataloging service regardless of whether it is a `RepositoryItem` for an
1441 `ExtrinsicObject` or whether it is content external to repository that is referenced by an `ExternalLink`.

1442

1443 5.4 Output Metadata

1444 This section describes the metadata produced by the OWL cataloging service produces as output.

1445 5.4.1 owl:Class → rim:ClassificationNode

1446 The OWL Cataloging service MUST automatically produce a rim:ClassificationNode instance for each
1447 owl:class element within the input OWL or its imports, as specified in the owl:Class →
1448 rim:ClassificationNode mapping earlier in this document.

1449 5.4.2 rdf:Property → rim:Association Type HasProperty

1450 The OWL Cataloging service MUST automatically produce an rim:Association instance with
1451 associationType HasProperty for each rdf:Property element within the input OWL or its imports, as
1452 specified in the rdf:Property → rim:Association Type HasProperty mapping earlier in this document.

1453 5.4.3 rdfs:subPropertyOf → rim:Association Type SubPropertyOf

1454 The OWL Cataloging service MUST automatically produce an rim:Association instance with
1455 associationType SubPropertyOf for each rdfs:subPropertyOf element within the input OWL or its imports,
1456 as specified in the rdfs:subPropertyOf → rim:Association Type SubPropertyOf mapping earlier in this
1457 document.

1458 5.4.4 rdfs:subClassOf → rim:Association Type subClassOf

1459 The OWL Cataloging service MUST automatically produce an rim:Association instance with
1460 associationType subClassOf for each rdfs:subClassOf element within the input OWL or its imports, as
1461 specified in the rdfs:subClassOf → rim:Association Type subClassOf mapping earlier in this document.

1462 5.4.5 owl:Individual → rim:ExtrinsicObject

1463 The OWL Cataloging service MUST automatically produce rim:ExtrinsicObject instances for each
1464 owl:Individual element within the input OWL or its imports, as specified in the owl:Individual →
1465 rim:ExtrinsicObject mapping earlier in this document.

1466 5.4.6 owl:equivalentClass, owl:equivalentProperty → rim:Association Type 1467 EquivalentTo

1468 The OWL Cataloging service MUST automatically produce rim:Association instances with
1469 associationType EquivalentTo for each owl:equivalentClass or owl:equivalentProperty element within the
1470 input OWL or its imports, as specified in the owl:equivalentClass, owl:equivalentProperty →
1471 rim:Association Type EquivalentTo mapping earlier in this document.

1472 5.4.7 owl:sameAs → rim:Association Type SameAs

1473 The OWL Cataloging service MUST automatically produce rim:Association instances with
1474 associationType SameAs for each owl:sameAs element within the input OWL or its imports, as specified
1475 in the owl:sameAs → rim:Association Type SameAs mapping earlier in this document.

1476 5.4.8 owl:differentFrom → rim:Association Type DifferentFrom

1477 The OWL Cataloging service MUST automatically produce rim:Association instances with
1478 associationType DifferentFrom for each owl:differentFrom element within the input OWL or its imports, as
1479 specified in the owl:differentFrom → rim:Association Type DifferentFrom mapping earlier in this
1480 document.

1481 5.4.9 owl:AllDifferent → rim:RegistryPackage

1482 The OWL Cataloging service MUST automatically produce rim:RegistryPackage instances for each

1483 owl:AllDifferent element within the input OWL or its imports, as specified in the owl:AllDifferent →
1484 rim:RegistryPackage mapping earlier in this document.

1485 [5.4.10 owl:ObjectProperty → rim:Association Type ObjectProperty](#)

1486 The OWL Cataloging service MUST automatically produce rim:Association instances with
1487 associationType ObjectProperty for each owl:ObjectProperty element within the input OWL or its imports,
1488 as specified in the owl:ObjectProperty → rim:Association Type ObjectProperty mapping earlier in this
1489 document.

1490 [5.4.11 owl:DatatypeProperty → rim:Association Type DatatypeProperty](#)

1491 The OWL Cataloging service MUST automatically produce rim:Association instances with
1492 associationType datatypeProperty for each owl:DatatypeProperty element within the input OWL or its
1493 imports, as specified in the owl:DatatypeProperty → rim:Association Type datatypeProperty mapping
1494 earlier in this document.

1495 [5.4.12 owl:TransitiveProperty → rim:Association Type TransitiveProperty](#)

1496 The OWL Cataloging service MUST automatically produce rim:Association instances with
1497 associationType TransitiveProperty for each owl:TransitiveProperty element within the input OWL or its
1498 imports, as specified in the owl:TransitiveProperty → rim:Association Type TransitiveProperty mapping
1499 earlier in this document.

1500 [5.4.13 owl:inverseOf → rim:Association Type InverseOf](#)

1501 The OWL Cataloging service MUST automatically produce rim:Association instances with
1502 associationType InverseOf for each owl:inverseOf element within the input OWL or its imports, as
1503 specified in the owl:inverseOf → rim:Association Type InverseOf mapping earlier in this document.

1504 [5.4.14 owl:SymmetricProperty → rim:Association Type SymetricProperty](#)

1505 The OWL Cataloging service MUST automatically produce rim:Association instances with
1506 associationType SymetricProperty for each owl:SymetricProperty element within the input OWL or its
1507 imports, as specified in the owl:SymetricProperty → rim:Association Type SymetricProperty mapping
1508 earlier in this document.

1509 [5.4.15 owl:FunctionalProperty → rim:Association Type FunctionalProperty](#)

1510 The OWL Cataloging service MUST automatically produce rim:Association instances with
1511 associationType FunctionalProperty for each owl:FunctionalProperty element within the input OWL or its
1512 imports, as specified in the owl:FunctionalProperty → rim:Association Type FunctionalProperty mapping
1513 earlier in this document.

1514 [5.4.16 owl:InverseFunctionalProperty → rim:Association Type 1515 InverseFunctionalProperty](#)

1516 The OWL Cataloging service MUST automatically produce rim:Association instances with
1517 associationType InverseFunctionalProperty for each owl:InverseFunctionalProperty element within the
1518 input OWL or its imports, as specified in the owl:InverseFunctionalProperty → rim:Association Type
1519 InverseFunctionalProperty mapping earlier in this document.

1520 [5.4.17 owl:minCardinality \(only 0 or 1\)](#)

1521 The OWL Cataloging service MUST automatically add a slot with name minCardinality to the relevant
1522 rim:Association instances for each owl:minCardinality element within the input OWL or its imports, as
1523 specified in section 4.5.1 where how to represent owl:minCardinality is described.

1524 **5.4.18 owl:maxCardinality (only 0 or 1)**

1525 The OWL Cataloging service MUST automatically add a slot with name maxCardinality to the relevant
1526 rim:Association instances for each owl:maxCardinality element within the input OWL or its imports, as
1527 specified in section 4.5.2 where how to represent owl:maxCardinality is described.

1528 **5.4.19 owl:cardinality**

1529 The OWL Cataloging service MUST automatically add a slot with name cardinality to the relevant
1530 rim:Association instances for each owl:cardinality element within the input OWL or its imports, as
1531 specified in section 4.5.3 where how to represent owl:cardinality is described.

1532 **5.4.20 owl:intersectionOf**

1533 The OWL Cataloging service MUST automatically produce a rim:RegistryPackage and a rim:Association
1534 instances with type IntersectionOf for each owl:intersectionOf element within the input OWL or its imports,
1535 as specified in section 4.6 where how to represent owl:intersectionOf is described.

1536 **5.4.21 rdfs:label**

1537 The OWL Cataloging service MUST automatically produce a rim:Name instance for each rdfs:label
1538 element within the input OWL or its imports, as specified in section 4.8.1 where how to represent
1539 rdfs:label is described.

1540 **5.4.22 rdfs:comment**

1541 The OWL Cataloging service MUST automatically produce a rim:Description instance for each
1542 rdfs:comment element within the input OWL or its imports, as specified in section 4.8.2 where how to
1543 represent rdfs:comment is described.

1544 **5.4.23 rdfs:seeAlso**

1545 The OWL Cataloging service MUST automatically produce a rim:ExternalLink and a rim:Association with
1546 type SeeAlso instances for each rdfs:seeAlso element within the input OWL or its imports, as specified in
1547 section 4.8.3 where how to represent rdfs:seeAlso is described.

6 Discovery Profile

1548

1549 The ebXML Registry provides the ability for a user defined parameterized queries to be configured for
1550 each type of content. The queries may be as complex or simple as the discovery use case requires. The
1551 complexity of the parameterized queries may hidden from the registry client by storing them within the
1552 ebXML Registry as instances of the AdhocQuery class, and being invoked by simply providing their
1553 parameters. Query parameters are often pattern strings that may contain wildcard characters '%' (matches any number of characters) and '_' (matches exactly one character) as described by [ebRS].

1554
1555 An ebXML Registry SHOULD provide a graphical user interface that displays any configured
1556 parameterized query as a form which contains an appropriate field for entering each query parameter.

1557 This chapter defines the queries that MUST be supported by an ebXML Registry implementing the OWL
1558 Profile for processing the semantics provided in the OWL content. An implementation MAY also support
1559 additional discovery queries for OWL content, some of which have already identified in this section.

1560 The queries defined in this chapter are parameterized queries stored in the Registry as instances of the
1561 AdhocQuery type, in the same manner as any other RegistryObject.

1562 In the subsequent section each query is described simply in terms of its supported parameters that serve
1563 as its search criteria. The actual AdhocQuery instances are much more complex in comparison but they
1564 are not exposed to the client making the query. Details on these queries are specified canonically in
1565 section 7.3 .

1566 Some of the queries that are necessary to process the semantics involved in OWL documents requires
1567 SQL recursion mechanism which is available through SQL 99 Standard. Since SQL 92, does not support
1568 recursion mechanism, those queries are stated to be implemented optionally. Additionally for these types
1569 of discovery queries, references to the "stored procedures" are presented in Section 7.3 for the interested
1570 users.

6.1 All SuperProperties Discovery Query

1571
1572 As presented in Section 4.1.3, a new ebXML RIM Association Type called "SubPropertyOf" MUST be
1573 defined to represent rdfs:subPropertyOf in ebRIM. Such a semantic enhancement brings the following
1574 processing need: given a property, it should be possible to retrieve all of its super properties. This requires
1575 a recursion mechanism in SQL queries.

1576 The AllSuperProperties discovery query MAY be implemented by an ebXML Registry implementing this
1577 profile. It allows the discovery of all super properties of a given property instance (Association instance in
1578 ebXML terminology) recursively in a property hierarchy (hierarchy of Association Types) in an ebXML
1579 Registry implementation supporting recursion. The canonical query corresponding to this discovery query
1580 is presented in Section 7.3.1.

6.1.1 Parameter \$propertyName

1581
1582 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1583 value of Associations that have associationType of Property.

6.1.2 Example of All SuperProperties Discovery Query

1584
1585 The following example illustrates how to find all the super properties of a given property having a name
1586 containing "creditCardPayment" if the query is implemented as an AdHoc Query.

1587

```
1588 <rs:RequestSlotList>  
1589   <rim:Slot  
1590     name="urn:oasis:names:tc:ebxml-  
1591     regrep:3.0:rs:AdhocQueryRequest:queryId">  
1592     <rim:ValueList>  
1593       <rim:Value>urn:oasis:names:tc:ebxml-  
1594       regrep:profile:webontology:query:FindAllSuperProperties</rim:Value>  
1595     </rim:ValueList>  
1596   </rim:Slot>
```



```

1597     <rim:Slot name="urn:oasis:names:tc:ebxml-
1598 regrep:rs:AdhocQueryRequest:queryId">
1599         <rim:ValueList>
1600             <rim:Value>urn:oasis:names:tc:ebxml-
1601 regrep:profile:webontology:query:FindAllSuperProperties</rim:Value>
1602         </rim:ValueList>
1603     </rim:Slot>
1604     <rim:Slot name="$propertyName">
1605         <rim:ValueList>
1606             <rim:Value>%creditCardPayment%</rim:Value>
1607         </rim:ValueList>
1608     </rim:Slot>
1609 </rs:RequestSlotList>
1610
1611 <query:ResponseOption returnComposedObjects="true"
1612     returnType="LeafClassWithRepositoryItem"/>
1613
1614 <rim:AdhocQuery id="temporaryId">
1615     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1616 regrep:QueryLanguage:SQL-92">
1617     </rim:QueryExpression>
1618 </rim:AdhocQuery>

```

1619 Example of All SuperProperties Discovery Query

1620 6.2 Immediate SuperClass Discovery Query

1621 The Immediate SuperClass discovery query MUST be implemented by an ebXML Registry implementing
1622 this profile. It allows the discovery of all of the immediate super classes of a given class. The canonical
1623 query corresponding to this discovery query is presented in Section 7.3.2.

1624 6.2.1 Parameter \$className

1625 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1626 value of ClassificationNodes.

1627 6.2.2 Example of Immediate SuperClass Discovery Query

1628 The following example illustrates how to find all the immediate super classes of a given class that have a
1629 name containing the string "AirReservationServices".

```

1630 <rs:RequestSlotList>
1631     <rim:Slot
1632         name="urn:oasis:names:tc:ebxml-
1633 regrep:3.0:rs:AdhocQueryRequest:queryId">
1634         <rim:ValueList>
1635             <rim:Value>urn:oasis:names:tc:ebxml-
1636 regrep:profile:webontology:query:FindImmediateSuperClasses</rim:Value>
1637         </rim:ValueList>
1638     </rim:Slot>
1639     <rim:Slot name="urn:oasis:names:tc:ebxml-
1640 regrep:rs:AdhocQueryRequest:queryId">
1641         <rim:ValueList>
1642             <rim:Value>urn:oasis:names:tc:ebxml-
1643 regrep:profile:webontology:query:FindImmediateSuperClasses</rim:Value>
1644         </rim:ValueList>
1645     </rim:Slot>
1646     <rim:Slot name="$className">
1647         <rim:ValueList>
1648             <rim:Value>%AirReservationServices%</rim:Value>
1649         </rim:ValueList>
1650     </rim:Slot>
1651 </rs:RequestSlotList>
1652
1653 <query:ResponseOption returnComposedObjects="true"
1654     returnType="LeafClassWithRepositoryItem"/>

```

```

1655 <rim:AdhocQuery id="temporaryId">
1656   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1657   regrep:QueryLanguage:SQL-92">
1658     </rim:QueryExpression>
1659   </rim:AdhocQuery>
1660

```

1661 Example of Immediate SuperClass Discovery Query

1662 6.3 Immediate SubClass Discovery Query

1663 The Immediate SubClass discovery query MUST be implemented by an ebXML Registry implementing
1664 this profile. It allows the discovery of all of the immediate subclasses of a given class. The canonical
1665 query corresponding to this discovery query is presented in Section 7.3.3.

1666 6.3.1 Parameter \$className

1667 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1668 value of ClassificationNode.

1669 6.3.2 Example of Immediate SubClass Discovery Query

1670 The following example illustrates how to find all the immediate subclasses of a given class that have a
1671 name containing the string "AirServices" .

```

1672 <rs:RequestSlotList>
1673   <rim:Slot
1674     name="urn:oasis:names:tc:ebxml-
1675     regrep:3.0:rs:AdhocQueryRequest:queryId">
1676     <rim:ValueList>
1677       <rim:Value>urn:oasis:names:tc:ebxml-
1678       regrep:profile:webontology:query:FindImmediateSubClasses</rim:Value>
1679     </rim:ValueList>
1680   </rim:Slot>
1681   <rim:Slot name="urn:oasis:names:tc:ebxml-
1682   regrep:rs:AdhocQueryRequest:queryId">
1683     <rim:ValueList>
1684       <rim:Value>urn:oasis:names:tc:ebxml-
1685       regrep:profile:webontology:query:FindImmediateSubClasses</rim:Value>
1686     </rim:ValueList>
1687   </rim:Slot>
1688   <rim:Slot name="$className">
1689     <rim:ValueList>
1690       <rim:Value>%AirServices%</rim:Value>
1691     </rim:ValueList>
1692   </rim:Slot>
1693 </rs:RequestSlotList>
1694
1695 <query:ResponseOption returnComposedObjects="true"
1696   returnType="LeafClassWithRepositoryItem"/>
1697
1698 <rim:AdhocQuery id="temporaryId">
1699   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1700   regrep:QueryLanguage:SQL-92">
1701     </rim:QueryExpression>
1702   </rim:AdhocQuery>

```

1703 Example of Immediate SubClass Discovery Query

1704 6.4 All SuperClasses Discovery Query

1705 It should be noted that, given a class, finding its immediate subclasses, super classes is necessary but
1706 not sufficient. Given a class, it should be possible to retrieve all of its subclasses, and all of its super
1707 classes. This requires a recursion mechanism in SQL queries.

1708 The All SuperClasses discovery query MAY be implemented by an ebXML Registry implementing this
1709 profile. It allows the discovery of all super classes of a given ClassificationNode recursively in an ebXML
1710 Registry implementation supporting recursion. The canonical query corresponding to this discovery query
1711 is presented in Section 7.3.4.

1712 6.4.1 Parameter \$className

1713 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1714 value of ClassificationNode.

1715 6.4.2 Example of All SuperClasses Discovery Query

1716 The following example illustrates how to find all the super classes of a given class recursively that have a
1717 name containing the string "AirReservationServices" if the query is implemented as an Adhoc Query .

```
1718 <rs:RequestSlotList>  
1719   <rim:Slot  
1720     name="urn:oasis:names:tc:ebxml-  
1721     regrep:3.0:rs:AdhocQueryRequest:queryId">  
1722     <rim:ValueList>  
1723       <rim:Value>urn:oasis:names:tc:ebxml-  
1724       regrep:profile:webontology:query:FindAllSuperClasses</rim:Value>  
1725     </rim:ValueList>  
1726   </rim:Slot>  
1727   <rim:Slot name="urn:oasis:names:tc:ebxml-  
1728   regrep:rs:AdhocQueryRequest:queryId">  
1729     <rim:ValueList>  
1730       <rim:Value>urn:oasis:names:tc:ebxml-  
1731       regrep:profile:webontology:query:FindAllSuperClasses</rim:Value>  
1732     </rim:ValueList>  
1733   </rim:Slot>  
1734   <rim:Slot name="$className">  
1735     <rim:ValueList>  
1736       <rim:Value>%AirReservationServices%</rim:Value>  
1737     </rim:ValueList>  
1738   </rim:Slot>  
1739 </rs:RequestSlotList>  
  
1740 <query:ResponseOption returnComposedObjects="true"  
1741   returnType="LeafClassWithRepositoryItem"/>  
1742  
1743 <rim:AdhocQuery id="temporaryId">  
1744   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
1745   regrep:QueryLanguage:SQL-92">  
1746     </rim:QueryExpression>  
1747 </rim:AdhocQuery>  
1748
```

1749 Example of All SuperClasses Discovery Query

1750 6.5 All SubClasses Discovery Query

1751 The All SubClasses discovery query MAY be implemented by an ebXML Registry implementing this
1752 profile. It allows the discovery of all subclasses of a given ClassificationNode recursively in an ebXML
1753 Registry implementation supporting recursion. The canonical query corresponding to this discovery query
1754 is presented in Section 7.3.5.

1755 6.5.1 Parameter \$className

1756 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1757 value of ClassificationNode.

1758 6.5.2 Example of All SubClasses Discovery Query

1759 The following example illustrates how to find all the subclasses of a given class recursively that have a

1760 name containing the string "AirServices" , if the query is implemented as an Adhoc Query.

```
1761 <rs:RequestSlotList>
1762   <rim:Slot
1763     name="urn:oasis:names:tc:ebxml-
1764   regrep:3.0:rs:AdhocQueryRequest:queryId">
1765     <rim:ValueList>
1766       <rim:Value>urn:oasis:names:tc:ebxml-
1767   regrep:profile:webontology:query:FindAllSubClasses</rim:Value>
1768     </rim:ValueList>
1769   </rim:Slot>
1770   <rim:Slot name="urn:oasis:names:tc:ebxml-
1771   regrep:rs:AdhocQueryRequest:queryId">
1772     <rim:ValueList>
1773       <rim:Value>urn:oasis:names:tc:ebxml-
1774   regrep:profile:webontology:query:FindAllSubClasses</rim:Value>
1775     </rim:ValueList>
1776   </rim:Slot>
1777   <rim:Slot name="$className">
1778     <rim:ValueList>
1779       <rim:Value>%AirServices%</rim:Value>
1780     </rim:ValueList>
1781   </rim:Slot>
1782 </rs:RequestSlotList>
1783
1784 <query:ResponseOption returnComposedObjects="true"
1785   returnType="LeafClassWithRepositoryItem"/>
1786
1787 <rim:AdhocQuery id="temporaryId">
1788   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1789   regrep:QueryLanguage:SQL-92">
1790     </rim:QueryExpression>
1791 </rim:AdhocQuery>
```

1792 Example of All SubClasses Discovery Query

1793 6.6 EquivalentClasses Discovery Query

1794 The EquivalentClasses discovery query MUST be implemented by an ebXML Registry implementing this
1795 profile. It allows the discovery of all the equivalent classes of a given ClassificationNode.

1796 6.6.1 Parameter \$className

1797 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1798 value of ClassificationNodes.

1799 6.6.2 Example of EquivalentClasses Discovery Query

1800 The following example illustrates how to find all the equivalent classes of a given class that have a name
1801 containing the string "AirServices" .

```
1802 <rs:RequestSlotList>
1803   <rim:Slot
1804     name="urn:oasis:names:tc:ebxml-
1805   regrep:3.0:rs:AdhocQueryRequest:queryId">
1806     <rim:ValueList>
1807       <rim:Value>urn:oasis:names:tc:ebxml-
1808   regrep:profile:webontology:query:FindEquivalentClasses</rim:Value>
1809     </rim:ValueList>
1810   </rim:Slot>
1811   <rim:Slot name="urn:oasis:names:tc:ebxml-
1812   regrep:rs:AdhocQueryRequest:queryId">
1813     <rim:ValueList>
1814       <rim:Value>urn:oasis:names:tc:ebxml-
1815   regrep:profile:webontology:query:FindEquivalentClasses</rim:Value>
1816     </rim:ValueList>
```

```

1817     </rim:Slot>
1818     <rim:Slot name="$className">
1819         <rim:ValueList>
1820             <rim:Value>%AirServices%</rim:Value>
1821         </rim:ValueList>
1822     </rim:Slot>
1823 </rs:RequestSlotList>
1824
1825 <query:ResponseOption returnComposedObjects="true"
1826     returnType="LeafClassWithRepositoryItem"/>
1827
1828 <rim:AdhocQuery id="temporaryId">
1829     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1830 regrep:QueryLanguage:SQL-92">
1831     </rim:QueryExpression>
1832 </rim:AdhocQuery>

```

1833 Example of Equivalent Classes Discovery Query

1834 6.7 EquivalentProperties Discovery Query

1835 The EquivalentProperties discovery query MUST be implemented by an ebXML Registry implementing
1836 this profile. It allows the discovery of all the equivalent properties of a given Association that have
1837 associationType of Property. The canonical query corresponding to this discovery query is presented in
1838 Section 7.3.7.

1839 6.7.1 Parameter \$propertyName

1840 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1841 value of Associations that have associationType of Property

1842 6.7.2 Example of EquivalentProperties Discovery Query

1843 The following example illustrates how to find all the equivalent properties(Association Type) of a given
1844 property (Association Type) that have a name containing the string "paymentMethods" .

```

1845 <rs:RequestSlotList>
1846     <rim:Slot
1847         name="urn:oasis:names:tc:ebxml-
1848 regrep:3.0:rs:AdhocQueryRequest:queryId">
1849         <rim:ValueList>
1850             <rim:Value>urn:oasis:names:tc:ebxml-
1851 regrep:profile:webontology:query:FindEquivalentProperties</rim:Value>
1852         </rim:ValueList>
1853     </rim:Slot>
1854     <rim:Slot name="urn:oasis:names:tc:ebxml-
1855 regrep:rs:AdhocQueryRequest:queryId">
1856         <rim:ValueList>
1857             <rim:Value>urn:oasis:names:tc:ebxml-
1858 regrep:profile:webontology:query:FindEquivalentProperties</rim:Value>
1859         </rim:ValueList>
1860     </rim:Slot>
1861     <rim:Slot name="$propertyName">
1862         <rim:ValueList>
1863             <rim:Value>%paymentMethods%</rim:Value>
1864         </rim:ValueList>
1865     </rim:Slot>
1866 </rs:RequestSlotList>
1867
1868 <query:ResponseOption returnComposedObjects="true"
1869     returnType="LeafClassWithRepositoryItem"/>
1870
1871 <rim:AdhocQuery id="temporaryId">
1872     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1873 regrep:QueryLanguage:SQL-92">

```

```
1874     </rim:QueryExpression>
1875 </rim:AdhocQuery>
```

1876 Example of Equivalent Properties Discovery Query

1877 6.8 SameExtrinsicObjects Discovery Query

1878 The SameExtrinsicObjects discovery query MUST be implemented by an ebXML Registry implementing
1879 this profile. It allows the discovery of all the "ExtrinsicObjects" defined to be the same with a given
1880 ExtrinsicObject. The canonical query corresponding to this discovery query is presented in Section 7.3.8.

1881 6.8.1 Parameter \$extrinsicObjectName

1882 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1883 value of ExtrinsicObjects.

1884 6.8.2 Example of SameExtrinsicObjects Discovery Query

1885 The following example illustrates how to find all the ExtrinsicObjects that are defined to be the same as
1886 the ExtrinsicObject that have a name containing the string "MyDocument" .

```
1887 <rs:RequestSlotList>
1888   <rim:Slot
1889     name="urn:oasis:names:tc:ebxml-
1890     regrep:3.0:rs:AdhocQueryRequest:queryId">
1891     <rim:ValueList>
1892       <rim:Value>urn:oasis:names:tc:ebxml-
1893       regrep:profile:webontology:query:FindTheSameExtrinsicObjects</rim:Value>
1894     </rim:ValueList>
1895   </rim:Slot>
1896   <rim:Slot name="urn:oasis:names:tc:ebxml-
1897   regrep:rs:AdhocQueryRequest:queryId">
1898     <rim:ValueList>
1899       <rim:Value>urn:oasis:names:tc:ebxml-
1900       regrep:profile:webontology:query:FindTheSameExtrinsicObjects</rim:Value>
1901     </rim:ValueList>
1902   </rim:Slot>
1903   <rim:Slot name="$extrinsicObjectName">
1904     <rim:ValueList>
1905       <rim:Value>%MyDocument%</rim:Value>
1906     </rim:ValueList>
1907   </rim:Slot>
1908 </rs:RequestSlotList>
1909
1910 <query:ResponseOption returnComposedObjects="true"
1911   returnType="LeafClassWithRepositoryItem"/>
1912
1913 <rim:AdhocQuery id="temporaryId">
1914   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1915   regrep:QueryLanguage:SQL-92">
1916     </rim:QueryExpression>
1917 </rim:AdhocQuery>
```

1919 Example of SameExtrinsicObjects Discovery Query

1920 6.9 DifferentExtrinsicObjects Discovery Query

1921 The DifferentExtrinsicObjects discovery query MUST be implemented by an ebXML Registry
1922 implementing this profile. It allows the discovery of all the "ExtrinsicObjects" defined to be the different
1923 from a given ExtrinsicObject. The canonical query corresponding to this discovery query is presented in
1924 Section 7.3.9.

1925 6.9.1 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 6.9.2 Example of DifferentExtrinsicObjects Discovery Query

1929 The following example illustrates how to find all the ExtrinsicObjects that are defined to be different from
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:FindDifferentExtrinsicObjects</rim:Value
1938   >
1939     </rim:ValueList>
1940   </rim:Slot>
1941   <rim:Slot name="urn:oasis:names:tc:ebxml-
1942   regrep:rs:AdhocQueryRequest:queryId">
1943     <rim:ValueList>
1944       <rim:Value>urn:oasis:names:tc:ebxml-
1945   regrep:profile:webontology:query:FindDifferentExtrinsicObjects</rim:Value
1946   >
1947     </rim:ValueList>
1948   </rim:Slot>
1949   <rim:Slot name="$extrinsicObjectName">
1950     <rim:ValueList>
1951       <rim:Value>%MyDocument%</rim:Value>
1952     </rim:ValueList>
1953   </rim:Slot>
1954 </rs:RequestSlotList>
1955
1956 <query:ResponseOption returnComposedObjects="true"
1957   returnType="LeafClassWithRepositoryItem"/>
1958
1959 <rim:AdhocQuery id="temporaryId">
1960   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1961   regrep:QueryLanguage:SQL-92">
1962     </rim:QueryExpression>
1963 </rim:AdhocQuery>
1964
```

1965 Example of DifferentExtrinsicObjects Discovery Query

1966 6.10 AllDifferentRegistryObject Discovery Query

1967 The AllDifferentRegistryObjects discovery query MUST be implemented by an ebXML Registry
1968 implementing this profile. Given a RegistryObject, it allows the discovery of all the other member
1969 "RegistryObjects" of a Registry package that are defined to be the different from each other through a
1970 allDifferent slot. The canonical query corresponding to this discovery query is presented in Section
1971 7.3.10.

1972 6.10.1 Parameter \$registryObjectName

1973 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1974 value of RegistryObjects.

1975 6.10.2 Example of AllDifferentRegistryObjects Discovery Query

1976 The following example illustrates how to find all the RegistryObjects that are defined to be different from
1977 the RegistryObject that have a name containing the string "MyDocument" .

```

1978
1979 <rs:RequestSlotList>
1980   <rim:Slot
1981     name="urn:oasis:names:tc:ebxml-
1982 regrep:3.0:rs:AdhocQueryRequest:queryId">
1983     <rim:ValueList>
1984       <rim:Value>urn:oasis:names:tc:ebxml-
1985 regrep:profile:webontology:query:FindAllDifferent</rim:Value>
1986     </rim:ValueList>
1987   </rim:Slot>
1988   <rim:Slot name="urn:oasis:names:tc:ebxml-
1989 regrep:rs:AdhocQueryRequest:queryId">
1990     <rim:ValueList>
1991       <rim:Value>urn:oasis:names:tc:ebxml-
1992 regrep:profile:webontology:query:FindAllDifferent</rim:Value>
1993     </rim:ValueList>
1994   </rim:Slot>
1995   <rim:Slot name="$registryObjectName">
1996     <rim:ValueList>
1997       <rim:Value>%MyDocument%</rim:Value>
1998     </rim:ValueList>
1999   </rim:Slot>
2000 </rs:RequestSlotList>
2001
2002 <query:ResponseOption returnComposedObjects="true"
2003   returnType="LeafClassWithRepositoryItem"/>
2004
2005 <rim:AdhocQuery id="temporaryId">
2006   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2007 regrep:QueryLanguage:SQL-92">
2008   </rim:QueryExpression>
2009 </rim:AdhocQuery>

```

2010 Example of AllDifferentRegistryObjects Discovery Query

2011 6.11 ObjectProperties Discovery Query

2012 The ObjectProperties discovery query MUST be implemented by an ebXML Registry implementing this
2013 profile. It allows the discovery of all of the objectProperties of a given classification node. The canonical
2014 query corresponding to this discovery query is presented in Section 7.3.11.

2015 6.11.1 Parameter \$className

2016 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2017 value of ClassificationNodes.

2018 6.11.2 Example of ObjectProperties Discovery Query

2019 The following example illustrates how to find all the object properties of a given classification node having
2020 a name containing "AirServices" .

```

2021
2022 <rs:RequestSlotList>
2023   <rim:Slot
2024     name="urn:oasis:names:tc:ebxml-
2025 regrep:3.0:rs:AdhocQueryRequest:queryId">
2026     <rim:ValueList>
2027       <rim:Value>urn:oasis:names:tc:ebxml-
2028 regrep:profile:webontology:query:FindObjectProperties</rim:Value>
2029     </rim:ValueList>
2030   </rim:Slot>
2031   <rim:Slot name="urn:oasis:names:tc:ebxml-
2032 regrep:rs:AdhocQueryRequest:queryId">
2033     <rim:ValueList>

```



```

2034         <rim:Value>urn:oasis:names:tc:ebxml-
2035 regrep:profile:webontology:query:FindObjectProperties</rim:Value>
2036     </rim:ValueList>
2037 </rim:Slot>
2038 <rim:Slot name="$className">
2039     <rim:ValueList>
2040         <rim:Value>%AirServices%</rim:Value>
2041     </rim:ValueList>
2042 </rim:Slot>
2043 </rs:RequestSlotList>
2044
2045 <query:ResponseOption returnComposedObjects="true"
2046     returnType="LeafClassWithRepositoryItem"/>
2047
2048 <rim:AdhocQuery id="temporaryId">
2049     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2050 regrep:QueryLanguage:SQL-92">
2051     </rim:QueryExpression>
2052 </rim:AdhocQuery>

```

2053 Example of ObjectProperties Discovery Query

2054 6.12 ImmediateInheritedObjectProperties Discovery Query

2055 The ImmediateInheritedObjectProperties discovery query MUST be implemented by an ebXML Registry
2056 implementing this profile. It allows the discovery of all of the objectProperties of a given classification node
2057 including the ones inherited from its immediate super classes. The canonical query corresponding to this
2058 discovery query is presented in Section 7.3.12.

2059 6.12.1 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 6.12.2 Example of ImmediateInheritedObjectProperties 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" including the ones inherited from its immediate super classes.

```

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:FindImmediateInheritedObjectProperties</
2073 rim:Value>
2074         </rim:ValueList>
2075     </rim:Slot>
2076     <rim:Slot name="urn:oasis:names:tc:ebxml-
2077 regrep:rs:AdhocQueryRequest:queryId">
2078         <rim:ValueList>
2079             <rim:Value>urn:oasis:names:tc:ebxml-
2080 regrep:profile:webontology:query:FindImmediateInheritedObjectProperties</
2081 rim:Value>
2082         </rim:ValueList>
2083     </rim:Slot>
2084     <rim:Slot name="$className">
2085         <rim:ValueList>
2086             <rim:Value>%AirServices%</rim:Value>
2087         </rim:ValueList>
2088     </rim:Slot>
2089 </rs:RequestSlotList>
2090

```

```

2091 <query:ResponseOption returnComposedObjects="true"
2092     returnType="LeafClassWithRepositoryItem"/>
2093
2094 <rim:AdhocQuery id="temporaryId">
2095     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2096     regrep:QueryLanguage:SQL-92">
2097         </rim:QueryExpression>
2098     </rim:AdhocQuery>

```

2099 Example of ImmediateInheritedObjectProperties Discovery Query

2100 6.13 AllInheritedObjectProperties Discovery Query

2101 It should be noted that, given a class, finding the object properties inherited from immediate super classes
2102 is necessary but not sufficient. Given a class, it should be possible to retrieve all of the object properties
2103 inherited from its super classes. This requires a recursion mechanism in SQL queries.

2104 The AllInheritedObjectProperties discovery query MAY be implemented by an ebXML Registry
2105 implementing this profile. It allows the discovery of all inherited ObjectProperties recursively of a given
2106 ClassificationNode in a ClassificationScheme in an ebXML Registry implementation supporting recursion.

2107 The canonical query corresponding to this discovery query is presented in Section 7.3.13.

2108 6.13.1 Parameter \$className

2109 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2110 value of ClassificationNodes.

2111 6.13.2 Example of AllInheritedObjectProperties Discovery Query

2112 The following example illustrates how to find all the object properties of a given classification node having
2113 a name containing "AirReservationServices" including the ones inherited from all of its super classes
2114 recursively, if the query is implemented as an Adhoc Query.

```

2115
2116 <rs:RequestSlotList>
2117     <rim:Slot
2118         name="urn:oasis:names:tc:ebxml-
2119     regrep:3.0:rs:AdhocQueryRequest:queryId">
2120         <rim:ValueList>
2121             <rim:Value>urn:oasis:names:tc:ebxml-
2122     regrep:profile:webontology:query:FindAllInheritedObjectProperties</rim:Va
2123     lue>
2124         </rim:ValueList>
2125     </rim:Slot>
2126     <rim:Slot name="urn:oasis:names:tc:ebxml-
2127     regrep:rs:AdhocQueryRequest:queryId">
2128         <rim:ValueList>
2129             <rim:Value>urn:oasis:names:tc:ebxml-
2130     regrep:profile:webontology:query:FindAll
2131     InheritedObjectProperties</rim:Value>
2132         </rim:ValueList>
2133     </rim:Slot>
2134     <rim:Slot name="$className">
2135         <rim:ValueList>
2136             <rim:Value>%AirReservationServices%</rim:Value>
2137         </rim:ValueList>
2138     </rim:Slot>
2139 </rs:RequestSlotList>
2140
2141 <query:ResponseOption returnComposedObjects="true"
2142     returnType="LeafClassWithRepositoryItem"/>
2143
2144 <rim:AdhocQuery id="temporaryId">
2145     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2146     regrep:QueryLanguage:SQL-92">

```

2147 </rim:QueryExpression>
2148 </rim:AdhocQuery>

2149 Example of AllInheritedObjectProperties Discovery Query

2150 6.14 DatatypeProperties Discovery Query

2151 The DatatypeProperties discovery query MUST be implemented by an ebXML Registry implementing this
2152 profile. It allows the discovery of all of the datatypeProperties of a given classification node. The
2153 canonical query corresponding to this discovery query is presented in Section 7.3.14.

2154 6.14.1 Parameter \$className

2155 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2156 value of ClassificationNodes.

2157 6.14.2 Example of DatatypeProperties Discovery Query

2158 The following example illustrates how to find all the datatype properties of a given classification node
2159 having a name containing "AirReservationServices" .

```
2160  
2161 <rs:RequestSlotList>  
2162   <rim:Slot  
2163     name="urn:oasis:names:tc:ebxml-  
2164     regrep:3.0:rs:AdhocQueryRequest:queryId">  
2165     <rim:ValueList>  
2166       <rim:Value>urn:oasis:names:tc:ebxml-  
2167       regrep:profile:webontology:query:FindDatatypeProperties</rim:Value>  
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:FindDatatypeProperties</rim:Value>  
2175     </rim:ValueList>  
2176   </rim:Slot>  
2177   <rim:Slot name="$className">  
2178     <rim:ValueList>  
2179       <rim:Value>%AirReservationServices%</rim:Value>  
2180     </rim:ValueList>  
2181   </rim:Slot>  
2182 </rs:RequestSlotList>  
2183  
2184 <query:ResponseOption returnComposedObjects="true"  
2185   returnType="LeafClassWithRepositoryItem"/>  
2186  
2187 <rim:AdhocQuery id="temporaryId">  
2188   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
2189   regrep:QueryLanguage:SQL-92">  
2190     </rim:QueryExpression>  
2191 </rim:AdhocQuery>
```

2192 Example of DatatypeProperties Discovery Query

2193 6.15 AllInheritedDatatypeProperties Discovery Query

2194 It should be noted that, given a class, finding the datatype properties inherited from immediate super
2195 classes is necessary but not sufficient. Given a class, it should be possible to retrieve all of the datatype
2196 properties inherited from its super classes. This requires a recursion mechanism in SQL queries.

2197 The AllInheritedDatatypeProperties discovery query MAY be implemented by an ebXML Registry
2198 implementing this profile. It allows the discovery of all inherited DatatypeProperties recursively of a given
2199 ClassificationNode in a ClassificationScheme in an ebXML Registry implementation supporting recursion.

2200 The canonical query corresponding to this discovery query is presented in Section 7.3.15.

2201 6.15.1 Parameter \$className

2202 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2203 value of ClassificationNodes.

2204 6.15.2 Example of AllInheritedDatatypeProperties Discovery Query

2205 The following example illustrates how to find all the datatype properties of a given classification node
2206 having a name containing "AirReservationServices" including the ones inherited from all of its super
2207 classes recursively, if the query is implemented as an Adhoc Query.

2208

```
2209 <rs:RequestSlotList>
2210   <rim:Slot
2211     name="urn:oasis:names:tc:ebxml-
2212   regrep:3.0:rs:AdhocQueryRequest:queryId">
2213     <rim:ValueList>
2214       <rim:Value>urn:oasis:names:tc:ebxml-
2215   regrep:profile:webontology:query:FindAllInheritedDatatypeProperties</rim:
2216   Value>
2217     </rim:ValueList>
2218   </rim:Slot>
2219   <rim:Slot name="urn:oasis:names:tc:ebxml-
2220   regrep:rs:AdhocQueryRequest:queryId">
2221     <rim:ValueList>
2222       <rim:Value>urn:oasis:names:tc:ebxml-
2223   regrep:profile:webontology:query:FindAllInheritedDatatypeProperties</rim:
2224   Value>
2225     </rim:ValueList>
2226   </rim:Slot>
2227   <rim:Slot name="$className">
2228     <rim:ValueList>
2229       <rim:Value>%AirReservationServices %</rim:Value>
2230     </rim:ValueList>
2231   </rim:Slot>
2232 </rs:RequestSlotList>
2233
2234 <query:ResponseOption returnComposedObjects="true"
2235   returnType="LeafClassWithRepositoryItem"/>
2236
2237 <rim:AdhocQuery id="temporaryId">
2238   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2239   regrep:QueryLanguage:SQL-92">
2240     </rim:QueryExpression>
2241 </rim:AdhocQuery>
```

2242 Example of AllInheritedDatatypeProperties Discovery Query

2243 6.16 TransitiveRelationships Discovery Query

2244 To make any use of the transitive property in ebXML registries, coding is necessary to find out the implied
2245 information. The TransitiveRelationships discovery query MUST be implemented by an ebXML Registry
2246 implementing this profile to handle this semantics.

2247 Given a class which is a source of a transitive property, this discovery query retrieves not only the target
2248 objects of a given transitive property, but if the target objects have the same property, it retrieves their
2249 target objects too. The canonical query corresponding to this discovery query is presented in Section
2250 7.3.16.

2251 6.16.1 Parameter \$className

2252 This parameter's value SHALL specify a string containing a pattern to match against the name attribute

2253 value of ClassificationNodes.

2254 6.16.2 Parameter \$propertyName

2255 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2256 value of Associations that have associationType of Property

2257 6.16.3 Example of TransitiveRelationships Discovery Query

2258 The following example illustrates how to retrieve all the target objects of the "succeeds" property of the
2259 "AirReservationServices" including the target objects implied by a transitive property relationship.

2260

```
2261 <rs:RequestSlotList>
2262   <rim:Slot
2263     name="urn:oasis:names:tc:ebxml-
2264   regrep:3.0:rs:AdhocQueryRequest:queryId">
2265     <rim:ValueList>
2266       <rim:Value>urn:oasis:names:tc:ebxml-
2267   regrep:profile:webontology:query:FindTransitiveRelationships</rim:Value>
2268     </rim:ValueList>
2269   </rim:Slot>
2270   <rim:Slot name="urn:oasis:names:tc:ebxml-
2271   regrep:rs:AdhocQueryRequest:queryId">
2272     <rim:ValueList>
2273       <rim:Value>urn:oasis:names:tc:ebxml-
2274   regrep:profile:webontology:query:FindTransitiveRelationships</rim:Value>
2275     </rim:ValueList>
2276   </rim:Slot>
2277   <rim:Slot name="$className">
2278     <rim:ValueList>
2279       <rim:Value>%AirReservationServices%</rim:Value>
2280     </rim:ValueList>
2281   </rim:Slot>
2282   <rim:Slot name="$propertyName">
2283     <rim:ValueList>
2284       <rim:Value>%succeeds%</rim:Value>
2285     </rim:ValueList>
2286   </rim:Slot>
2287 </rs:RequestSlotList>
2288
2289 <query:ResponseOption returnComposedObjects="true"
2290   returnType="LeafClassWithRepositoryItem"/>
2291
2292 <rim:AdhocQuery id="temporaryId">
2293   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2294   regrep:QueryLanguage:SQL-92">
2295     </rim:QueryExpression>
2296 </rim:AdhocQuery>
```

2297 Example of TransitiveRelationships Discovery Query

2298 6.17 TargetObjects Discovery Query

2299 The TargetObjects discovery query MUST be implemented by an ebXML Registry implementing this
2300 profile. It allows the discovery of the targetObjects from the Registry, given a Classification Node
2301 (sourceObject) and a property name (Association Type). The canonical query corresponding to this
2302 discovery query is presented in Section 7.3.17.

2303 6.17.1 Parameter \$className

2304 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2305 value of ClassificationNodes.

2306 6.17.2 Parameter \$propertyName

2307 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2308 value of Associations that have associationType of Property.

2309 6.17.3 Example of TargetObjects Discovery Query

2310 The following example illustrates how to retrieve all the target objects of the "paymentMethod" property of
2311 the "AirReservationServices".

2312

```
2313 <rs:RequestSlotList>
2314   <rim:Slot
2315     name="urn:oasis:names:tc:ebxml-
2316   regrep:3.0:rs:AdhocQueryRequest:queryId">
2317     <rim:ValueList>
2318       <rim:Value>urn:oasis:names:tc:ebxml-
2319   regrep:profile:webontology:query:FindTargetObjects</rim:Value>
2320     </rim:ValueList>
2321   </rim:Slot>
2322   <rim:Slot name="urn:oasis:names:tc:ebxml-
2323   regrep:rs:AdhocQueryRequest:queryId">
2324     <rim:ValueList>
2325       <rim:Value>urn:oasis:names:tc:ebxml-
2326   regrep:profile:webontology:query:FindTargetObjects</rim:Value>
2327     </rim:ValueList>
2328   </rim:Slot>
2329   <rim:Slot name="$className">
2330     <rim:ValueList>
2331       <rim:Value>%AirReservationServices%</rim:Value>
2332     </rim:ValueList>
2333   </rim:Slot>
2334   <rim:Slot name="$propertyName">
2335     <rim:ValueList>
2336       <rim:Value>%paymentMethod%</rim:Value>
2337     </rim:ValueList>
2338   </rim:Slot>
2339 </rs:RequestSlotList>
2340
2341 <query:ResponseOption returnComposedObjects="true"
2342   returnType="LeafClassWithRepositoryItem"/>
2343
2344 <rim:AdhocQuery id="temporaryId">
2345   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2346   regrep:QueryLanguage:SQL-92">
2347     </rim:QueryExpression>
2348 </rim:AdhocQuery>
```

2349

Example of TargetObjects Discovery Query

2350

2351 6.18 TargetObjectsInverseOf Discovery Query

2352 The TargetObjectsInverseOf discovery query MUST be implemented by an ebXML Registry implementing
2353 this profile. Given a Classification Node (sourceObject) and a property name (Association Type), this
2354 query retrieves the source objects of the properties which are stated to be inverseOf the property name
2355 given as a parameter, and considering the Classification Node name as the targetObject of these
2356 properties. The canonical query corresponding to this discovery query is presented in Section 7.3.18.

2357 6.18.1 Parameter \$className

2358 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2359 value of ClassificationNodes.

2360 6.18.2 Parameter \$propertyName

2361 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2362 value of Associations that have associationType of Property.

2363 6.18.3 Example of TargetObjectsInverseOf Discovery Query

2364 The following example illustrates how to retrieve all the source objects of the properties which are stated
2365 to the the inverseOf the property "succeeds", considering the "AirReservationServices" as the target object
2366 of these properties.

2367

```
2368 <rs:RequestSlotList>
2369   <rim:Slot
2370     name="urn:oasis:names:tc:ebxml-
2371   regrep:3.0:rs:AdhocQueryRequest:queryId">
2372     <rim:ValueList>
2373       <rim:Value>urn:oasis:names:tc:ebxml-
2374   regrep:profile:webontology:query:FindTOinverseOf</rim:Value>
2375     </rim:ValueList>
2376   </rim:Slot>
2377   <rim:Slot name="urn:oasis:names:tc:ebxml-
2378   regrep:rs:AdhocQueryRequest:queryId">
2379     <rim:ValueList>
2380       <rim:Value>urn:oasis:names:tc:ebxml-
2381   regrep:profile:webontology:query:FindTOinverseOf</rim:Value>
2382     </rim:ValueList>
2383   </rim:Slot>
2384   <rim:Slot name="$className">
2385     <rim:ValueList>
2386       <rim:Value>%AirReservationServices%</rim:Value>
2387     </rim:ValueList>
2388   </rim:Slot>
2389   <rim:Slot name="$propertyName">
2390     <rim:ValueList>
2391       <rim:Value>%succeeds%</rim:Value>
2392     </rim:ValueList>
2393   </rim:Slot>
2394 </rs:RequestSlotList>
2395
2396 <query:ResponseOption returnComposedObjects="true"
2397   returnType="LeafClassWithRepositoryItem"/>
2398
2399 <rim:AdhocQuery id="temporaryId">
2400   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2401   regrep:QueryLanguage:SQL-92">
2402     </rim:QueryExpression>
2403 </rim:AdhocQuery>
```

2404 Example of TargetObjectsInverseOf Discovery Query

2405

2406 6.19 InverseRanges Discovery Query

2407 The InverseRanges discovery query MUST be implemented by an ebXML Registry implementing this
2408 profile to handle this semantics. Given a Classification Node (sourceObject) and a property name
2409 (Association Type), this query retrieves not only the target objects of this property, but also the source
2410 objects of the properties which are stated to be inverseOf the property name given as a parameter, and
2411 considering the Classification Node name as the targetObject of these properties. This query can be
2412 thought as the union of the queries presented in Sections 6.17 and 6.18. The canonical query
2413 corresponding to this discovery query is presented in Section 7.3.19.

2414 6.19.1 Parameter \$className

2415 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2416 value of ClassificationNodes.

2417 6.19.2 Parameter \$propertyName

2418 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2419 value of Associations that have associationType of Property

2420 6.19.3 Example of InverseRanges Discovery Query

2421 Consider, for example, the "succeeds" property defined in Section 4.3.3. To denote that a certain Web
2422 service instance precedes another during execution, we may define the "precedes" property as an inverse
2423 of the "succeeds" property as follows:

2424

```
2425 <owl:ObjectProperty rdf:ID="precedes">  
2426   <owl:inverseOf rdf:resource="#succeeds" />  
2427 </owl:ObjectProperty>
```

2428 Example owl:inverseOf Property

2429 Assume that we want to find all the Web services which can succeed a given Web service. In such a
2430 case, we need not only find all the Web services which succeeds this given Web service, that is the target
2431 objects of "succeeds" Association instance, but we also need to find all the sourceObjects of the
2432 "precedes" Association instance since "precedes" is declared to be the "inverseOf" succeeds Association
2433 instance.

2434 The following example illustrates how to retrieve all the services that "succeeds" "AirReservationServices"
2435 by also making use of its "precedes" property.

2436

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



```

2469     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2470     regrep:QueryLanguage:SQL-92">
2471     </rim:QueryExpression>
2472 </rim:AdhocQuery>

```

2473 Example of InverseRanges Discovery Query

2474 6.20 SymmetricProperties Discovery Query

2475 The SymmetricProperties discovery query MUST be implemented by an ebXML Registry implementing
2476 this profile. It allows the discovery of all of the Symmetric Properties of a given classification node. The
2477 canonical query corresponding to this discovery query is presented in Section 7.3.20.

2478 6.20.1 Parameter \$className

2479 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2480 value of ClassificationNodes.

2481 6.20.2 Example of SymmetricProperties Discovery Query

2482 The following example illustrates how to find all the symmetric properties of a given classification node
2483 having a name containing "AirReservationServices" .

```

2484
2485 <rs:RequestSlotList>
2486   <rim:Slot
2487     name="urn:oasis:names:tc:ebxml-
2488     regrep:3.0:rs:AdhocQueryRequest:queryId">
2489     <rim:ValueList>
2490       <rim:Value>urn:oasis:names:tc:ebxml-
2491       regrep:profile:webontology:query:FindSymmetricProperties</rim:Value>
2492     </rim:ValueList>
2493   </rim:Slot>
2494   <rim:Slot name="urn:oasis:names:tc:ebxml-
2495     regrep:rs:AdhocQueryRequest:queryId">
2496     <rim:ValueList>
2497       <rim:Value>urn:oasis:names:tc:ebxml-
2498       regrep:profile:webontology:query:FindSymmetricProperties</rim:Value>
2499     </rim:ValueList>
2500   </rim:Slot>
2501   <rim:Slot name="$className">
2502     <rim:ValueList>
2503       <rim:Value>%AirReservationServices%</rim:Value>
2504     </rim:ValueList>
2505   </rim:Slot>
2506 </rs:RequestSlotList>
2507
2508 <query:ResponseOption returnComposedObjects="true"
2509   returnType="LeafClassWithRepositoryItem"/>
2510
2511 <rim:AdhocQuery id="temporaryId">
2512   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2513   regrep:QueryLanguage:SQL-92">
2514   </rim:QueryExpression>
2515 </rim:AdhocQuery>

```

2516 Example of SymmetricProperties Discovery Query

2517 6.21 FunctionalProperties Discovery Query

2518 The FunctionalProperties discovery query MUST be implemented by an ebXML Registry implementing
2519 this profile. It allows the discovery of all of the Functional Properties of a given classification node. The
2520 canonical query corresponding to this discovery query is presented in Section 7.3.21.

2521 6.21.1 Parameter \$className

2522 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2523 value of ClassificationNodes.

2524 6.21.2 Example of FunctionalProperties Discovery Query

2525 The following example illustrates how to find all the functional properties of a given classification node
2526 having a name containing "AirReservationServices" .

2527

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

2559

Example of Functional Properties Discovery Query

2560 6.22 InverseFunctionalProperties Discovery Query

2561 The InverseFunctionalProperties discovery query MUST be implemented by an ebXML Registry
2562 implementing this profile. It allows the discovery of all of the Inverse Functional Properties of a given
2563 classification node. The canonical query corresponding to this discovery query is presented in Section
2564 7.3.22.

2565 6.22.1 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 6.22.2 Example of InverseFunctionalProperties Discovery Query

2569 The following example illustrates how to find all the inverse functional properties of a given classification
2570 node 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:FindInverseFunctionalProperties</rim:Val
2579 ue>
2580         </rim:ValueList>
2581     </rim:Slot>
2582     <rim:Slot name="urn:oasis:names:tc:ebxml-
2583 regrep:rs:AdhocQueryRequest:queryId">
2584         <rim:ValueList>
2585             <rim:Value>urn:oasis:names:tc:ebxml-
2586 regrep:profile:webontology:query:FindInverseFunctionalProperties</rim:Val
2587 ue>
2588         </rim:ValueList>
2589     </rim:Slot>
2590     <rim:Slot name="$className">
2591         <rim:ValueList>
2592             <rim:Value>%AirReservationServices%</rim:Value>
2593         </rim:ValueList>
2594     </rim:Slot>
2595 </rs:RequestSlotList>
2596
2597 <query:ResponseOption returnComposedObjects="true"
2598     returnType="LeafClassWithRepositoryItem"/>
2599
2600 <rim:AdhocQuery id="temporaryId">
2601     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2602 regrep:QueryLanguage:SQL-92">
2603     </rim:QueryExpression>
2604 </rim:AdhocQuery>

```

2605 Example of InverseFunctional Properties Discovery Query

2606 6.23 Instances Discovery Query

2607 When an intersection definition is used to create a complex class (a new ClassificationNode) in RIM as
2608 described in Section 4.6, it becomes possible to infer that the objects (instances) classified by all of the
2609 classes (ClassificationNodes) specified in the Intersection definition, are also classified by this complex
2610 class.

2611 The Instances discovery query MUST be implemented by an ebXML Registry implementing this profile. It
2612 allows the discovery of all of the direct instances of a given classification node and if it is a complex class
2613 which is an intersection two classes, it also allows to retrieve the intersection of the instances of both of
2614 the classes involved in the intersection definition. The canonical query corresponding to this discovery
2615 query is presented in Section 7.3.23.

2616 6.23.1 Parameter \$className

2617 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2618 value of ClassificationNodes.

2619 6.23.2 Example of Instances Discovery Query

2620 Consider the "AirReservationServices" definition presented in Section 4.6. The following example
2621 illustrates how to find all the direct instances of the "AirReservationServices" and also the instances
2622 classified by both "AirServices" and also the "ReservationServices".

```

2623
2624 <rs:RequestSlotList>
2625     <rim:Slot
2626         name="urn:oasis:names:tc:ebxml-
2627 regrep:3.0:rs:AdhocQueryRequest:queryId">
2628         <rim:ValueList>

```

```

2629         <rim:Value>urn:oasis:names:tc:ebxml-
2630 regrep:profile:webontology:query:FindInstances</rim:Value>
2631     </rim:ValueList>
2632 </rim:Slot>
2633     <rim:Slot name="urn:oasis:names:tc:ebxml-
2634 regrep:rs:AdhocQueryRequest:queryId">
2635         <rim:ValueList>
2636             <rim:Value>urn:oasis:names:tc:ebxml-
2637 regrep:profile:webontology:query:FindInstances</rim:Value>
2638         </rim:ValueList>
2639     </rim:Slot>
2640     <rim:Slot name="$className">
2641         <rim:ValueList>
2642             <rim:Value>%AirReservationServices%</rim:Value>
2643         </rim:ValueList>
2644     </rim:Slot>
2645 </rs:RequestSlotList>
2646
2647 <query:ResponseOption returnComposedObjects="true"
2648     returnType="LeafClassWithRepositoryItem"/>
2649
2650 <rim:AdhocQuery id="temporaryId">
2651     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2652 regrep:QueryLanguage:SQL-92">
2653         </rim:QueryExpression>
2654 </rim:AdhocQuery>

```

Example of Instances Discovery Query

2655

2656 7 Canonical Metadata Definitions

2657 This chapter specifies the canonical metadata defined by this profile.

2658 7.1 ObjectType Extensions

2659 The following new extensions to the canonical ObjectType ClassificationScheme are described by this
2660 profile:

```
2661  
2662 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2663 regrep:ObjectType:RegistryObject:ExtrinsicObject"  
2664 lid="urn:oasis:names:tc:ebxml-  
2665 regrep:profile:webontology:ObjectType:RegistryObject:ExtrinsicObject:OWL"  
2666 code="OWL" id="urn:oasis:names:tc:ebxml-  
2667 regrep:profile:webontology:ObjectType:RegistryObject:ExtrinsicObject:OWL"  
2668 >  
2669 <rim:Name>  
2670 <rim:LocalizedString charset="UTF-8" value="OWL"/>  
2671 </rim:Name>  
2672 </rim:ClassificationNode>
```

2673 7.2 AssociationType Extensions

2674 The following new extensions to the AssociationType ClassificationScheme are described by this profile:

```
2675  
2676 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2677 regrep:classificationScheme:AssociationType"  
2678 lid="urn:oasis:names:tc:ebxml-  
2679 regrep:profile:webontology:AssociationType:OWL" code="OWL"  
2680 id="urn:oasis:names:tc:ebxml-  
2681 regrep:profile:webontology:AssociationType:OWL">  
2682 <rim:Name>  
2683 <rim:LocalizedString charset="UTF-8" value="OWL"/>  
2684 </rim:Name>  
2685 </rim:ClassificationNode>  
2686 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2687 regrep:profile:webontology:AssociationType:OWL"  
2688 lid="urn:oasis:names:tc:ebxml-  
2689 regrep:profile:webontology:AssociationType:OWL:ObjectProperty"  
2690 code="ObjectProperty" id="urn:oasis:names:tc:ebxml-  
2691 regrep:profile:webontology:AssociationType:OWL:ObjectProperty">  
2692 <rim:Name>  
2693 <rim:LocalizedString charset="UTF-8"  
2694 value="ObjectProperty"/>  
2695 </rim:Name>  
2696 </rim:ClassificationNode>  
2697 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2698 regrep:profile:webontology:AssociationType:OWL"  
2699 lid="urn:oasis:names:tc:ebxml-  
2700 regrep:profile:webontology:AssociationType:OWL:HasProperty"  
2701 code="Property" id="urn:oasis:names:tc:ebxml-  
2702 regrep:profile:webontology:AssociationType:OWL:HasProperty">  
2703 <rim:Name>  
2704 <rim:LocalizedString charset="UTF-8" value="Property"/>  
2705 </rim:Name>  
2706 </rim:ClassificationNode>  
2707 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2708 regrep:profile:webontology:AssociationType:OWL"  
2709 lid="urn:oasis:names:tc:ebxml-  
2710 regrep:profile:webontology:AssociationType:OWL:SubPropertyOf"  
2711 code="SubPropertyOf" id="urn:oasis:names:tc:ebxml-  
2712 regrep:profile:webontology:AssociationType:OWL:SubPropertyOf">  
2713 <rim:Name>
```

```

2714         <rim:LocalizedString charset="UTF-8" value="SubPropertyOf"/>
2715         </rim:Name>
2716     </rim:ClassificationNode>
2717     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2718     regrep:profile:webontology:AssociationType:OWL"
2719     lid="urn:oasis:names:tc:ebxml-
2720     regrep:profile:webontology:AssociationType:OWL:SubClassOf"
2721     code="SubClassOf" id="urn:oasis:names:tc:ebxml-
2722     regrep:profile:webontology:AssociationType:OWL:SubClassOf">
2723         <rim:Name>
2724             <rim:LocalizedString charset="UTF-8" value="SubClassOf"/>
2725         </rim:Name>
2726     </rim:ClassificationNode>
2727
2728     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2729     regrep:profile:webontology:AssociationType:OWL "
2730     lid="urn:oasis:names:tc:ebxml-
2731     regrep:profile:webontology:AssociationType:OWL:IntersectionOf"
2732     code="IntersectionOf" id="urn:oasis:names:tc:ebxml-
2733     regrep:profile:webontology:AssociationType:OWL:IntersectionOf">
2734         <rim:Name>
2735             <rim:LocalizedString charset="UTF-8"
2736     value="IntersectionOf"/>
2737         </rim:Name>
2738     </rim:ClassificationNode>
2739
2740     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2741     regrep:profile:webontology:AssociationType:OWL"
2742     lid="urn:oasis:names:tc:ebxml-
2743     regrep:profile:webontology:AssociationType:OWL:SameAs" code="SameAs"
2744     id="urn:oasis:names:tc:ebxml-
2745     regrep:profile:webontology:AssociationType:OWL:SameAs">
2746         <rim:Name>
2747             <rim:LocalizedString charset="UTF-8" value="SameAs"/>
2748         </rim:Name>
2749     </rim:ClassificationNode>
2750
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:Restriction"
2755     code="restriction" id="urn:oasis:names:tc:ebxml-
2756     regrep:profile:webontology:AssociationType:OWL:Restriction">
2757         <rim:Name>
2758             <rim:LocalizedString charset="UTF-8" value="Restriction"/>
2759         </rim:Name>
2760     </rim:ClassificationNode>
2761
2762     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2763     regrep:profile:webontology:AssociationType:OWL "
2764     lid="urn:oasis:names:tc:ebxml-
2765     regrep:profile:webontology:AssociationType:OWL:DifferentFrom"
2766     code="DifferentFrom" id="urn:oasis:names:tc:ebxml-
2767     regrep:profile:webontology:AssociationType:OWL:DifferentFrom">
2768         <rim:Name>
2769             <rim:LocalizedString charset="UTF-8" value="DifferentFrom"/>
2770         </rim:Name>
2771     </rim:ClassificationNode>
2772
2773     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2774     regrep:profile:webontology:AssociationType:OWL "
2775     lid="urn:oasis:names:tc:ebxml-
2776     regrep:profile:webontology:AssociationType:OWL:DatatypeProperty"
2777     code="DatatypeProperty" id="urn:oasis:names:tc:ebxml-
2778     regrep:profile:webontology:AssociationType:OWL:DatatypeProperty">
2779         <rim:Name>

```

```

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

```

```

2844 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2845 regrep:profile:webontology:AssociationType:OWL "
2846 lid="urn:oasis:names:tc:ebxml-
2847 regrep:profile:webontology:AssociationType:OWL:SeeAlso" code="SeeAlso"
2848 id="urn:oasis:names:tc:ebxml-
2849 regrep:profile:webontology:AssociationType:OWL:SeeAlso">
2850   <rim:Name>
2851     <rim:LocalizedString charset="UTF-8" value="SeeAlso"/>
2852   </rim:Name>
2853 </rim:ClassificationNode>

```

2854 Extensions to the AssociationType ClassificationScheme

2855 7.3 Canonical Queries

2856 The following new canonical queries are described by this profile. Note that while these queries are
2857 complex, the complexity is hidden from clients by exposing only the query parameters to them.

2858 7.3.1 All SuperProperties Discovery Query

2859 Recursion is not supported by SQL-92, for this reason the stored procedure for this query coded in SQL
2860 99 Standard is available from:

2861 <http://www.srdc.metu.edu.tr/ebxml/ebXMLRegistryProfileForOWL/StoredProceduresSupportingebXMLRegistryProfileforOWL.htm>.

2863

2864 7.3.2 Immediate SuperClass Discovery Query

```

2865 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2866 regrep:profile:webontology:query:FindImmediateSuperClasses"
2867 id="urn:oasis:names:tc:ebxml-
2868 regrep:profile:webontology:query:FindImmediateSuperClasses">
2869   <rim:Name>
2870     <rim:LocalizedString
2871 value="label.FindImmediateSuperClasses"/>
2872   </rim:Name>
2873   <rim:Description>
2874     <rim:LocalizedString
2875 value="label.FindImmediateSuperClasses.desc"/>
2876   </rim:Description>
2877   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2878 regrep:QueryLanguage:SQL-92">
2879     SELECT C2.*
2880     FROM ClassificationNode C2, Association A, Name_ N,
2881 ClassificationNode C1
2882     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2883 regrep:profile:webontology:AssociationType:OWL:SubClassOf' AND
2884     C1.id = N.parent AND
2885     N.value LIKE '$className' AND
2886     A.sourceObject = C1.id AND
2887     A.targetObject = C2.id
2888   </rim:QueryExpression>
2889 </rim:AdhocQuery>
2890

```

2891 **The Adhoc Query retrieving immediate super classes of a given classification node**

2892 7.3.3 Immediate SubClass Discovery Query

```

2893 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2894 regrep:profile:webontology:query:FindImmediateSubClasses"
2895 id="urn:oasis:names:tc:ebxml-
2896 regrep:profile:webontology:query:FindImmediateSubClasses">
2897   <rim:Name>

```



```

2898         <rim:LocalizedString value="label.FindImmediateSubClasses"/>
2899     </rim:Name>
2900     <rim:Description>
2901         <rim:LocalizedString
2902 value="label.FindImmediateSubClasses.desc"/>
2903     </rim:Description>
2904     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2905 regrep:QueryLanguage:SQL-92">
2906         SELECT C2.*
2907         FROM ClassificationNode C2, Association A, Name_ N,
2908 ClassificationNode C1
2909         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2910 regrep:profile:webontology:AssociationType:OWL:SubClassOf' AND
2911         C1.id = N.parent AND
2912         N.value LIKE '$className' AND
2913         A.sourceObject = C2.id AND
2914         A.targetObject = C1.id
2915     </rim:QueryExpression>
2916 </rim:AdhocQuery>

```

2917 **The Adhoc Query retrieving immediate subclasses of a given classification node**

2918 7.3.4 All SuperClasses Discovery Query

2919 Recursion is not supported by SQL-92, for this reason the stored procedure for this query coded in SQL
2920 99 Standard is available from:

2921 <http://www.srdc.metu.edu.tr/ebxml/ebXMLRegistryProfileForOWL/StoredProceduresSupportingebXMLRegistryProfileforOWL.htm>.

2923

2924 7.3.5 All SubClasses Discovery Query

2925 Recursion is not supported by SQL-92, for this reason the stored procedure for this query coded in SQL
2926 99 Standard is available from:

2927 <http://www.srdc.metu.edu.tr/ebxml/ebXMLRegistryProfileForOWL/StoredProceduresSupportingebXMLRegistryProfileforOWL.htm>.

2928

2929 7.3.6 EquivalentClasses Discovery Query

```

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

```

2954 **Adhoc Query retrieving all the equivalent classes of a given classification node**

2955 7.3.7 EquivalentProperties Discovery Query

```
2956 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2957 regrep:profile:webontology:query:FindEquivalentProperties"
2958 id="urn:oasis:names:tc:ebxml-
2959 regrep:profile:webontology:query:FindEquivalentProperties">
2960 <rim:Name>
2961 <rim:LocalizedString
2962 value="label.FindEquivalentProperties"/>
2963 </rim:Name>
2964 <rim:Description>
2965 <rim:LocalizedString
2966 value="label.FindEquivalentProperties.desc"/>
2967 </rim:Description>
2968 <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2969 regrep:QueryLanguage:SQL-92">
2970 SELECT A3.*
2971 FROM Association A3, Association A1, Name_ N, Association
2972 A2
2973 WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
2974 regrep:profile:webontology:AssociationType:OWL:EquivalentTo' AND
2975 A2.id = N.parent AND
2976 N.value LIKE '$propertyName' AND
2977 A1.sourceObject = A2.id AND
2978 A1.targetObject = A3.id
2979 </rim:QueryExpression>
2980 </rim:AdhocQuery>
```

2981 **Adhoc Query retrieving all the equivalent Association Type of a given Association Type**

2982 7.3.8 SameExtrinsicObjects Discovery Query

```
2983 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2984 regrep:profile:webontology:query:FindTheSameExtrinsicObjects"
2985 id="urn:oasis:names:tc:ebxml-
2986 regrep:profile:webontology:query:FindTheSameExtrinsicObjects">
2987 <rim:Name>
2988 <rim:LocalizedString
2989 value="label.FindTheSameExtrinsicObjects"/>
2990 </rim:Name>
2991 <rim:Description>
2992 <rim:LocalizedString
2993 value="label.FindTheSameExtrinsicObjects.desc"/>
2994 </rim:Description>
2995 <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2996 regrep:QueryLanguage:SQL-92">
2997 SELECT E2.*
2998 FROM ExtrinsicObject E2, Association A, Name_ N,
2999 ExtrinsicObject E
3000 WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3001 regrep:profile:webontology:AssociationType:OWL:SameAs' AND
3002 E.id = N.parent AND
3003 N.value LIKE '$extrinsicObjectName' AND
3004 A.sourceObject = E.id AND
3005 A.targetObject = E2.id
3006 </rim:QueryExpression>
3007 </rim:AdhocQuery>
```

3008 **Adhoc Query retrieving all the "ExtrinsicObjects" defined to be the same with a given**
3009 **ExtrinsicObject**

3010 7.3.9 DifferentExtrinsicObjects Discovery Query

```
3011 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3012 regrep:profile:webontology:query:FindDifferentExtrinsicObjects"
3013 id="urn:oasis:names:tc:ebxml-
3014 regrep:profile:webontology:query:FindDifferentExtrinsicObjects">
3015   <rim:Name>
3016     <rim:LocalizedString
3017 value="label.FindDifferentExtrinsicObjects"/>
3018   </rim:Name>
3019   <rim:Description>
3020     <rim:LocalizedString
3021 value="label.FindDifferentExtrinsicObjects.desc"/>
3022   </rim:Description>
3023   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3024 regrep:QueryLanguage:SQL-92">
3025     SELECT E2.*
3026     FROM ExtrinsicObject E2, Association A, Name_ N,
3027 ExtrinsicObject E
3028     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3029 regrep:profile:webontology:AssociationType:OWL:DifferentFrom' AND
3030     E.id = N.parent AND
3031     N.value LIKE '$extrinsicObjectName' AND
3032     A.sourceObject = E.id AND
3033     A.targetObject = E2.id
3034   </rim:QueryExpression>
3035 </rim:AdhocQuery>
```

3036 **Adhoc Query retrieving all the "ExtrinsicObjects" defined to be different from a given**
3037 **ExtrinsicObject**

3038 7.3.10 AllDifferentRegistryObject Discovery Query

```
3039 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3040 regrep:profile:webontology:query:FindAllDifferent"
3041 id="urn:oasis:names:tc:ebxml-
3042 regrep:profile:webontology:query:FindAllDifferent">
3043   <rim:Name>
3044     <rim:LocalizedString value="label.FindAllDifferent"/>
3045   </rim:Name>
3046   <rim:Description>
3047     <rim:LocalizedString value="label.FindAllDifferent.desc"/>
3048   </rim:Description>
3049   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3050 regrep:QueryLanguage:SQL-92">
3051     SELECT RO2.*
3052     FROM RegistryObject RO2, Association A1, Association A2,
3053 Name_ N, RegistryObject RO,
3054 RegistryPackage RP<!--, Slot S-->
3055     WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
3056 regrep:profile:webontology:AssociationType:OWL:HasMember' AND
3057     RO.id = N.parent AND
3058     N.value LIKE '$registryObjectName' AND
3059     A1.sourceObject = RP.id AND
3060     <!-- S.parent = RP.id AND
3061     S.name_ LIKE 'packageType' AND S.value LIKE
3062 'allDifferent' AND -->
3063     A1.targetObject = RO.id AND
3064     A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3065 regrep:profile:webontology:AssociationType:OWL:HasMember' AND
3066     A2.sourceObject = RP.id AND
3067     A2.targetObject != RO.id AND
3068     A2.targetObject = RO2.id
3069   </rim:QueryExpression>
3070 </rim:AdhocQuery>
```

3071 **Adhoc Query retrieving all the "RegistryObjects" defined to be different from a given**

3072 RegistryObject through a "allDifferent" construct

3073 7.3.11 ObjectProperties Discovery Query

```
3074 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3075 regrep:profile:webontology:query:FindObjectProperties"
3076 id="urn:oasis:names:tc:ebxml-
3077 regrep:profile:webontology:query:FindObjectProperties">
3078   <rim:Name>
3079     <rim:LocalizedString value="label.FindObjectProperties"/>
3080   </rim:Name>
3081   <rim:Description>
3082     <rim:LocalizedString
3083 value="label.FindObjectProperties.desc"/>
3084   </rim:Description>
3085   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3086 regrep:QueryLanguage:SQL-92">
3087     SELECT A.*
3088     FROM Association A, Name_N, ClassificationNode C
3089     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3090 regrep:profile:webontology:AssociationType:OWL:ObjectProperty' AND
3091     C.id = N.parent AND
3092     N.value LIKE '$className' AND
3093     A.sourceObject = C.id
3094   </rim:QueryExpression>
3095 </rim:AdhocQuery>
```

3096 **Adhoc Query retrieving all the object properties of a given classification node**

3097 7.3.12 ImmediateInheritedObjectProperties Discovery Query

```
3098 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3099 regrep:profile:webontology:query:FindImmediateInheritedObjectProperties"
3100 id="urn:oasis:names:tc:ebxml-
3101 regrep:profile:webontology:query:FindImmediateInheritedObjectProperties">
3102   <rim:Name>
3103     <rim:LocalizedString
3104 value="label.FindImmediateInheritedObjectProperties"/>
3105   </rim:Name>
3106   <rim:Description>
3107     <rim:LocalizedString
3108 value="label.FindImmediateInheritedObjectProperties.desc"/>
3109   </rim:Description>
3110   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3111 regrep:QueryLanguage:SQL-92">
3112     SELECT A2.*
3113     FROM Association A, Name_N, ClassificationNode C1,
3114 ClassificationNode C2, Association A2
3115     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3116 regrep:profile:webontology:AssociationType:OWL:SubClassOf' AND
3117     C1.id = N.parent AND
3118     N.value LIKE '$className' AND
3119     A.sourceObject = C1.id AND
3120     A.targetObject = C2.id AND
3121     A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3122 regrep:profile:webontology:AssociationType:OWL:ObjectProperty' AND
3123     A2.sourceObject=C2.id
3124   </rim:QueryExpression>
3125 </rim:AdhocQuery>
```

3126 **Adhoc Query retrieving all of the properties of a given classification node including the ones**
3127 **inherited from its immediate super classes**

3128 7.3.13 AllInheritedObjectProperties Discovery Query

3129 Recursion is not supported by SQL-92, for this reason the stored procedure for this query coded in SQL

3130 99 Standard is available from:
3131 <http://www.srdc.metu.edu.tr/ebxml/ebXMLRegistryProfileForOWL/StoredProceduresSupportingebXMLRegistryProfileforOWL.htm>
3132

3133 7.3.14 DatatypeProperties Discovery Query

```
3134 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-  
3135 regrep:profile:webontology:query:FindDatatypeProperties"  
3136 id="urn:oasis:names:tc:ebxml-  
3137 regrep:profile:webontology:query:FindDatatypeProperties">  
3138   <rim:Name>  
3139     <rim:LocalizedString value="label.FindDatatypeProperties"/>  
3140   </rim:Name>  
3141   <rim:Description>  
3142     <rim:LocalizedString  
3143 value="label.FindDatatypeProperties.desc"/>  
3144   </rim:Description>  
3145   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
3146 regrep:QueryLanguage:SQL-92">  
3147     SELECT A.*  
3148     FROM Association A, Name_N, ClassificationNode C  
3149     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-  
3150 regrep:profile:webontology:AssociationType:OWL:DatatypeProperty' AND  
3151     C.id = N.parent AND  
3152     N.value LIKE '$className' AND  
3153     A.sourceObject = C.id  
3154   </rim:QueryExpression>  
3155 </rim:AdhocQuery>
```

3156 **Adhoc Query retrieving all the datatype properties of a given classification node**

3157 7.3.15 AllInheritedDatatypeProperties Discovery Query

3158 Recursion is not supported by SQL-92, for this reason the stored procedure for this query coded in SQL
3159 99 Standard is available from:

3160 <http://www.srdc.metu.edu.tr/ebxml/ebXMLRegistryProfileForOWL/StoredProceduresSupportingebXMLRegistryProfileforOWL.htm>
3161

3162 7.3.16 TransitiveRelationships Discovery Query

```
3163 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-  
3164 regrep:profile:webontology:query:FindTransitiveRelationships"  
3165 id="urn:oasis:names:tc:ebxml-  
3166 regrep:profile:webontology:query:FindTransitiveRelationships">  
3167   <rim:Name>  
3168     <rim:LocalizedString  
3169 value="label.FindTransitiveRelationships"/>  
3170   </rim:Name>  
3171   <rim:Description>  
3172     <rim:LocalizedString  
3173 value="label.FindTransitiveRelationships.desc"/>  
3174   </rim:Description>  
3175   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
3176 regrep:QueryLanguage:SQL-92">  
3177     SELECT C.*  
3178     FROM ClassificationNode C, Association A1, Association A2,  
3179     Name_ N1, Name_ N2, Name_ N3  
3180     WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-  
3181 regrep:profile:webontology:AssociationType:OWL:TransitiveProperty' AND  
3182     A1.id = N1.parent AND  
3183     N1.value LIKE '$propertyName' AND  
3184     A1.sourceObject = N3.parent AND  
3185     N3.value LIKE '$className' AND  
3186     A2.sourceObject = A1.targetObject AND
```

```

3187         A2.id = N2.parent AND
3188         N2.value LIKE '$propertyName' AND
3189         A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
regrep:profile:webontology:AssociationType:OWL:TransitiveProperty' AND
3190         A2.targetObject = C.id
3191         <!-- UNION
3192         SELECT C.*
3193         FROM ClassificationNode C, Association A1, Name_ N1, Name_
3194         N3
3195         WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
regrep:profile:webontology:AssociationType:OWL:TransitiveProperty' AND
3196         A1.id = N1.parent AND
3197         N1.value LIKE '$propertyName' AND
3198         A1.sourceObject = N3.parent AND
3199         N3.value LIKE '$className' AND
3200         A1.targetObject = C.id -->
3201     </rim:QueryExpression>
3202 </rim:AdhocQuery>

```

3205 **Adhoc Query retrieving the objects in transitive relationship with a given object**

3206 7.3.17 TargetObjects Discovery Query

```

3207 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3208 regrep:profile:webontology:query:FindTargetObjects"
3209 id="urn:oasis:names:tc:ebxml-
3210 regrep:profile:webontology:query:FindTargetObjects">
3211     <rim:Name>
3212         <rim:LocalizedString value="label.FindTargetObjects"/>
3213     </rim:Name>
3214     <rim:Description>
3215         <rim:LocalizedString value="label.FindTargetObjects.desc"/>
3216     </rim:Description>
3217     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3218 regrep:QueryLanguage:SQL-92">
3219         SELECT C2.*
3220         FROM ClassificationNode C2, Association A, Name_ N, Name_
3221         N2, ClassificationNode C1
3222         WHERE A.id=N2.parent AND
3223         N2.value LIKE '$propertyName' AND
3224         C1.id = N.parent AND
3225         N.value LIKE '$className' AND
3226         A.sourceObject = C1.id AND
3227         A.targetObject = C2.id
3228     </rim:QueryExpression>
3229 </rim:AdhocQuery>

```

3230 **Adhoc Query retrieving the Target Objects from the Registry, given a Source Object and an**
3231 **Association**

3232 7.3.18 TargetObjectsInverseOf Discovery Query

```

3233 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3234 regrep:profile:webontology:query:FindTOinverseOf"
3235 id="urn:oasis:names:tc:ebxml-
3236 regrep:profile:webontology:query:FindTOinverseOf">
3237     <rim:Name>
3238         <rim:LocalizedString value="label.FindTOinverseOf"/>
3239     </rim:Name>
3240     <rim:Description>
3241         <rim:LocalizedString value="label.FindTOinverseOf.desc"/>
3242     </rim:Description>
3243     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3244 regrep:QueryLanguage:SQL-92">
3245         SELECT C2.*
3246         FROM ClassificationNode C2, Association A1, Association A2,
3247         Association A3, Name_ N, Name_ N2, ClassificationNode C1

```

```

3248         WHERE A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3249 regrep:profile:webontology:AssociationType:OWL:InverseOf' AND
3250         A1.id = N.parent AND
3251         N.value LIKE '$propertyName' AND
3252         A2.sourceObject = A1.id AND
3253         A2.targetObject = A3.id AND
3254         C1.id = N2.parent AND
3255         N2.value LIKE '$className' AND
3256         A3.targetObject = C1.id AND
3257         A3.sourceObject = C2.id
3258     </rim:QueryExpression>
3259 </rim:AdhocQuery>

```

3260 **Adhoc query retrieving the Source Objects of an Association which is in "inverseOf"**
3261 **relationship to this Association**

3262 7.3.19 InverseRanges Discovery Query

```

3263 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3264 regrep:profile:webontology:query:FindInverseRanges"
3265 id="urn:oasis:names:tc:ebxml-
3266 regrep:profile:webontology:query:FindInverseRanges">
3267     <rim:Name>
3268         <rim:LocalizedString value="label.FindInverseRanges"/>
3269     </rim:Name>
3270     <rim:Description>
3271         <rim:LocalizedString value="label.FindInverseRanges.desc"/>
3272     </rim:Description>
3273     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3274 regrep:QueryLanguage:SQL-92">
3275         <!-- SELECT C2.*
3276         FROM Association A, Name_ N, Name_ N2, ClassificationNode
3277 C1, ClassificationNode C2
3278         WHERE A.id=N2.parent AND
3279         N2.value LIKE '$propertyName' AND
3280         C1.id = N.parent AND
3281         N.value LIKE '$className' AND
3282         A.sourceObject = C1.id AND
3283         A.targetObject = C2.id
3284         UNION -->
3285         SELECT C2.*
3286         FROM ClassificationNode C2, Association A1, Association A2,
3287 Association A3, Name_ N, NAME_ N2, ClassificationNode C1
3288         WHERE A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3289 regrep:profile:webontology:AssociationType:OWL:InverseOf' AND
3290         A1.id = N.parent AND
3291         N.value LIKE '$propertyName' AND
3292         A2.sourceObject = A1.id AND
3293         A2.targetObject = A3.id AND
3294         C1.id = N2.parent AND
3295         N2.value LIKE '$className' AND
3296         A1.sourceObject = C1.id AND
3297         A3.sourceObject = C2.id
3298     </rim:QueryExpression>
3299 </rim:AdhocQuery>

```

3300 **Adhoc Query Retrieving both the Target Objects of a given Association and the Source**
3301 **Objects of an Association which is in "inverseOf" relationship to this Association**

3302 7.3.20 SymmetricProperties Discovery Query

```

3303 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3304 regrep:profile:webontology:query:FindSymmetricProperties"
3305 id="urn:oasis:names:tc:ebxml-
3306 regrep:profile:webontology:query:FindSymmetricProperties">
3307     <rim:Name>
3308         <rim:LocalizedString value="label.FindSymmetricProperties"/>

```

```

3309     </rim:Name>
3310     <rim:Description>
3311         <rim:LocalizedString
3312 value="label.FindSymmetricProperties.desc"/>
3313     </rim:Description>
3314     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3315 regrep:QueryLanguage:SQL-92">
3316         SELECT A.*
3317         FROM Association A, Name_N, ClassificationNode C
3318         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3319 regrep:profile:webontology:AssociationType:OWL:SymmetricProperty' AND
3320         C.id = N.parent AND
3321         N.value LIKE '$className' AND
3322         A.sourceObject = C.id
3323     </rim:QueryExpression>
3324 </rim:AdhocQuery>

```

3325 **Adhoc Query retrieving all the Symmetric properties of a given classification node**

3326 7.3.21 FunctionalProperties Discovery Query

```

3327 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3328 regrep:profile:webontology:query:FindFunctionalProperties"
3329 id="urn:oasis:names:tc:ebxml-
3330 regrep:profile:webontology:query:FindFunctionalProperties">
3331     <rim:Name>
3332         <rim:LocalizedString
3333 value="label.FindFunctionalProperties"/>
3334     </rim:Name>
3335     <rim:Description>
3336         <rim:LocalizedString
3337 value="label.FindFunctionalProperties.desc"/>
3338     </rim:Description>
3339     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3340 regrep:QueryLanguage:SQL-92">
3341         SELECT A.*
3342         FROM Association A, Name_N, ClassificationNode C
3343         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3344 regrep:profile:webontology:AssociationType:OWL:FunctionalProperty' AND
3345         C.id = N.parent AND
3346         N.value LIKE '$className' AND
3347         A.sourceObject = C.id
3348     </rim:QueryExpression>
3349 </rim:AdhocQuery>

```

3350 **Adhoc Query retrieving all the Functional properties of a given classification node**

3351 7.3.22 InverseFunctionalProperties Discovery Query

```

3352 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3353 regrep:profile:webontology:query:FindInverseFunctionalProperties"
3354 id="urn:oasis:names:tc:ebxml-
3355 regrep:profile:webontology:query:FindInverseFunctionalProperties">
3356     <rim:Name>
3357         <rim:LocalizedString
3358 value="label.FindInverseFunctionalProperties"/>
3359     </rim:Name>
3360     <rim:Description>
3361         <rim:LocalizedString
3362 value="label.FindInverseFunctionalProperties.desc"/>
3363     </rim:Description>
3364     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3365 regrep:QueryLanguage:SQL-92">
3366         SELECT A.*
3367         FROM Association A, Name_N, ClassificationNode C

```



```

3368         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3369 regrep:profile:webontology:AssociationType:OWL:InverseFunctionalProperty'
3370 ' AND
3371         C.id = N.parent AND
3372         N.value LIKE '$className' AND
3373         A.sourceObject = C.id
3374     </rim:QueryExpression>
3375 </rim:AdhocQuery>

```

3376 **Adhoc Query retrieving all the Inverse Functional properties of a given classification node**

3377 7.3.23 Instances Discovery Query Discovery Query

```

3378 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3379 regrep:profile:webontology:query:FindInstances"
3380 id="urn:oasis:names:tc:ebxml-
3381 regrep:profile:webontology:query:FindInstances">
3382     <rim:Name>
3383         <rim:LocalizedString value="label.FindInstances"/>
3384     </rim:Name>
3385     <rim:Description>
3386         <rim:LocalizedString value="label.FindInstances.desc"/>
3387     </rim:Description>
3388     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3389 regrep:QueryLanguage:SQL-92">
3390         <!-- SELECT S.* FROM Service S, (
3391         SELECT S1.value AS id
3392         FROM Slot S1, Name_ N, ClassificationNode C
3393         WHERE S1.parent = C.id AND
3394
3395
3396         C.id = N.parent AND
3397         N.value LIKE '$className' AND
3398         S1.name_ LIKE 'urn:oasis:names:tc:ebxml-
3399 regrep:profile:webontology:slot:intersectionOf '
3400         )
3401         AS T1, (
3402         SELECT S1.value AS id
3403         FROM Slot S1, Name_ N, ClassificationNode C
3404         WHERE S1.parent = C.id AND
3405         C.id = N.parent AND
3406         N.value LIKE '$className' AND
3407         S1.name_ LIKE 'urn:oasis:names:tc:ebxml-
3408 regrep:profile:webontology:slot:intersectionOf '
3409         ) AS T2
3410         WHERE S.id IN (
3411         SELECT classifiedObject
3412         FROM Classification
3413         WHERE classificationNode=T1.id
3414         INTERSECT
3415         SELECT classifiedObject
3416         FROM Classification
3417         WHERE classificationNode=T2.id
3418         ) AND T1.id!=T2.id
3419         UNION -->
3420         SELECT S.*
3421         FROM Service S, Classification C, ClassificationNode CN,
3422         Name_ N
3423         WHERE S.id = C. classifiedObject AND
3424         C.classificationNode = CN.id AND
3425         N.value LIKE '$className' AND
3426         N.parent = CN.id
3427     </rim:QueryExpression>
3428 </rim:AdhocQuery>

```

3429 **Adhoc Query Retrieving the instances of intersected classes**

3430 8 OWL Profile References

3431 8.1 Normative References

- 3432 [Bechhofer, Harmelen, Hendler, Horrocks, McGuinness, Patel-Schneider, Stein]
3433 Bechhofer, S., Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., Stein, L.
3434 A., OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004
3435 <http://www.w3.org/TR/2004/REC-owl-ref-20040210/>
3436
- 3437 [Brickley, Guha] Brickley, D., Guha, R.V., RDF Vocabulary Description Language 1.0: RDF Schema
3438 W3C Recommendation 10 February 2004
3439 <http://www.w3.org/TR/rdf-schema/>
3440
- 3441 [DAML+OIL] <http://www.daml.org/>
3442 [ebRIM] ebXML Registry Information Model version 3.0
3443 <http://docs.oasis-open.org/regrep/regrep-rim/v3.0/regrep-rim-3.0-os.pdf>
3444
- 3445 [ebRS] ebXML Registry Services Specification version 3.0
3446 <http://docs.oasis-open.org/regrep/regrep-rs/v3.0/regrep-rs-3.0-os.pdf>
- 3447 [ebRR-DPT] Deployment Profile Template For ebXML Registry 3.0 OASIS Specifications V_0.1.2
- 3448 [ebMS-DPT] Deployment Profile Template For OASIS Specification ebXML Message Service 2.0
- 3449 [McGuinness, Harmelen] McGuinness, D. L., Harmelen, F., OWL Web Ontology Language Overview,
3450 W3C Recommendation 10 February 2004, <http://www.w3.org/TR/owl-features/>
- 3451 [OWL] Web Ontology Language (OWL), <http://www.w3.org/2004/OWL/>
- 3452 [RDF] Resource Description Framework, <http://www.w3.org/TR/rdf-concepts/>
- 3453 [RDFS] RDF Vocabulary Description Language 1.0: RDF Schema <http://www.w3.org/TR/rdf-schema/>
- 3454 [Smith, Welty, McGuinness] Smith, M. K., Welty, C., McGuinness, D. L.,
3455 OWL Web Ontology Language Guide, W3C Recommendation 10 February 2004,
3456 <http://www.w3.org/TR/owl-guide/>
- 3457 [SQL 92] SQL ISO/IEC 9075:1992 Information technology - Database languages - SQL.
- 3458 [SQL 99] ISO/IEC 9075:1999(E) Information technology - Database languages – SQL.
- 3459 [UML] Unified Modeling Language version 1.5
3460 <http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.pdf>
- 3461 [WSDL] WSDL Specification
3462 <http://www.w3.org/TR/wsdl>

3463 **8.2 Informative References**

- 3464 [Dogac, et. al. 2005] Dogac A., Kabak Y., Laleci G. C. Mattocks, F. Najmi, J. Pollock
3465 Enhancing ebXML Registries to Make them OWL Aware
3466 Distributed and Parallel Databases Journal, Springer-Verlag, Vol. 18, No. 1, July 2005, pp. 9-36.
3467
- 3468 [Dogac et. al. 2006] Dogac A., Laleci G., Kabak Y., Unal S., Beale T., Heard S., Elkin P., Najmi F.,
3469 Mattocks C., Webber D., Kernberg M.
3470 Exploiting ebXML Registry Semantic Constructs for Handling Archetype Metadata in Healthcare
3471 Informatics
3472 International Journal of Metadata, Semantics and Ontologies, Volume 1, No. 1, 2006.
3473
- 3474 [IMPL] ebXML Registry 3.0 Implementations
3475 freebXML Registry: A royalty free, open source ebXML Registry Implementation
3476 <http://ebxmlrr.sourceforge.net>
3477
- 3478 [LeeHendler]
3479 Berners-Lee, T., Hendler, J., Lassila, O., "The Semantic Web", Scientific American, May 2001.
3480
- 3481 [StaabStuder] Staab, S., Studer, R., Handbook on Ontologies, Springer, 2004.