



ebXML Registry Profile for Web Ontology

Language (OWL)

Version 1.0 Draft 2

Draft OASIS Profile, January 27, 2006

Document identifier:

regrep-owl-profile-1.0-draft 2

Location:

<http://www.oasis-open.org/committees/regrep-semantic/documents/profile/regrep-owl-profile-1.0-draft-1.pdf>

Editors:

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
....	

Abstract:

This document defines the ebXML Registry profile for enhancing ebXML Registry with OWL semantics to make it OWL aware.

19

20 **Status:**

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

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

29 For information on whether any patents have been disclosed that may be essential to
30 implementing this specification, and any offers of patent licensing terms, please refer to the
31 Intellectual Property Rights section of the OASIS ebXML Registry TC web page (<http://www.oasis-open.org/committees/regrep/>).
32

1 Table of Contents

33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77

1 Table of Contents.....	3
1 Introduction.....	10
1.1 Terminology.....	10
1.2 Conventions.....	11
2 OWL Overview.....	12
2.1 RDF Schema Features.....	12
2.2 (In)Equality.....	12
2.3 Property Characteristics	12
2.4 Property Restrictions.....	13
2.5 Restricted Cardinality.....	13
2.6 Class Intersection.....	13
2.7 Versioning.....	13
2.8 Annotation Properties	13
2.9 Datatypes	13
3 ebXML Registry Overview.....	14
3.1 Overview of [ebRIM].....	14
3.1.1 RegistryObject.....	15
3.1.2 Object Identification.....	15
3.1.3 Object Naming and Description.....	16
3.1.4 Object Attributes.....	16
3.1.4.1 Slot Attributes.....	16
3.1.5 Object Classification.....	17
3.1.6 Object Association.....	17
3.1.7 Object References To Web Content.....	18
3.1.8 Object Packaging.....	18
3.1.9 ExtrinsicObject	19
3.1.10 Service Description.....	19
3.2 Overview of [ebRS].....	19
4 Representing OWL Constructs in ebRIM and Providing Processing Support for Additional Semantics..	20
4.1 Representing RDF Schema Features in ebRIM.....	20
4.1.1 owl:Class → rim:ClassificationNode.....	20
4.1.2 rdf:Property → rim:Association Type Property.....	20
4.1.3 rdfs:subPropertyOf → rim:Association Type subPropertyOf.....	21
4.1.4 rdfs:subClassOf → rim:Association Type subClassOf.....	22
4.1.5 owl:Individual → rim:ExtrinsicObject.....	25
4.2 Representing OWL (In)Equality Constructs in ebXML RIM.....	26
4.2.1 owl:equivalentClass, owl:equivalentProperty → rim:Association Type EquivalentTo	26
4.2.2 owl:sameAs → rim:Association Type sameAs.....	27
4.2.3 owl:differentFrom → rim:Association Type differentFrom.....	27
4.2.4 owl:AllDifferent.....	28
4.3 Representing OWL Property Characteristics in ebRIM.....	29
4.3.1 owl:ObjectProperty → rim:Association Type objectProperty.....	29
4.3.2 owl:DatatypeProperty → rim:Association Type DatatypeProperty.....	32
4.3.3 owl:TransitiveProperty → rim:Association Type transitiveProperty.....	33

78	4.3.4 owl:inverseOf → rim:Association Type inverseOf.....	35
79	4.3.4.1 Retrieving the Target Objects of a given Association of a given ClassificationNode.....	35
80	4.3.4.2 Retrieving the Target Objects from the Registry which are in "inverseOf" relationship to a given	
81	Association of a given ClassificationNode.....	36
82	4.3.4.3 A Clarifying Example.....	36
83	4.3.5 owl:SymmetricProperty→ rim:Association Type SymmetricProperty.....	37
84	4.3.6 owl:FunctionalProperty→ rim:Association Type FunctionalProperty.....	38
85	4.3.7 owl:InverseFunctionalProperty→ rim:Association Type InverseFunctionalProperty.....	39
86	4.4 OWL Property Restrictions in ebXML RIM.....	40
87	4.5 Representing OWL Restricted Cardinality in ebXML RIM.....	41
88	4.5.1 owl:minCardinality (only 0 or 1).....	41
89	4.5.2 owl:maxCardinality (only 0 or 1).....	41
90	4.5.3 owl:cardinality.....	42
91	4.6 Representing OWL Class Intersection in ebXML RIM.....	42
92	4.7 Representing OWL Versioning in ebXML RIM.....	45
93	4.7.1 owl:versionInfo, owl:priorVersion.....	45
94	4.8 Representing OWL Annotation Properties in ebXML RIM.....	45
95	4.8.1 rdfs:label.....	45
96	4.8.2 rdfs:comment.....	46
97	4.8.3 rdfs:seeAlso.....	46
98	4.9 OWL Datatypes in ebXML RIM.....	46
99	5 Cataloging Service Profile.....	48
100	5.1 Invocation Control File.....	48
101	5.2 Input Metadata.....	48
102	5.3 Input Content.....	48
103	5.4 Output Metadata.....	49
104	5.4.1 Changes to Input Metadata.....	49
105	5.4.2 owl:Class → rim:ClassificationNode.....	49
106	5.4.3 rdf:Property → rim:Association Type Property.....	49
107	5.4.4 rdfs:subPropertyOf → rim:Association Type subPropertyOf.....	49
108	5.4.5 rdfs:subClassOf → rim:Association Type subClassOf.....	49
109	5.4.6 owl:Individual → rim:ExtrinsicObject.....	49
110	5.4.7 owl:equivalentClass, owl:equivalentProperty → rim:Association Type EquivalentTo	49
111	5.4.8 owl:sameAs → rim:Association Type sameAs	50
112	5.4.9 owl:differentFrom → rim:Association Type differentFrom.....	50
113	5.4.10 owl:AllDifferent → rim:RegistryPackage.....	50
114	5.4.11 owl:ObjectProperty → rim:Association Type objectProperty.....	50
115	5.4.12 owl:DatatypeProperty → rim:Association Type DatatypeProperty.....	50
116	5.4.13 owl:TransitiveProperty → rim:Association Type transitiveProperty.....	50
117	5.4.14 owl:inverseOf → rim:Association Type inverseOf.....	50
118	5.4.15 owl:SymmetricProperty→ rim:Association Type SymetricProperty.....	50
119	5.4.16 owl:FunctionalProperty→ rim:Association Type FunctionalProperty.....	50
120	5.4.17 owl:InverseFunctionalProperty→ rim:Association Type InverseFunctionalProperty.....	51
121	5.4.18 owl:minCardinality (only 0 or 1).....	51
122	5.4.19 owl:maxCardinality (only 0 or 1).....	51
123	5.4.20 owl:cardinality.....	51
124	5.4.21 owl:intersectionOf.....	51

125	5.4.22 rdfs:label.....	51
126	5.4.23 rdfs:comment.....	51
127	5.4.24 rdfs:seeAlso.....	51
128	6 Discovery Profile.....	52
129	6.1 All SuperProperties Discovery Query.....	52
130	6.1.1 Parameter \$propertyName.....	52
131	6.1.2 Example of All SuperProperties Discovery Query.....	52
132	6.2 Immediate SuperClass Discovery Query.....	53
133	6.2.1 Parameter \$className.....	53
134	6.2.2 Example of Immediate SuperClass Discovery Query.....	53
135	6.3 Immediate SubClass Discovery Query.....	53
136	6.3.1 Parameter \$className.....	54
137	6.3.2 Example of Immediate SubClasses Discovery Query.....	54
138	6.4 All SuperClasses Discovery Query.....	54
139	6.4.1 Parameter \$className.....	54
140	6.4.2 Example of All SuperClasses Discovery Query.....	54
141	6.5 All SubClasses Discovery Query.....	55
142	6.5.1 Parameter \$className.....	55
143	6.5.2 Example of All SubClasses Discovery Query.....	55
144	6.6 EquivalentClasses Discovery Query.....	56
145	6.6.1 Parameter \$className.....	56
146	6.6.2 Example of EquivalentClasses Discovery Query.....	56
147	6.7 EquivalentProperties Discovery Query.....	57
148	6.7.1 Parameter \$propertyName.....	57
149	6.7.2 Example of EquivalentProperties Discovery Query.....	57
150	6.8 SameExtrinsicObjects Discovery Query.....	57
151	6.8.1 Parameter \$extrinsicObjectName.....	57
152	6.8.2 Example of SameExtrinsicObjects Discovery Query.....	58
153	6.9 DifferentExtrinsicObjects Discovery Query.....	58
154	6.9.1 Parameter \$extrinsicObjectName.....	58
155	6.9.2 Example of DifferentExtrinsicObjects Discovery Query.....	58
156	6.10 AllDifferentRegistryObject Discovery Query.....	59
157	6.10.1 Parameter \$registryObjectName.....	59
158	6.10.2 Example of AllDifferentRegistryObjects Discovery Query.....	59
159	6.11 ObjectProperties Discovery Query.....	60
160	6.11.1 Parameter \$className.....	60
161	6.11.2 Example of ObjectProperties Discovery Query.....	60
162	6.12 ImmediateInheritedObjectProperties Discovery Query.....	61
163	6.12.1 Parameter \$className.....	61
164	6.12.2 Example of ImmediateInheritedObjectProperties Discovery Query.....	61
165	6.13 AllInheritedObjectProperties Discovery Query.....	61
166	6.13.1 Parameter \$className.....	62
167	6.13.2 Example of AllInheritedObjectProperties Discovery Query.....	62
168	6.14 DatatypeProperties Discovery Query.....	62
169	6.14.1 Parameter \$className.....	62
170	6.14.2 Example of DatatypeProperties Discovery Query.....	62

171	6.15 AllInheritedDatatypeProperties Discovery Query.....	63
172	6.15.1 Parameter \$className.....	63
173	6.15.2 Example of AllInheritedDatatypeProperties Discovery Query.....	63
174	6.16 TransitiveRelationships Discovery Query.....	64
175	6.16.1 Parameter \$className.....	64
176	6.16.2 Parameter \$propertyName.....	64
177	6.16.3 Example of TransitiveRelationships Discovery Query.....	64
178	6.17 TargetObjects Discovery Query.....	65
179	6.17.1 Parameter \$className.....	65
180	6.17.2 Parameter \$propertyName.....	65
181	6.17.3 Example of TargetObjects Discovery Query.....	65
182	6.18 TargetObjectsInverseOf Discovery Query.....	66
183	6.18.1 Parameter \$className.....	66
184	6.18.2 Parameter \$propertyName.....	66
185	6.18.3 Example of TargetObjectsInverseOf Discovery Query.....	66
186	6.19 InverseRanges Discovery Query.....	67
187	6.19.1 Parameter \$className.....	67
188	6.19.2 Parameter \$propertyName.....	67
189	6.19.3 Example of InverseRanges Discovery Query.....	67
190	6.20 SymmetricProperties Discovery Query.....	68
191	6.20.1 Parameter \$className.....	68
192	6.20.2 Example of SymmetricProperties Discovery Query.....	68
193	6.21 FunctionalProperties Discovery Query.....	69
194	6.21.1 Parameter \$className.....	69
195	6.21.2 Example of FunctionalProperties Discovery Query.....	69
196	6.22 InverseFunctionalProperties Discovery Query.....	70
197	6.22.1 Parameter \$className.....	70
198	6.22.2 Example of InverseFunctionalProperties Discovery Query.....	70
199	6.23 Instances Discovery Query.....	71
200	6.23.1 Parameter \$className.....	71
201	6.23.2 Example of Instances Discovery Query.....	71
202	7 Canonical Metadata Definitions.....	72
203	7.1 AssociationType Extensions.....	72
204	7.2 Canonical Queries.....	74
205	7.2.1 All SuperProperties Discovery Query.....	74
206	7.2.2 Immediate SuperClass Discovery Query.....	74
207	7.2.3 Immediate SubClass Discovery Query.....	75
208	7.2.4 All SuperClasses Discovery Query.....	75
209	7.2.5 All SubClasses Discovery Query.....	75
210	7.2.6 EquivalentClasses Discovery Query.....	76
211	7.2.7 EquivalentProperties Discovery Query.....	76
212	7.2.8 SameExtrinsicObjects Discovery Query.....	77
213	7.2.9 DifferentExtrinsicObjects Discovery Query.....	77
214	7.2.10 AllDifferentRegistryObject Discovery Query.....	78
215	7.2.11 ObjectProperties Discovery Query.....	78
216	7.2.12 ImmediateInheritedObjectProperties Discovery Query.....	79

217	7.2.13 AllInheritedObjectProperties Discovery Query.....	79
218	7.2.14 DatatypeProperties Discovery Query.....	80
219	7.2.15 AllInheritedDatatypeProperties Discovery Query.....	80
220	7.2.16 TransitiveRelationships Discovery Query.....	81
221	7.2.17 TargetObjects Discovery Query.....	81
222	7.2.18 TargetObjectsInverseOf Discovery Query.....	82
223	7.2.19 InverseRanges Discovery Query.....	82
224	7.2.20 SymmetricProperties Discovery Query.....	83
225	7.2.21 FunctionalProperties Discovery Query.....	83
226	7.2.22 InverseFunctionalProperties Discovery Query.....	84
227	7.2.23 Instances Discovery Query Discovery Query.....	84
228	8 OWL Profile References.....	86
229	8.1 Normative References.....	86
230	8.2 Informative References.....	86
231		

Illustration Index

Figure 1: ebXML Registry Information Model, High Level Public View.....	12
Figure 2: ebXML Registry Information Model, Inheritance View.....	13

232

Index of Tables

233

1 Introduction

234

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

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

243 However, currently semantics is becoming a much broader issue than it used to be since several
244 application domains are making use of ontologies to add the knowledge to their data and applications
245 [StaabStuder]. One of the driving forces for ontologies is the Semantic Web initiative [LeeHendler]. As a
246 part of this initiative, W3C's Web Ontology Working Group defined Web Ontology Language [OWL].

247 Naturally, there is lot to be gained from using a standard ontology definition language, like OWL, to
248 express semantics in ebXML registries.

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

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

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

268 The document is organized as follows:

- 269 • Chapter 1 provides an introduction to the rest of this document.
- 270 • Chapter 2 provides an overview of the Web Ontology Language.
- 271 • Chapter 3 provides an overview of the ebXML Registry standard.
- 272 • Chapter 4 specifies the mapping between Web Ontology Language constructs and ebXML
273 Registry Information Model. The stored procedures needed for the enhanced semantics is also
274 given in this chapter.
- 275 • Chapter 5 provides normative and informative references that are used within or relevant to this
276 document.

1.1 Terminology

277

278 The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT,
279 RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in IETF RFC
280 2119 [RFC211].

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

284 1.2 Conventions

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

- 287 • UML Diagrams

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

- 290 • Identifier Placeholders

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

296

297

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

298 1.3 Recommendations

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

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

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

308

2 OWL Overview

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

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

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

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

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

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

327 2.1 RDF Schema Features

- 328 • Class (Thing, Nothing)
- 329 • rdfs:subClassOf
- 330 • rdf:Property
- 331 • rdfs:subPropertyOf
- 332 • rdfs:domain
- 333 • rdfs:range
- 334 • Individual

335 2.2 (In)Equality

- 336 • equivalentClass
- 337 • equivalentProperty
- 338 • sameAs
- 339 • differentFrom
- 340 • AllDifferent
- 341 • distinctMembers

342 2.3 Property Characteristics

- 343 • ObjectProperty
- 344 • DatatypeProperty
- 345 • inverseOf
- 346 • TransitiveProperty
- 347 • SymmetricProperty
- 348 • FunctionalProperty

- 349 • InverseFunctionalProperty

350 2.4 Property Restrictions

- 351 • Restriction
- 352 • onProperty
- 353 • allValuesFrom
- 354 • someValuesFrom

355 2.5 Restricted Cardinality

- 356 • minCardinality (only 0 or 1)
- 357 • maxCardinality (only 0 or 1)
- 358 • cardinality (only 0 or 1)

359 2.6 Class Intersection

- 360 • intersectionOf

361 2.7 Versioning

- 362 • versionInfo
- 363 • priorVersion
- 364 • backwardCompatibleWith
- 365 • incompatibleWith
- 366 • DeprecatedClass
- 367 • DeprecatedProperty

368 2.8 Annotation Properties

- 369 • rdfs:label
- 370 • rdfs:comment
- 371 • rdfs:seeAlso
- 372 • rdfs:isDefinedBy
- 373 • AnnotationProperty
- 374 • OntologyProperty

375 2.9 Datatypes

- 376 • xsd datatypes

377

3 ebXML Registry Overview

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

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

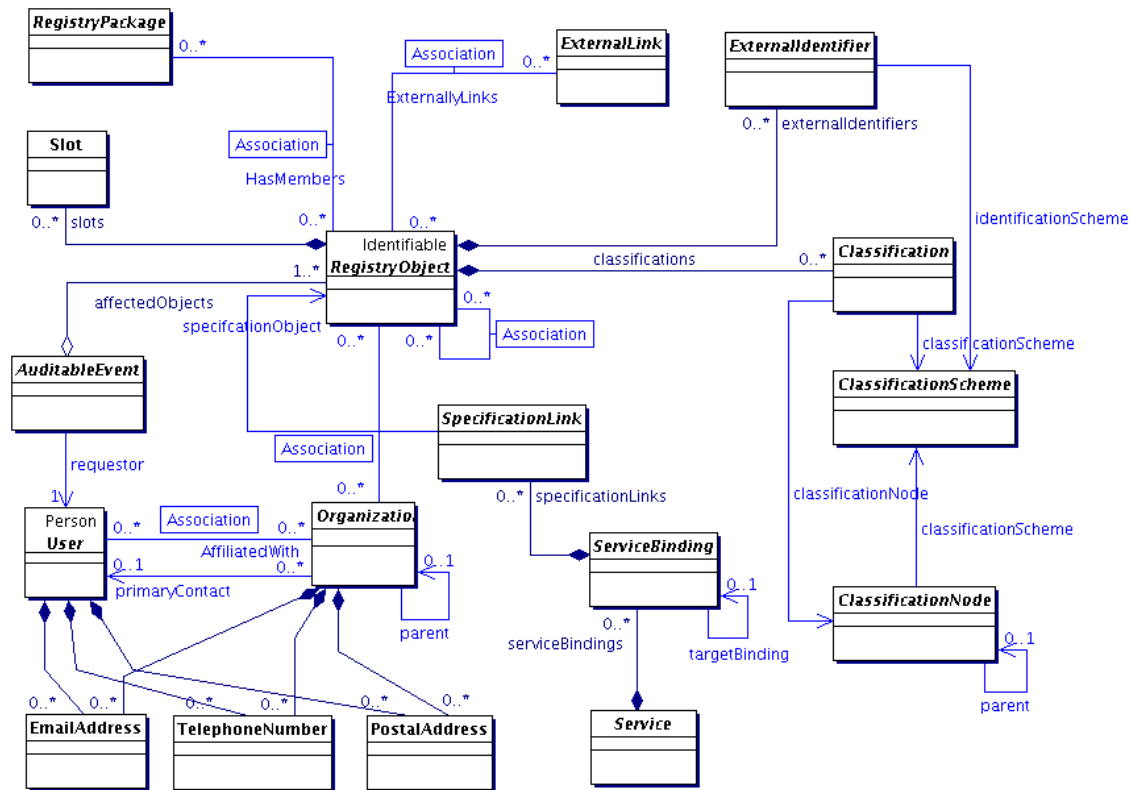
381 The information presented is informative and is not intended to replace the normative information defined
382 by ebXML Registry.

383 3.1 Overview of [ebRIM]

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

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

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

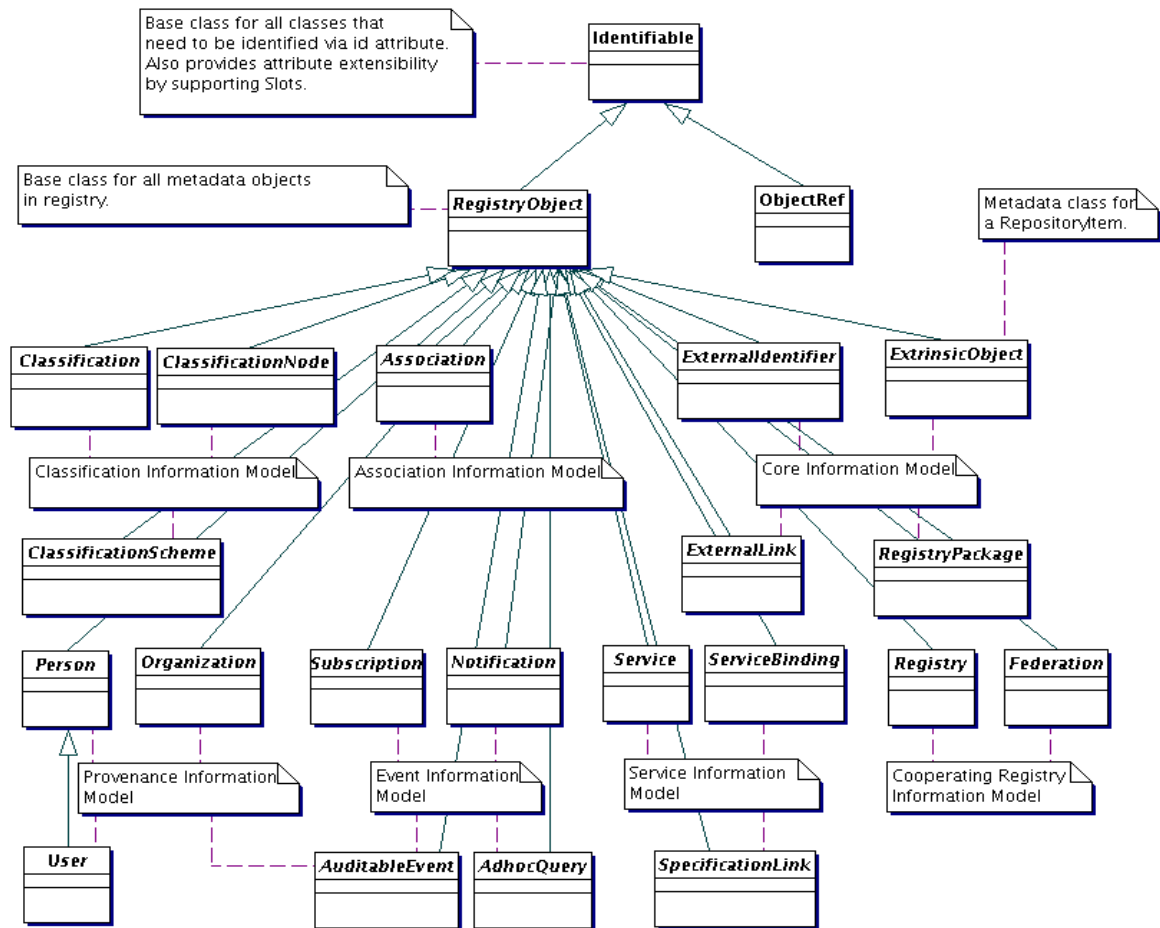


391 **Figure 1: ebXML Registry Information Model, High Level Public View**

392

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

397



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

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

401 3.1.1 RegistryObject

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

405 3.1.2 Object Identification

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

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

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

409 The id attribute value MAY potentially be human friendly.

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

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

412 Since a RegistryObject MAY have several versions, a logical id (called lid) is also defined which is unique
 413 for different logical objects. However the lid attribute value MUST be the same for all versions of the same
 414 logical object. The lid attribute value is a URN that, as well for id attribute, MAY potentially be human
 415 friendly.

417 friendly:

418

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

421

Listing 3: Example of lid Attribute

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

424

```
425 <rim:Organization id=${ACME_ORG_ID}  
426   lid="urn:acme:ACMEOrganization">  
427   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}  
428     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}  
429     value="ACME"/>  
430   </rim:ExternalIdentifier>  
431 </rim:Organization>
```

432

Listing 4: Example of ExternalIdentifier

435 3.1.3 Object Naming and Description

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

438

```
439 <rim:Organization id=${ACME_ORG_ID}  
440   lid="urn:acme:ACMEOrganization">  
441   <rim>Name>  
442     <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>  
443   </rim>Name>  
444   <rim>Description>  
445     <rim:LocalizedString value="ACME is a provider of Java software."  
446       xml:lang="en-US"/>  
447   </rim>Description>  
448   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}  
449     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}  
450     value="ACME"/>  
451   </rim:ExternalIdentifier>  
452 </rim:Organization>
```

453

Listing 5: Example of Name and Description

454

457 3.1.4 Object Attributes

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

460 3.1.4.1 Slot Attributes

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

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

467

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

470


```

471 <rim:Slot name="urn:acme:slot:NASDAQSymbol">
472   <rim:ValueList>
473     <rim:Value>ACME</rim:Value>
474   </rim:ValueList>
475 </rim:Slot>
476
477   <rim:Name>
478     <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
479   </rim:Name>
480   <rim:Description>
481     <rim:LocalizedString value="ACME makes Java. Provider of free Java
482 software."
483       xml:lang="en-US"/>
484   </rim:Description>
485   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
486     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
487     value="ACME"/>
488   </rim:ExternalIdentifier>
489 </rim:Organization>

```

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.

```

496
497 <rim:Organization id=${ACME_ORG_ID}
498   lid="urn:acme:ACMEOrganization">
499   <rim:Slot name="urn:acme:slot:NASDAQSymbol">
500     <rim:ValueList>
501       <rim:Value>ACME</rim:Value>
502     </rim:ValueList>
503   </rim:Slot>
504   <rim:Name>
505     <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
506   </rim:Name>
507   <rim:Description>
508     <rim:LocalizedString value="ACME makes Java. Provider of free Java
509 software." xml:lang="en-US"/>
510   </rim:Description>
511   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
512     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
513     value="ACME"/>
514   </rim:ExternalIdentifier>
515
516   <!--Classify Organization as a Software Publisher using NAICS Taxonomy-->
517   <rim:Classification id=${CLASSIFICATION_ID}
518     classificationNode=${NAICS_SOFTWARE_PUBLISHER_NODE_ID}
519     classifiedObject=${ACME_ORG_ID}>
520
521 </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.xml

533 [ebRIM] allows this scheme to be extensible.

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

537

```
538 <rim:Association  
539     id=${ASSOCIATION_ID}  
540     associationType=${CANONICAL_ASSOCIATION_TYPE_OFFERS_SERVICE_ID}  
541     sourceObject=${ACME_ORG_ID}  
542     targetObject=${ACME_SERVICE1_ID}/>
```

543

Listing 8: Example of Object Association

544 3.1.7 Object References To Web Content

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

550

```
551 <rim:ExternalLink externalURI="http://www.acme.com"  
552     id=${ACME_WEBSITE_EXTERNAL_ID}>  
553 <rim:Association  
554     id=${EXTERNALLYLINKS_ASSOCIATION_ID}  
555     associationType=${CANONICAL_ASSOCIATION_TYPE_EXTERNALLY_LINKS_ID}  
556     sourceObject=${ACME_WEBSITE_EXTERNAL_ID}  
557     targetObject=${ACME_ORG_ID}/>
```

558

Listing 9: Example of Reference to Web Content Using ExternalLink

559 3.1.8 Object Packaging

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

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

567

```
568 <rim:RegistryPackage  
569     id=${ACME_SERVICES_PACKAGE_ID}>  
570     <rim:RegistryObjectList>  
571         <rim:ObjectRef id=${ACME_SERVICE1_ID}>  
572             <rim:RegistryPackage  
573                 id=${ACME_PURCHASING_SERVICES_PACKAGE_ID}>  
574                 <rim:ObjectRef id=${ACME_PURCHASING_SERVICE1_ID}>  
575                 <rim:ObjectRef id=${ACME_PURCHASING_SERVICE2_ID}>  
576             </rim:RegistryPackage>  
577             <rim:RegistryPackage  
578                 id=${ACME_HR_SERVICES_PACKAGE_ID}>  
579                 <rim:ObjectRef id=${ACME_HR_SERVICE1_ID}>  
580                 <rim:ObjectRef id=${ACME_HR_SERVICE2_ID}>  
581             </rim:RegistryPackage>  
582         </rim:RegistryObjectList>  
583     </rim:RegistryPackage>
```

584

Listing 10: Example of Object Packaging Using RegistryPackages

585 3.1.9 ExtrinsicObject

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

590 3.1.10 Service Description

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

594 3.2 Overview of [ebRS]

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

4 Representing OWL Constructs in ebRIM and Providing Processing Support for Additional Semantics

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

The following ebRIM standard relational schema is used in coding the stored queries given in this section.

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

ebXML Registry Relations

Detailed explanation on how to represent some of the OWL Lite constructs in ebRIM is available from [Dogac, et. al.].

4.1 Representing RDF Schema Features in ebRIM

4.1.1 owl:Class → rim:ClassificationNode

An owl:Class MUST be mapped to a rim:ClassificationNode. For example, an OWL Class “City” which is a subclass of the Class “Country” can be mapped to ebRIM as follows: Two ClassificationNodes “City” and “Country” are defined where “City” is related to “Country” through the “parent” attribute of the ClassificationNode as shown in the following examples:

```
<owl:Class rdf:ID="City">
  <rdfs:subClassOf rdf:resource="#Country" />
</owl:Class>
```

Example owl:Class

```
<rim:ClassificationNode id='City' parent='Country' code='City'>
</rim:ClassificationNode>
```

Example Corresponding ebRIM construct ClassificationNode

4.1.2 rdf:Property → rim:Association Type Property

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

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

646

```
647 <rdf:Property rdf:ID="hasAirport">  
648   <rdfs:domain rdf:resource="#City"/>  
649   <rdfs:range rdf:resource="#AirPort"/>  
650 </rdf:Property>
```

651

Example rdf:Property

652

```
653 <rim:Association id='hasAirport' associationType='urn:oasis:names:tc:ebxml-  
654   regrep:AssociationType:Property'  
655   sourceObject= 'City'  
656   targetObject='Airport' >  
657 </rim:Association>
```

658

Example: ebRIM construct Association corresponding to rdf:Property

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

661 4.1.3 rdfs:subPropertyOf → rim:Association Type subPropertyOf

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

665

```
666 <rdf:Property rdf:ID="creditCardPayment">  
667   <rdfs:subPropertyOf rdf:Resource="#paymentMethods"/>  
668 </rdf:Property>
```

669

Example rdfs:subPropertyOf

670 A new ebXML RIM Association Type called "SubPropertyOf" MUST be defined to represent
671 rdfs:subPropertyOf in ebRIM. Such a semantic enhancement brings the following processing need: given
672 a property, it should be possible to retrieve all of its super properties as described in Section 6.1.

673 4.1.4 rdfs:subClassOf → rim:Association Type subclassOf

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

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

684

```
685 <owl:Class rdf:ID="AirReservationServices">  
686   <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-  
687     s/1.0/Profile.owl#Profile"/>  
688   <rdfs:subClassOf rdf:resource="#AirServices"/>  
689 </owl:Class>
```

690

Example rdfs:subClassOf

691 To express this semantics through ebXML RIM constructs, "AirReservationServices" ClassificationNode is
692 associated both with the "OWL-S Profile" and "AirServices" ClassificationNodes through the "targetObject"
693 and "sourceObject" attributes of the two instances of the newly created "subclassOf" ebXML Association

694 Type as shown in the following:

695

```
696 <rim:Association id='subClassOf1' associationType='urn:oasis:names:tc:ebxml-  
697   regrep:AssociationType:SubClassOf'  
698   sourceObject='AirReservationServices' targetObject='OWL-S Profile' >  
699 </rim:Association>  
700 <rim:Association id='subClassOf2' associationType='urn:oasis:names:tc:ebxml-  
701   regrep:AssociationType:SubClassOf'  
702   sourceObject='AirReservationServices' targetObject='AirServices' >  
703 </rim:Association>
```

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

708 4.1.5 owl:Individual → rim:ExtrinsicObject

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

712

```
713 <owl:Class rdf:ID="TravelWebService">  
714 </owl:Class>  
715  
716 <owl:ObjectProperty rdf:ID="paymentMethod">  
717   <rdfs:domain rdf:resource="#TravelWebService"/>  
718   <rdfs:range rdf:resource="#PossiblePaymentMethods"/>  
719 </owl:ObjectProperty >
```

720 Example owl:Class example

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

725

```
726 <TravelWebService rdf:ID="MyTravelWebService">  
727   <paymentMethod> Cash </paymentMethod>  
728 </TravelWebService>
```

729 Example owl:Individual example

730 In ebXML Registry the class instances can be stored in the Registry or in the Repository. However, since
731 ebXML philosophy is to store metadata in the Registry and the data (i.e., the instances) in the Repository,
732 it may be more appropriate to store class instances in the Repository and describe their metadata through
733 ExtrinsicObjects in the Registry.

734 4.2 Representing OWL (In)Equality Constructs in ebXML RIM

735 4.2.1 owl:equivalentClass, owl:equivalentProperty → rim:Association Type 736 EquivalentTo

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

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

744 4.2.2 owl:sameAs → rim:Association Type sameAs

745 ebXML Registry contains the metadata of the objects stored in the repository. In other words, the
746 instances are stored in repository and represented through "ExtrinsicObjects" in the registry.

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

749

```
750 <rdf:Description rdf:about="#MyAirReservationService">  
751   <owl:sameAs rdf:resource="#THYAirReservationService"/>  
752 </rdf:Description>
```

753 Example owl:sameAs

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

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

758

759 4.2.3 owl:differentFrom → rim:Association Type differentFrom

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

764

```
765 <rdf:Description rdf:about="#MyAirReservationService">  
766   <owl:differentFrom rdf:resource="#THYAirReservationService"/>  
767 </rdf:Description>
```

768 Example owl:differentFrom

769 This translates into declaring two "ExtrinsicObjects" in the ebXML registry to be different from each other.
770 For this purpose a new Association Type "differentFrom" MUST be defined in the ebXML registry to
771 explicitly indicate that the sourceRegistryObject is different from the targetRegistryObject. The adhoc
772 query presented in Section 6.9 can be used to process this semantics.

773 4.2.4 owl:AllDifferent

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

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

```
780 <owl:AllDifferent>  
781   <owl:distinctMembers rdf:parseType="Collection">  
782     <WebService rdf:about="#MyCarService"/>  
783     <WebService rdf:about="#MyFlightService"/>  
784     <WebService rdf:about="#MyHotelService"/>  
785   </owl:distinctMembers>  
786 </owl:AllDifferent>
```

787 Example owl:AllDifferentFrom

788 owl:AllDifferent SHOULD be represented in ebRIM as follows: the RegistryObjects under consideration
789 SHOULD be grouped as a RegistryPackage called "Collection". Then the RegistryObjects in the collection
790 MUST be associated with this RegistryPackage with "hasMember" Association Type. One slot of the
791 registry package MUST be used to indicate that all members are different.

792 **IMPORTANT NOTE:** When trying to submit the following "SubmitObjectsRequest", we get the following

793 unexpected error from the freebXML which implies that in the new Registry implementation it is not
794 possible to associate "slots" with RegistryPackages which seems there is a bug in the software.

795 javax.xml.bind.UnmarshalException: Unexpected element {urn:oasis:names:tc:ebxml-
796 regrep:xsd:rim:3.0}:Slot

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

798 4.3 Representing OWL Property Characteristics in ebRIM

799 4.3.1 owl:ObjectProperty → rim:Association Type objectProperty

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

803

```
804 <owl:ObjectProperty rdf:ID="hasAirport">  
805   <rdfs:domain rdf:resource="#City"/>  
806   <rdfs:range rdf:resource="#AirPort"/>  
807 </owl:ObjectProperty>
```

808

808 Example owl:ObjectProperty

809

```
810 <rim:Association id='hasAirport' associationType='urn:oasis:names:tc:ebxml-  
811 regrep:AssociationType:ObjectProperty'  
812   sourceObject= 'City' targetObject='Airport' >  
813 </rim:Association>
```

814

814 Example Corresponding ebRIM construct Association

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

818 4.3.2 owl:DatatypeProperty → rim:Association Type DatatypeProperty

819 Similarly, to represent OWL DatatypeProperty in ebXML, a new Association Type called
820 "DatatypeProperty" MUST be defined. Consider the following example which defines an datatype property
821 "hasPrice" whose domain is the "AirReservationServices" and whose range is "XMLSchema
822 nonNegativeInteger". How OWL XML Schema types are handled in ebXML RIM is described in Section
823 4.9.

```
824 <owl:DatatypeProperty rdf:ID="hasPrice">  
825   <rdfs:subpropertyOf rdf:resource="http://www.daml.org/services/daml-  
826 s/2001/05/Profile.owl"/>  
827   <rdfs:domain rdf:resource="#AirReservationServices"/>  
828   <rdfs:range  
829   rdf:resource="http://www.w3.org/2000/10/XMLSchema/nonNegativeInteger"/>  
830 </owl:DatatypeProperty>
```

831

831 Example owl:DatatypeProperty

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

834 4.3.3 owl:TransitiveProperty → rim:Association Type transitiveProperty

835 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)
836 [McGuinness, Harmelen]. Transitive property is a subproperty of ObjectProperty and MUST be defined as
837 a new Association Type called "transitiveProperty" in ebRIM.

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

840


```

841 <owl:ObjectProperty rdf:ID="succeeds">
842   <rdf:type rdf:resource="&owl;TransitiveProperty" />
843   <rdfs:domain rdf:resource="#TravelWebService" />
844   <rdfs:range rdf:resource="#TravelWebService" />
845 </owl:ObjectProperty>

```

846 **Example owl:TransitiveProperty**

847 Assume the following two definitions which declare three Web service instances from TravelWebService
848 class where "MyHotelAvailabilityService" service succeeds "MyAirReservationService" and
849 "MyInsuranceService" succeeds MyHotelAvailabilityService". Since "succeeds" is a transitive property, it
850 follows that "MyInsuranceService" succeeds "MyAirReservationService" although this fact is not explicitly
851 stated.

852

```

853 <TravelWebService rdf:ID="MyHotelAvailabilityService">
854   <succeeds rdf:resource="#MyAirReservationService" />
855 </TravelWebService>
856
857 <TravelWebService rdf:ID="MyInsuranceService">
858   <succeeds rdf:resource="#MyHotelAvailabilityService" />
859 </TravelWebService>

```

860 **Example owl:TransitiveProperty instances**

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

864 [4.3.4 owl:inverseOf → rim:Association Type inverseOf](#)

865 In OWL, one property may be stated to be the inverse of another property. If the property P1 is stated to
866 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
867 P1 property [McGuinness, Harmelen].

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

871

```

872 <owl:ObjectProperty rdf:ID="precedes">
873   <owl:inverseOf rdf:resource="#succeeds" />
874 </owl:ObjectProperty>

```

875 **Example owl:inverseOf Property**

876 Assume that we want to find all the Web services which can succeed a given Web service. In such a
877 case, we need not only find all the Web services which succeeds this given Web service, that is the target
878 objects of "succeeds" Association instance, but we also need to find all the sourceObjects of the
879 "precedes" Association instance since "precedes" is declared to be the "inverseOf" succeeds Association
880 instance. This can be achieved through the adhoc query presented in Section 6.19.

881 [4.3.5 owl:SymmetricProperty → rim:Association Type SymmetricProperty](#)

882 In OWL, if a property is symmetric, then if the pair (x,y) is an instance of the symmetric property P, then
883 the pair (y,x) is also an instance of P [McGuinness, Harmelen]. Symmetric property is a subproperty of
884 ObjectProperty in OWL. Consider the OWL class "WebService" and the "complements" symmetric
885 property:

```

886 <owl:Class rdf:ID="WebService">
887   <rdfs:subClassOf
888     rdf:resource="http://www.w3.org/2000/01/rdfschema#Resource"/>
889 </owl:Class>
890 <owl:SymmetricProperty rdf:ID="complements">
891   <rdfs:domain rdf:resource="#WebService"/>
892   <rdfs:range rdf:resource="#WebService"/>

```

```
893 </owl:SymmetricProperty>
```

894 **Example owl:SymmetricProperty**

895 Given that HotelReservationWebService complements AirReservationWebService, it is possible to
896 deduce that AirReservationWebService complements HotelReservationWebService.

897 owl:SymmetricProperty MUST be defined as a new type of Association in ebRIM called
898 "SymmetricProperty". Furthermore the adhoc query presented in Section 6.20 MUST be available in the
899 Registry to retrieve symmetric Associations of a ClassificationNode.

900 **4.3.6 owl:FunctionalProperty → rim:Association Type FunctionalProperty**

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

```
905 <owl:DatatypeProperty rdf:ID="hasPrice">  
906   <rdf:type rdf:resource="&owl;FunctionalProperty" />  
907   <rdfs:domain rdf:resource="#AirReservationServices"/>  
908   <rdfs:range  
909   rdf:resource="http://www.w3.org/2000/10/XMLSchema/nonNegativeInteger"/>  
910 </owl:DatatypeProperty>
```

911 **Example owl:FunctionalProperty**

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

915 **4.3.7 owl:InverseFunctionalProperty → rim:Association Type 916 InverseFunctionalProperty**

917 In OWL, if a property is inverse functional then the inverse of the property is functional. Thus the inverse
918 of the property has at most one value for each individual [McGuinness, Harmelen].

919 As an example, the ObjectProperty "departsFrom" indicates that each flight originates from only one
920 airport.

```
921 <owl:ObjectProperty rdf:ID="departsFrom">  
922   <rdf:type rdf:resource="&owl;InverseFunctionalProperty" />  
923   <rdfs:domain rdf:resource="#Airport"/>  
924   <rdfs:range rdf:resource="#Airport"/>  
925 </owl:ObjectProperty>
```

926 **Example owl:InverseFunctionalProperty**

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

930 **4.4 OWL Property Restrictions in ebXML RIM**

931 An important construct of OWL is "owl:Restriction". In RDF, a property has a global scope, that is, no
932 matter what class the property is applied to, the range of the property is the same. "owl:Restriction", on the
933 other hand, has a local scope; restriction is applied on the property within the scope of the class where it is
934 defined. The aim is to make ontologies more extendable and hence more reusable.

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

```
939  
940 <owl:ObjectProperty rdf:ID="paymentMethod">  
941   <rdfs:domain rdf:resource="#TravelWebService"/>
```

```
942 <rdfs:range rdf:resource="#PossiblePaymentMethods"/>
943 </owl:ObjectProperty >
```

944 **Example owl:ObjectProperty "paymentMethod"**

```
945
946 <owl:Class rdf:ID="AirReservationServices">
947   <rdfs:subClassOf>
948     <owl:Restriction>
949       <owl:onProperty rdf:resource="#paymentMethod"/>
950       <owl:allValuesFrom rdf:resource= "#CreditCard"/>
951     </owl:Restriction>
952   </rdfs:subClassOf>
953 </owl:Class>
```

954 **Example owl:Restriction on ObjectProperty "paymentMethod"**

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

```
958
959 <owl:ObjectProperty rdf:ID="paymentMethodCC">
960   <rdfs:domain rdf:resource="#AirReservationServices"/>
961   <rdfs:range rdf:resource="#CreditCard"/>
962 </owl:ObjectProperty >
```

963 **Example owl:ObjectProperty "paymentMethodCC"**

964 We believe that defining a generic Association Type and and keeping track of its various restrictions in
965 relational tables will bring considerable overhead to the system. Since an Association Type can always be
966 defined in ebXML between any ReristryObjects, we also think that the expressive power is already there.

967 4.5 Representing OWL Restricted Cardinality in ebXML RIM

968 4.5.1 owl:minCardinality (only 0 or 1)

969 In OWL, cardinality is stated on a property with respect to a particular class. If a minCardinality of 1 is
970 stated on a property with respect to a class, then any instance of that class will be related to at least one
971 individual by that property. This restriction is another way of saying that the property is required to have a
972 value for all instances of the class. In OWL Lite, the only minimum cardinalities allowed are 0 or 1. A
973 minimum cardinality of zero on a property just states (in the absence of any more specific information)
974 that the property is optional with respect to a class [McGuinness, Harmelen].

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

```
977 <owl:Class rdf:ID="WebService">
978   <rdfs:subClassOf>
979     <owl:Restriction>
980       <owl:onProperty rdf:resource="#hasPrice"/>
981       <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">
982 1 </owl:minCardinality>
983     </owl:Restriction>
984   </rdfs:subClassOf>
985 </owl:Class>
```

986 **Example owl:minCardinality**

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

```
989
990 <rim:Association id = "hasPriceMinCardinalityRestriction"
991 associationType = "urn:oasis:names:tc:ebxml-
992 regrep:AssociationType:ObjectProperty" sourceObject = "WebService"
993 targetObject = "Price">
994   <rim:Name>
```

```

995         <rim:LocalizedString value = 'hasPrice' />
996     </rim:Name>
997     <rim:Slot name="minCardinality">
998         <rim:ValueList>
999             <rim:Value>1</rim:Value>
1000         </rim:ValueList>
1001     </rim:Slot>
1002 </rim:Association>

```

1003 Example Representing owl:minCardinality in ebRIM

1004 4.5.2 owl:maxCardinality (only 0 or 1)

1005 In OWL, cardinality is stated on a property with respect to a particular class. If a maxCardinality of 1 is
1006 stated on a property with respect to a class, then any instance of that class will be related to at most one
1007 individual by that property. A maxCardinality 1 restriction is sometimes called a functional or unique
1008 property. It may be useful to state that certain classes have no values for a particular property. This
1009 situation is represented by a maximum cardinality of zero on the property [McGuinness, Harmelen].

1010 Consider for example the following OWL code which states that each instance of a “WebService” class
1011 can have at most one price:

```

1012 <owl:Class rdf:ID="WebService">
1013     <rdfs:subClassOf>
1014         <owl:Restriction>
1015             <owl:onProperty rdf:resource="#hasPrice"/>
1016             <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
1017 1 </owl:maxCardinality>
1018         </owl:Restriction>
1019     </rdfs:subClassOf>
1020 </owl:Class>

```

1021 Example owl:maxCardinality

1022 In ebXML RIM, cardinalities of Association Types MUST be defined by associating a maxCardinality slot
1023 with the Association Types as shown in the following example:

```

1024
1025 <rim:Association id = "hasPriceMaxCardinalityRestriction"
1026 associationType = "urn:oasis:names:tc:ebxml-
1027 regrep:AssociationType:ObjectProperty" sourceObject = "WebService"
1028 targetObject = "Price">
1029     <rim:Name>
1030         <rim:LocalizedString value = 'hasPrice' />
1031     </rim:Name>
1032     <rim:Slot name="maxCardinality">
1033         <rim:ValueList>
1034             <rim:Value>1</rim:Value>
1035         </rim:ValueList>
1036     </rim:Slot>
1037 </rim:Association>

```

1038 Example Representing owl:maxCardinality in ebRIM

1039 4.5.3 owl:cardinality

1040 In OWL, cardinality is provided as a convenience when it is useful to state that a property on a class has
1041 both minCardinality 0 and maxCardinality 0 or both minCardinality 1 and maxCardinality 1 [McGuinness,
1042 Harmelen].

1043 Consider for example the following OWL code which states that each instance of a “WebService” class
1044 must have exactly one price:

```

1045 <owl:Class rdf:ID="WebService">
1046     <rdfs:subClassOf>
1047         <owl:Restriction>
1048             <owl:onProperty rdf:resource="#hasPrice"/>

```

```

1049         <owl:Cardinality rdf:datatype="&xsd;nonNegativeInteger"> 1
1050     </owl:Cardinality>
1051     </owl:Restriction>
1052 </rdfs:subClassOf>
1053 </owl:Class>

```

1054 **Example owl:Cardinality**

1055 In ebXML RIM, cardinalities of Association Types MUST be defined by associating a Cardinality slot with
 1056 the Association Types as shown in the following example:

```

1057
1058 <rim:Association id = "hasPriceCardinalityRestriction"
1059 associationType = "urn:oasis:names:tc:ebxml-
1060 regrep:AssociationType:ObjectProperty" sourceObject = "WebService"
1061 targetObject = "Price">
1062     <rim:Name>
1063         <rim:LocalizedString value = 'hasPrice' />
1064     </rim:Name>
1065     <rim:Slot name="cardinality">
1066         <rim:ValueList>
1067             <rim:Value>1</rim:Value>
1068         </rim:ValueList>
1069     </rim:Slot>
1070 </rim:Association>

```

1071 **Example Representing owl:Cardinality in ebRIM**

1072 **4.6 Representing OWL Class Intersection in ebXML RIM**

1073 OWL provides the means to manipulate class extensions using basic set operators. In OWL Lite, only
 1074 "owl:intersectionOf" is available which defines a class that consists of exactly all objects that belong to
 1075 both of the classes. In the following example, "AirReservationServices" is defined as the intersection of
 1076 "AirServices" and "ReservationServices":

```

1077
1078 <owl:Class rdf:ID="AirReservationServices">
1079     <owl:intersectionOf rdf:parseType="Collection">
1080         <owl:Class rdf:about="#AirServices" />
1081         <owl:Class rdf:about="#ReservationServices" />
1082     </owl:intersectionOf>
1083 </owl:Class>

```

1084 **Example owl:intersectionOf**

1085 In ebXML RIM "owl:intersectionOf" set operator MUST be represented as follows:

- 1086 • A new Association Type called "intersectionOf" MUST be created.
- 1087 • A new ClassificationNode to denote the intersection of the classes MUST be created. For the
 1088 example, this could be "AirReservationServices" ClassificationNode.
- 1089 • Each of the intersected classes MUST be represented as members of a new RegistryPackage.
 1090 For the example, the RegistryPackage should contain "AirServices" and the
 1091 "RegistrationServices".
- 1092 • The new ClassificationNode denoting the intersection MUST be assigned as the sourceObject of
 1093 the "intersectionOf" association. For the example, "AirReservationServices" must be the the
 1094 sourceObject of the "intersectionOf" association.
- 1095 • The target class of the "intersectionOf" association MUST be set to the newly created
 1096 RegistryPackage. For the example given above, the RegistryPackage containing "AirServices"
 1097 and the "RegistrationServices" should be the target class of the "intersectionOf" association.

```

1098
1099 <rim:ClassificationNode id = "AirReservationServices" parent= "Service"
1100 code = "AirReservationServices">
1101     <rim:Name>
1102         <rim:LocalizedString value = "AirReservationServices" />

```

```

1103     </rim:Name>
1104 </rim:ClassificationNode>
1105
1106
1107 <rim:RegistryPackage id = "IntersectionOfRegistryPackage" >
1108     <rim:Name>
1109         <rim:LocalizedString value =
1110 "IntersectionOfRegistryPackage"/>
1111     </rim:Name>
1112 </rim:RegistryPackage>
1113
1114 <rim:Association id = "HasMemberRegistryPackageAssoc1"
1115 associationType = "urn:oasis:names:tc:ebxml-
1116 regrep:AssociationType:HasMember" sourceObject =
1117 "IntersectionOfRegistryPackage"
1118 targetObject = "AirServices" />
1119
1120 <rim:Association id = "HasMemberRegistryPackageAssoc2"
1121 associationType = "urn:oasis:names:tc:ebxml-
1122 regrep:AssociationType:HasMember" sourceObject =
1123 "IntersectionOfRegistryPackage"
1124 targetObject = "ReservationServices" />
1125
1126 <rim:Association id = "IntersectionOfRegistryPackageAssoc"
1127 associationType = "urn:oasis:names:tc:ebxml-
1128 regrep:AssociationType:IntersectionOf" sourceObject =
1129 "AirReservationServices"
1130 targetObject = " IntersectionOfRegistryPackage " />
1131

```

1132 **Example Defining Intersection of ClassificationNodes in ebRIM**

1133 When such a representation is used to create a complex class (a new ClassificationNode) in RIM, it
1134 becomes possible to infer that the objects (instances) classified by both of the classes
1135 (ClassificationNodes) constituting the intersection are also the instances of this complex class. The adhoc
1136 query presented in Section 6.23 MUST be available in the ebXML Registry to retrieve the direct instances
1137 of the complex class and also the instances of the intersection of the classes.

1138 **4.7 Representing OWL Versioning in ebXML RIM**

1139 **4.7.1 owl:versionInfo, owl:priorVersion**

1140 An owl:versionInfo statement generally has as its object a string giving information about this version, for
1141 example RCS/ CVS keywords. This statement does not contribute to the logical meaning of the ontology
1142 other than that given by the RDF(S) model theory [McGuinness, Harmelen].

1143 An owl:priorVersion statement contains a reference to another ontology. This identifies the specified
1144 ontology as a prior version of the containing ontology [McGuinness, Harmelen].

1145 In ebXML, since a RegistryObject MAY have several versions, a logical id (called lid) is also defined which
1146 is unique for different logical objects. However the lid attribute value MUST be the same for all versions of
1147 the same logical object. Therefore, almost all the underlying ebXML relational tables keep version
1148 information through "versionName" and "comment_" attributes.

1149 "owl:version" information MUST be stored in the "versionName" and "comment_" attributes of the table
1150 ClassScheme in the Registry.

1151 **4.8 Representing OWL Annotation Properties in ebXML RIM**

1152 **4.8.1 rdfs:label**

1153 rdfs:label is an instance of rdf:Property that may be used to provide a human-readable version of a
1154 resource's name [Brickley, Guha].

1155 In ebXML RIM, human readable names of resources are provided through rim:Name. rdfs:label MUST be

1156 expressed through rim:Name.

1157

```
1158 <owl:Class rdf:ID="AirReservationServices">
1159   <rdfs:label>Air Reservation Services</rdfs:label>
1160 </owl:Class>
```

1161 Example rdfs:label

1162

```
1163 <rim:ClassificationNode id = 'AirReservationServices' parent=
1164 'TravelServices' code = 'AirReservationServices'>
1165   <rim:Name>
1166     <rim:LocalizedString value = 'Air Reservation Services' />
1167   </rim:Name>
1168 </rim:ClassificationNode>
```

1169 Example rim:Name

1170 4.8.2 rdfs:comment

1171 rdfs:comment is an instance of rdf:Property that may be used to provide a human-readable description of
1172 a resource [Brickley, Guha].

1173 In ebXML RIM, this construct MUST be expressed through rim:Description.

1174

```
1175 <owl:Class rdf:ID="AirReservationServices">
1176   <rdfs:comment>Open Travel Alliance Air Reservation Services
1177   </rdfs:comment>
1178 </owl:Class>
```

1179 Example rdfs:comment

1180

```
1181 <rim:ClassificationNode id = 'AirReservationServices' parent=
1182 'TravelServices' code = 'AirReservationServices'>
1183   <rim:Description>
1184     <rim:LocalizedString value = 'Open Travel Alliance Air
1185 Reservation Services' />
1186   </rim:Description>
1187 </rim:ClassificationNode>
```

1188 Example: rim:Description

1189 4.8.3 rdfs:seeAlso

1190 rdfs:seeAlso is an instance of rdf:Property that is used to indicate a resource that might provide additional
1191 information about the subject resource [Brickley, Guha].

1192 This construct MUST be expressed in ebXML RIM by defining an ExternalLink, called,
1193 "seeAlsoExternalLink".

1194

```
1195 <owl:Class rdf:ID="AirReservationServices">
1196   <rdfs:seeAlso rdf:resource="http://www.opentravel.org" />
1197 </owl:Class>
```

1198 Example rdfs:seeAlso

```
1199 <rim:ClassificationNode id = 'AirReservationServices' parent=
1200 'TravelServices' code = 'AirReservationServices'>
1201 </rim:ClassificationNode>
1202
1203 <rim:ExternalLink id = "seeAlsoExternalLink"
1204   externalURI= "http://www.opentravel.org" >
1205 </rim:ExternalLink>
```

1206

```
1207 <rim:Association id = 'seeAlsoAssociation'  
1208     associationType = 'urn:oasis:names:tc:ebxml-  
1209     regrep:AssociationType:ExternallyLinks'  
1210     sourceObject = 'AirReservationServices'  
1211     targetObject = 'seeAlsoExternalLink' />
```

1212 **Example rim:seeAlsoExternalLink**

1213 4.9 OWL Datatypes in ebXML RIM

1214 OWL allows the use of XML Schema datatypes to describe part of the datatype domain by simply
1215 including their URIs within an OWL ontology [McGuinness, Harmelen]. In ebXML, XML Schema datatypes
1216 SHOULD be used by providing an external link from the registry.

1217 The following example demonstrates how XML Schema datatype “integer” can be referenced through an
1218 ExternalLink called 'integer' and how to define a DatatypeProperty, namely, “hasPrice”, whose target
1219 object is the defined to be ExternalLink 'integer':

1220

```
1221 <rim:ExternalLink id = "integer"  
1222     externalURI="http://www.w3.org/2001/XMLSchema#integer" >  
1223     <rim:Name> <rim:LocalizedString value = "XML Schema integer"/>  
1224     </rim:Name>  
1225 </rim:ExternalLink>  
1226 <rim:Association id = 'hasPrice' associationType = 'urn:oasis:names:tc:ebxml-  
1227     regrep:AssociationType:DatatypeProperty'  
1228     sourceObject = 'AirReservationServices'  
1229     targetObject = 'integer' >  
1230     <rim:Name> <rim:LocalizedString value ="hasPrice"/></rim:Name>  
1231 </rim:Association>
```

1232

1233 **Example Corresponding ebRIM construct Association**

1234 5 Cataloging Service Profile

1235 The ebXML Registry provides the ability for a content cataloging service to be configured for any type of
1236 content. The cataloging service serves the following purposes:

- 1237 • Automates the mapping from the source information model (in this case OWL) to ebRIM. This
1238 hides the complexity of the mapping from the OWL publisher and eliminates the need for any
1239 special UI tools to be provided by the registry implementor for publishing OWL documents.
- 1240 • Selectively converts content into ebRIM compatible metadata when the content is cataloged after
1241 being published. The generated metadata enables the selected content to be used as
1242 parameter(s) in content specific parameterized queries.

1243 This section describes the cataloging service for cataloging OWL content.

1244 An OWL document, when published to an ebXML Registry implementing the OWL Profile, **MUST** be
1245 cataloged as specified in this section using a OWL Content Cataloging Service as defined by [ebRS].

1246 5.1 Invocation Control File

1247 The OWL cataloging service **MAY** optionally support an invocation control file that declaratively specifies
1248 the transforms necessary to catalog published OWL documents.

1249 5.2 Input Metadata

1250 The OWL cataloging service **MUST** be pre-configured to be automatically invoked when the following
1251 types of metadata are published, as defined by the [ebRS] specifications.

1252 These are the only types of metadata that **MAY** describe a OWL document being published:

- 1253 • An `ExtrinsicObject` whose `ObjectType` references the canonical OWL `ClassificationNode`
1254 specified in Section 7. The `ExtrinsicObject` **MUST** have an OWL document as its `RepositoryItem`.
- 1255 • An `ExternalLink` whose `ObjectType` references the canonical OWL `ClassificationNode` specified in
1256 Section 7. In case of `ExternalLink` the OWL document **MUST** be resolvable via a URL described
1257 by the value of the `externalURI` attribute of the `ExternalLink`. Recall that, in the `ExternalLink` case
1258 the OWL document is not be stored in the repository.

1259

```
1260 <rim:ExtrinsicObject id="urn:acmeinc:ebxml:registry:3.0:owl">  
1261 ...  
1262 </rim:ExtrinsicObject>
```

1263 Example of ExtrinsicObject Input Metadata

1264

```
1265 <rim:ExternalLink  
1266 id="urn:acmeinc:ebxml:registry:3.0:owl"  
1267 externalURI="http://www.acme.com/owl/ebXMLRegistryService.owl"  
1268 >  
1269 ...  
1270 </rim:ExternalLink>
```

1271 Example of ExternalLink Input Metadata

1272 5.3 Input Content

1273 The OWL cataloging service expects an OWL document as its input content. The input content **MUST** be
1274 processed by the OWL cataloging service regardless of whether it is a `RepositoryItem` for an
1275 `ExtrinsicObject` or whether it is content external to repository that is referenced by an `ExternalLink`.

1276

1277 5.4 Output Metadata

1278 This section describes the metadata produced by the OWL cataloging service produces as output.

1279 5.4.1 owl:Class → rim:ClassificationNode

1280 The OWL Cataloging service MUST automatically produce a rim:ClassificationNode instance for each
1281 owl:class element within the input OWL or its imports, as specified in the owl:Class →
1282 rim:ClassificationNode mapping earlier in this document.

1283 5.4.2 rdf:Property → rim:Association Type Property

1284 The OWL Cataloging service MUST automatically produce an rim:Association instance with
1285 associationType Property for each rdf:Property element within the input OWL or its imports, as specified
1286 in the rdf:Property → rim:Association Type Property mapping earlier in this document.

1287 5.4.3 rdfs:subPropertyOf → rim:Association Type subPropertyOf

1288 The OWL Cataloging service MUST automatically produce an rim:Association instance with
1289 associationType subPropertyOf for each rdfs:subPropertyOf element within the input OWL or its imports,
1290 as specified in the rdfs:subPropertyOf → rim:Association Type subPropertyOf mapping earlier in this
1291 document.

1292 5.4.4 rdfs:subClassOf → rim:Association Type subClassOf

1293 The OWL Cataloging service MUST automatically produce an rim:Association instance with
1294 associationType subClassOf for each rdfs:subClassOf element within the input OWL or its imports, as
1295 specified in the rdfs:subClassOf → rim:Association Type subClassOf mapping earlier in this document.

1296 5.4.5 owl:Individual → rim:ExtrinsicObject

1297 The OWL Cataloging service MUST automatically produce rim:ExtrinsicObject instances for each
1298 owl:Individual element within the input OWL or its imports, as specified in the owl:Individual →
1299 rim:ExtrinsicObject mapping earlier in this document.

1300 5.4.6 owl:equivalentClass, owl:equivalentProperty → rim:Association Type 1301 EquivalentTo

1302 The OWL Cataloging service MUST automatically produce rim:Association instances with
1303 associationType EquivalentTo for each owl:equivalentClass or owl:equivalentProperty element within the
1304 input OWL or its imports, as specified in the owl:equivalentClass, owl:equivalentProperty →
1305 rim:Association Type EquivalentTo mapping earlier in this document.

1306 5.4.7 owl:sameAs → rim:Association Type sameAs

1307 The OWL Cataloging service MUST automatically produce rim:Association instances with
1308 associationType sameAs for each owl:sameAs element within the input OWL or its imports, as specified
1309 in the owl:sameAs → rim:Association Type sameAs mapping earlier in this document.

1310 5.4.8 owl:differentFrom → rim:Association Type differentFrom

1311 The OWL Cataloging service MUST automatically produce rim:Association instances with
1312 associationType differentFrom for each owl:differentFrom element within the input OWL or its imports, as
1313 specified in the owl:differentFrom → rim:Association Type differentFrom mapping earlier in this document.

1314 5.4.9 owl:AllDifferent → rim:RegistryPackage

1315 The OWL Cataloging service MUST automatically produce rim:RegistryPackage instances for each
1316 owl:AllDifferent element within the input OWL or its imports, as specified in the owl:AllDifferent →

1317 rim:RegistryPackage mapping earlier in this document.

1318 **5.4.10 owl:ObjectProperty → rim:Association Type objectProperty**

1319 The OWL Cataloging service MUST automatically produce rim:Association instances with
1320 associationType objectProperty for each owl:ObjectProperty element within the input OWL or its imports,
1321 as specified in the owl:ObjectProperty → rim:Association Type objectProperty mapping earlier in this
1322 document.

1323 **5.4.11 owl:DatatypeProperty → rim:Association Type DatatypeProperty**

1324 The OWL Cataloging service MUST automatically produce rim:Association instances with
1325 associationType datatypeProperty for each owl:DatatypeProperty element within the input OWL or its
1326 imports, as specified in the owl:DatatypeProperty → rim:Association Type datatypeProperty mapping
1327 earlier in this document.

1328 **5.4.12 owl:TransitiveProperty → rim:Association Type transitiveProperty**

1329 The OWL Cataloging service MUST automatically produce rim:Association instances with
1330 associationType transitiveProperty for each owl:TransitiveProperty element within the input OWL or its
1331 imports, as specified in the owl:TransitiveProperty → rim:Association Type transitiveProperty mapping
1332 earlier in this document.

1333 **5.4.13 owl:inverseOf → rim:Association Type inverseOf**

1334 The OWL Cataloging service MUST automatically produce rim:Association instances with
1335 associationType inverseOf for each owl:inverseOf element within the input OWL or its imports, as
1336 specified in the owl:inverseOf → rim:Association Type inverseOf mapping earlier in this document.

1337 **5.4.14 owl:SymmetricProperty → rim:Association Type SymetricProperty**

1338 The OWL Cataloging service MUST automatically produce rim:Association instances with
1339 associationType SymetricProperty for each owl:SymetricProperty element within the input OWL or its
1340 imports, as specified in the owl:SymetricProperty → rim:Association Type SymetricProperty mapping
1341 earlier in this document.

1342 **5.4.15 owl:FunctionalProperty → rim:Association Type FunctionalProperty**

1343 The OWL Cataloging service MUST automatically produce rim:Association instances with
1344 associationType FunctionalProperty for each owl:FunctionalProperty element within the input OWL or its
1345 imports, as specified in the owl:FunctionalProperty → rim:Association Type FunctionalProperty mapping
1346 earlier in this document.

1347 **5.4.16 owl:InverseFunctionalProperty → rim:Association Type 1348 InverseFunctionalProperty**

1349 The OWL Cataloging service MUST automatically produce rim:Association instances with
1350 associationType InverseFunctionalProperty for each owl:InverseFunctionalProperty element within the
1351 input OWL or its imports, as specified in the owl:InverseFunctionalProperty → rim:Association Type
1352 InverseFunctionalProperty mapping earlier in this document.

1353 **5.4.17 owl:minCardinality (only 0 or 1)**

1354 The OWL Cataloging service MUST automatically add a slot with name minCardinality to the relevant
1355 rim:Association instances for each owl:minCardinality element within the input OWL or its imports, as
1356 specified in section 4.5.1 where how to represent owl:minCardinality is described.

1357 **5.4.18 owl:maxCardinality (only 0 or 1)**

1358 The OWL Cataloging service MUST automatically add a slot with name maxCardinality to the relevant
1359 rim:Association instances for each owl:maxCardinality element within the input OWL or its imports, as
1360 specified in section 4.5.2 where how to represent owl:maxCardinality is described.

1361 **5.4.19 owl:cardinality**

1362 The OWL Cataloging service MUST automatically add a slot with name cardinality to the relevant
1363 rim:Association instances for each owl:cardinality element within the input OWL or its imports, as
1364 specified in section 4.5.3 where how to represent owl:cardinality is described.

1365 **5.4.20 owl:intersectionOf**

1366 The OWL Cataloging service MUST automatically produce a rim:RegistryPackage and a rim:Association
1367 instances with type IntersectionOf for each owl:intersectionOf element within the input OWL or its imports,
1368 as specified in section 4.6 where how to represent owl:intersectionOf is described.

1369 **5.4.21 rdfs:label**

1370 The OWL Cataloging service MUST automatically produce a rim:Name instance for each rdfs:label
1371 element within the input OWL or its imports, as specified in section 4.8.1 where how to represent
1372 rdfs:label is described.

1373 **5.4.22 rdfs:comment**

1374 The OWL Cataloging service MUST automatically produce a rim:Description instance for each
1375 rdfs:comment element within the input OWL or its imports, as specified in section 4.8.2 where how to
1376 represent rdfs:comment is described.

1377 **5.4.23 rdfs:seeAlso**

1378 The OWL Cataloging service MUST automatically produce a rim:ExternalLink and a rim:Association with
1379 type ExternallyLinks instances for each rdfs:seeAlso element within the input OWL or its imports, as
1380 specified in section 4.8.3 where how to represent rdfs:seeAlso is described.

1381

6 Discovery Profile

1382 The ebXML Registry provides the ability for a user defined parameterized queries to be configured for
1383 each type of content. The queries may be as complex or simple as the discovery use case requires. The
1384 complexity of the parameterized queries may hidden from the registry client by storing them within the
1385 ebXML Registry as instances of the AdhocQuery class, and being invoked by simply providing their
1386 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].

1388 An ebXML Registry SHOULD provide a graphical user interface that displays any configured
1389 parameterized query as a form which contains an appropriate field for entering each query parameter.

1390 This chapter defines the queries that MUST be support by an ebXML Registry implementing the OWL
1391 Profile for processing the semantics provided in the OWL content. An implementation MAY also support
1392 additional discovery queries for OWL content, some of which have already identified in this section.

1393 The queries defined in this chapter are parameterized queries stored in the Registry as instances of the
1394 AdhocQuery type, in the same manner as any other RegistryObject.

1395 In the subsequent section each query is described simply in terms of its supported parameters that serve
1396 as its search criteria. The actual AdhocQuery instances are much more complex in comparison but they
1397 are not exposed to the client making the query. Details on these queries are specified canonically in
1398 section 7.3 .

1399 Some of the queries that are necessary to process the semantics involved in OWL documents requires
1400 SQL recursion mechanism. Since SQL 92, does not support recursion mechanism, those queries are
1401 stated to be implemented optionally. Additionally for these types of discovery queries, the “stored
1402 procedures” are presented in Section 7.3.

1403 6.1 All SuperProperties Discovery Query

1404 As presented in Section 4.1.3, a new ebXML RIM Association Type called “SubPropertyOf” MUST be
1405 defined to represent rdfs:subPropertyOf in ebRIM. Such a semantic enhancement brings the following
1406 processing need: given a property, it should be possible to retrieve all of its super properties. This requires
1407 a recursion mechanism in SQL queries.

1408 The freebXML implementation allows various relational database products such as Oracle, PostgreSQL
1409 and MS SQL Server 2005 to be used as the database. These products have different support for
1410 recursion mechanism in SQL Queries.

1411 The AllSuperProperties discovery query MAY be implemented by an ebXML Registry implementing this
1412 profile. It allows the discovery of all super properties of a given property instance (Association instance in
1413 ebXML terminology) recursively in a property hierarchy (hierarchy of Association Types) in freebXML
1414 Registry implementations using MS SQL Server 2005 as the database.

1415 6.1.1 Parameter \$propertyName

1416 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1417 value of Associations that have associationType of Property.

1418 6.1.2 Example of All SuperProperties Discovery Query

1419 The following example illustrates how to find all the super properties of a given property having a name
1420 containing “creditCardPayment” if the query is implemented as an AdHoc Query.

1421

```
1422 <<rs:RequestSlotList>  
1423   <rim:Slot  
1424     name="urn:oasis:names:tc:ebxml-  
1425     regrep:3.0:rs:AdhocQueryRequest:queryId">  
1426     <rim:ValueList>  
1427       <rim:Value>urn:oasis:names:tc:ebxml-  
1428       regrep:query:FindAllSuperProperties</rim:Value>
```

```

1429         </rim:ValueList>
1430     </rim:Slot>
1431     <rim:Slot name="urn:oasis:names:tc:ebxml-
1432 regrep:rs:AdhocQueryRequest:queryId">
1433         <rim:ValueList>
1434             <rim:Value>urn:oasis:names:tc:ebxml-
1435 regrep:query:FindAllSuperProperties</rim:Value>
1436         </rim:ValueList>
1437     </rim:Slot>
1438     <rim:Slot name="$propertyName">
1439         <rim:ValueList>
1440             <rim:Value>%creditCardPayment%</rim:Value>
1441         </rim:ValueList>
1442     </rim:Slot>
1443 </rs:RequestSlotList>
1444
1445 <query:ResponseOption returnComposedObjects="true"
1446     returnType="LeafClassWithRepositoryItem"/>
1447
1448 <rim:AdhocQuery id="temporaryId">
1449     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1450 regrep:QueryLanguage:SQL-92">
1451         </rim:QueryExpression>
1452 </rim:AdhocQuery>

```

1453 Example of All SuperProperties Discovery Query

1454 6.2 Immediate SuperClass Discovery Query

1455 The Immediate SuperClass discovery query MUST be implemented by an ebXML Registry implementing
1456 this profile. It allows the discovery of all of the immediate super classes of a given class.

1457 6.2.1 Parameter \$className

1458 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1459 value of ClassificationNodes.

1460 6.2.2 Example of Immediate SuperClass Discovery Query

1461 The following example illustrates how to find all the immediate super classes of a given class that have a
1462 name containing the string "AirReservationServices".

```

1463 <rs:RequestSlotList>
1464     <rim:Slot
1465         name="urn:oasis:names:tc:ebxml-
1466 regrep:3.0:rs:AdhocQueryRequest:queryId">
1467         <rim:ValueList>
1468             <rim:Value>urn:oasis:names:tc:ebxml-
1469 regrep:query:FindImmediateSuperClasses</rim:Value>
1470         </rim:ValueList>
1471     </rim:Slot>
1472     <rim:Slot name="urn:oasis:names:tc:ebxml-
1473 regrep:rs:AdhocQueryRequest:queryId">
1474         <rim:ValueList>
1475             <rim:Value>urn:oasis:names:tc:ebxml-
1476 regrep:query:FindImmediateSuperClasses</rim:Value>
1477         </rim:ValueList>
1478     </rim:Slot>
1479     <rim:Slot name="$className">
1480         <rim:ValueList>
1481             <rim:Value>%AirReservationServices%</rim:Value>
1482         </rim:ValueList>
1483     </rim:Slot>
1484 </rs:RequestSlotList>
1485
1486 <query:ResponseOption returnComposedObjects="true"

```



```

1487         returnType="LeafClassWithRepositoryItem"/>
1488
1489 <rim:AdhocQuery id="temporaryId">
1490     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1491     regrep:QueryLanguage:SQL-92">
1492         </rim:QueryExpression>
1493     </rim:AdhocQuery>

```

1494 Example of Immediate SuperClass Discovery Query

1495 6.3 Immediate SubClass Discovery Query

1496 The Immediate SubClass discovery query MUST be implemented by an ebXML Registry implementing
1497 this profile. It allows the discovery of all of the immediate subclasses of a given class.

1498 6.3.1 Parameter \$className

1499 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1500 value of ClassificationNode.

1501 6.3.2 Example of Immediate SubClass Discovery Query

1502 The following example illustrates how to find all the immediate subclasses of a given class that have a
1503 name containing the string "AirServices" .

```

1504 <rs:RequestSlotList>
1505     <rim:Slot
1506         name="urn:oasis:names:tc:ebxml-
1507     regrep:3.0:rs:AdhocQueryRequest:queryId">
1508         <rim:ValueList>
1509             <rim:Value>urn:oasis:names:tc:ebxml-
1510     regrep:query:FindImmediateSubClasses</rim:Value>
1511         </rim:ValueList>
1512     </rim:Slot>
1513     <rim:Slot name="urn:oasis:names:tc:ebxml-
1514     regrep:rs:AdhocQueryRequest:queryId">
1515         <rim:ValueList>
1516             <rim:Value>urn:oasis:names:tc:ebxml-
1517     regrep:query:FindImmediateSubClasses</rim:Value>
1518         </rim:ValueList>
1519     </rim:Slot>
1520     <rim:Slot name="$className">
1521         <rim:ValueList>
1522             <rim:Value>%AirServices%</rim:Value>
1523         </rim:ValueList>
1524     </rim:Slot>
1525 </rs:RequestSlotList>
1526
1527 <query:ResponseOption returnComposedObjects="true"
1528     returnType="LeafClassWithRepositoryItem"/>
1529
1530 <rim:AdhocQuery id="temporaryId">
1531     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1532     regrep:QueryLanguage:SQL-92">
1533         </rim:QueryExpression>
1534     </rim:AdhocQuery>

```

1535 Example of Immediate SubClass Discovery Query

1536 6.4 All SuperClasses Discovery Query

1537 It should be noted that, given a class, finding its immediate subclasses, super classes is necessary but
1538 not sufficient. Given a class, it should be possible to retrieve all of its subclasses, and all of its super
1539 classes. This requires a recursion mechanism in SQL queries. The freebXML implementation allows
1540 various relational database products such as Oracle, PostgreSQL and MS SQL Server 2005 to be used

1541 as the database. These products have different support for recursion mechanisms in SQL Queries.

1542 The All SuperClasses discovery query MAY be implemented by an ebXML Registry implementing this
1543 profile. It allows the discovery of all super classes of a given ClassificationNode recursively in freebXML
1544 Registry implementations using MS SQL Server 2005 as the database.

1545 6.4.1 Parameter \$className

1546 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1547 value of ClassificationNode.

1548 6.4.2 Example of All SuperClasses Discovery Query

1549 The following example illustrates how to find all the super classes of a given class recursively that have a
1550 name containing the string "AirReservationServices" if the query is implemented as an Adhoc Query .

```
1551 <rs:RequestSlotList>  
1552   <rim:Slot  
1553     name="urn:oasis:names:tc:ebxml-  
1554   regrep:3.0:rs:AdhocQueryRequest:queryId">  
1555     <rim:ValueList>  
1556       <rim:Value>urn:oasis:names:tc:ebxml-  
1557   regrep:query:FindAllSuperClasses</rim:Value>  
1558     </rim:ValueList>  
1559   </rim:Slot>  
1560   <rim:Slot name="urn:oasis:names:tc:ebxml-  
1561   regrep:rs:AdhocQueryRequest:queryId">  
1562     <rim:ValueList>  
1563       <rim:Value>urn:oasis:names:tc:ebxml-  
1564   regrep:query:FindAllSuperClasses</rim:Value>  
1565     </rim:ValueList>  
1566   </rim:Slot>  
1567   <rim:Slot name="$className">  
1568     <rim:ValueList>  
1569       <rim:Value>%AirReservationServices%</rim:Value>  
1570     </rim:ValueList>  
1571   </rim:Slot>  
1572 </rs:RequestSlotList>  
  
1573  
1574 <query:ResponseOption returnComposedObjects="true"  
1575   returnType="LeafClassWithRepositoryItem"/>  
1576  
1577 <rim:AdhocQuery id="temporaryId">  
1578   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
1579   regrep:QueryLanguage:SQL-92">  
1580     </rim:QueryExpression>  
1581 </rim:AdhocQuery>
```

1582 Example of All SuperClasses Discovery Query

1583 6.5 All SubClasses Discovery Query

1584 The All SubClasses discovery query MAY be implemented by an ebXML Registry implementing this
1585 profile. It allows the discovery of all subclasses of a given ClassificationNode recursively in a freebXML
1586 Registry implementations supporting recursion.

1587 6.5.1 Parameter \$className

1588 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1589 value of ClassificationNode.

1590 6.5.2 Example of All SubClasses Discovery Query

1591 The following example illustrates how to find all the subclasses of a given class recursively that have a

1592 name containing the string "AirServices" , if the query is implemented as an Adhoc Query.

```
1593 <rs:RequestSlotList>
1594   <rim:Slot
1595     name="urn:oasis:names:tc:ebxml-
1596   regrep:3.0:rs:AdhocQueryRequest:queryId">
1597     <rim:ValueList>
1598       <rim:Value>urn:oasis:names:tc:ebxml-
1599   regrep:query:FindAllSubClasses</rim:Value>
1600     </rim:ValueList>
1601   </rim:Slot>
1602   <rim:Slot name="urn:oasis:names:tc:ebxml-
1603   regrep:rs:AdhocQueryRequest:queryId">
1604     <rim:ValueList>
1605       <rim:Value>urn:oasis:names:tc:ebxml-
1606   regrep:query:FindAllSubClasses</rim:Value>
1607     </rim:ValueList>
1608   </rim:Slot>
1609   <rim:Slot name="$className">
1610     <rim:ValueList>
1611       <rim:Value>%AirServices%</rim:Value>
1612     </rim:ValueList>
1613   </rim:Slot>
1614 </rs:RequestSlotList>
1615
1616 <query:ResponseOption returnComposedObjects="true"
1617   returnType="LeafClassWithRepositoryItem"/>
1618
1619 <rim:AdhocQuery id="temporaryId">
1620   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1621   regrep:QueryLanguage:SQL-92">
1622     </rim:QueryExpression>
1623 </rim:AdhocQuery>
```

1624 Example of All SubClasses Discovery Query

1625 6.6 EquivalentClasses Discovery Query

1626 The EquivalentClasses discovery query MUST be implemented by an ebXML Registry implementing this
1627 profile. It allows the discovery of all the equivalent classes of a given ClassificationNode.

1628 6.6.1 Parameter \$className

1629 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1630 value of ClassificationNodes.

1631 6.6.2 Example of EquivalentClasses Discovery Query

1632 The following example illustrates how to find all the equivalent classes of a given class that have a name
1633 containing the string "AirServices" .

```
1634 <rs:RequestSlotList>
1635   <rim:Slot
1636     name="urn:oasis:names:tc:ebxml-
1637   regrep:3.0:rs:AdhocQueryRequest:queryId">
1638     <rim:ValueList>
1639       <rim:Value>urn:oasis:names:tc:ebxml-
1640   regrep:query:FindEquivalentClasses</rim:Value>
1641     </rim:ValueList>
1642   </rim:Slot>
1643   <rim:Slot name="urn:oasis:names:tc:ebxml-
1644   regrep:rs:AdhocQueryRequest:queryId">
1645     <rim:ValueList>
1646       <rim:Value>urn:oasis:names:tc:ebxml-
1647   regrep:query:FindEquivalentClasses</rim:Value>
1648     </rim:ValueList>
```

```

1649     </rim:Slot>
1650     <rim:Slot name="$className">
1651         <rim:ValueList>
1652             <rim:Value>%AirServices%</rim:Value>
1653         </rim:ValueList>
1654     </rim:Slot>
1655 </rs:RequestSlotList>
1656
1657 <query:ResponseOption returnComposedObjects="true"
1658     returnType="LeafClassWithRepositoryItem"/>
1659
1660 <rim:AdhocQuery id="temporaryId">
1661     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1662     regrep:QueryLanguage:SQL-92">
1663     </rim:QueryExpression>
1664 </rim:AdhocQuery>

```

1665 Example of Equivalent Classes Discovery Query

1666 6.7 EquivalentProperties Discovery Query

1667 The EquivalentProperties discovery query MUST be implemented by an ebXML Registry implementing
1668 this profile. It allows the discovery of all the equivalent properties of a given Association that have
1669 associationType of Property.

1670 6.7.1 Parameter \$propertyName

1671 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1672 value of Associations that have associationType of Property

1673 6.7.2 Example of EquivalentProperties Discovery Query

1674 The following example illustrates how to find all the equivalent properties(Association Type) of a given
1675 property (Association Type) that have a name containing the string "paymentMethods" .

```

1676 <rs:RequestSlotList>
1677     <rim:Slot
1678         name="urn:oasis:names:tc:ebxml-
1679     regrep:3.0:rs:AdhocQueryRequest:queryId">
1680         <rim:ValueList>
1681             <rim:Value>urn:oasis:names:tc:ebxml-
1682     regrep:query:FindEquivalentProperties</rim:Value>
1683         </rim:ValueList>
1684     </rim:Slot>
1685     <rim:Slot name="urn:oasis:names:tc:ebxml-
1686     regrep:rs:AdhocQueryRequest:queryId">
1687         <rim:ValueList>
1688             <rim:Value>urn:oasis:names:tc:ebxml-
1689     regrep:query:FindEquivalentProperties</rim:Value>
1690         </rim:ValueList>
1691     </rim:Slot>
1692     <rim:Slot name="$propertyName">
1693         <rim:ValueList>
1694             <rim:Value>%paymentMethods%</rim:Value>
1695         </rim:ValueList>
1696     </rim:Slot>
1697 </rs:RequestSlotList>
1698
1699 <query:ResponseOption returnComposedObjects="true"
1700     returnType="LeafClassWithRepositoryItem"/>
1701
1702 <rim:AdhocQuery id="temporaryId">
1703     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1704     regrep:QueryLanguage:SQL-92">
1705     </rim:QueryExpression>

```

1706 </rim:AdhocQuery>

1707 Example of Equivalent Properties Discovery Query

1708 6.8 SameExtrinsicObjects Discovery Query

1709 The SameExtrinsicObjects discovery query MUST be implemented by an ebXML Registry implementing
1710 this profile. It allows the discovery of all the "ExtrinsicObjects" defined to be the same with a given
1711 ExtrinsicObject.

1712 6.8.1 Parameter \$extrinsicObjectName

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

1715 6.8.2 Example of SameExtrinsicObjects Discovery Query

1716 The following example illustrates how to find all the ExtrinsicObjects that are defined to be the same as
1717 the ExtrinsicObject that have a name containing the string "MyDocument" .

```
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:query:FindTheSameExtrinsicObjects</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:query:FindTheSameExtrinsicObjects</rim:Value>
1732     </rim:ValueList>
1733   </rim:Slot>
1734   <rim:Slot name="$extrinsicObjectName">
1735     <rim:ValueList>
1736       <rim:Value>%MyDocument%</rim:Value>
1737     </rim:ValueList>
1738   </rim:Slot>
1739 </rs:RequestSlotList>
1740
1741 <query:ResponseOption returnComposedObjects="true"
1742   returnType="LeafClassWithRepositoryItem"/>
1743
1744 <rim:AdhocQuery id="temporaryId">
1745   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1746   regrep:QueryLanguage:SQL-92">
1747     </rim:QueryExpression>
1748 </rim:AdhocQuery>
```

1750 Example of SameExtrinsicObjects Discovery Query

1751 6.9 DifferentExtrinsicObjects Discovery Query

1752 The DifferentExtrinsicObjects discovery query MUST be implemented by an ebXML Registry
1753 implementing this profile. It allows the discovery of all the "ExtrinsicObjects" defined to be the different
1754 from a given ExtrinsicObject.

1755 6.9.1 Parameter \$extrinsicObjectName

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

1757 value of ExtrinsicObjects.

1758 6.9.2 Example of DifferentExtrinsicObjects Discovery Query

1759 The following example illustrates how to find all the ExtrinsicObjects that are defined to be different from
1760 the ExtrinsicObject that have a name containing the string "MyDocument" .

```
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:query:FindDifferentExtrinsicObjects</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:query:FindDifferentExtrinsicObjects</rim:Value>
1775     </rim:ValueList>
1776   </rim:Slot>
1777   <rim:Slot name="$extrinsicObjectName">
1778     <rim:ValueList>
1779       <rim:Value>%MyDocument%</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
```

1793 Example of DifferentExtrinsicObjects Discovery Query

1794 6.10 AllDifferentRegistryObject Discovery Query

1795 The AllDifferentRegistryObjects discovery query MUST be implemented by an ebXML Registry
1796 implementing this profile. Given a RegistryObject, it allows the discovery of all the other member
1797 "RegistryObjects" of a Registry package that are defined to be the different from each other through a
1798 allDifferent slot.

1799 6.10.1 Parameter \$registryObjectName

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

1802 6.10.2 Example of AllDifferentRegistryObjects Discovery Query

1803 The following example illustrates how to find all the RegistryObjects that are defined to be different from
1804 the RegistryObject that have a name containing the string "MyDocument" .

```
1805 <rs:RequestSlotList>
1806   <rim:Slot
1807     name="urn:oasis:names:tc:ebxml-
1808   regrep:3.0:rs:AdhocQueryRequest:queryId">
1809     <rim:ValueList>
1810
```

```

1811         <rim:Value>urn:oasis:names:tc:ebxml-
1812 regrep:query:FindAllDifferent</rim:Value>
1813     </rim:ValueList>
1814 </rim:Slot>
1815     <rim:Slot name="urn:oasis:names:tc:ebxml-
1816 regrep:rs:AdhocQueryRequest:queryId">
1817         <rim:ValueList>
1818             <rim:Value>urn:oasis:names:tc:ebxml-
1819 regrep:query:FindAllDifferent</rim:Value>
1820         </rim:ValueList>
1821     </rim:Slot>
1822     <rim:Slot name="$registryObjectName">
1823         <rim:ValueList>
1824             <rim:Value>%MyDocument%</rim:Value>
1825         </rim:ValueList>
1826     </rim:Slot>
1827 </rs:RequestSlotList>
1828
1829 <query:ResponseOption returnComposedObjects="true"
1830     returnType="LeafClassWithRepositoryItem"/>
1831
1832 <rim:AdhocQuery id="temporaryId">
1833     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1834 regrep:QueryLanguage:SQL-92">
1835     </rim:QueryExpression>
1836 </rim:AdhocQuery>

```

1837 Example of AllDifferentRegistryObjects Discovery Query

1838 6.11 ObjectProperties Discovery Query

1839 The ObjectProperties discovery query MUST be implemented by an ebXML Registry implementing this
1840 profile. It allows the discovery of all of the objectProperties of a given classification node.

1841 6.11.1 Parameter \$className

1842 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1843 value of ClassificationNodes.

1844 6.11.2 Example of ObjectProperties Discovery Query

1845 The following example illustrates how to find all the object properties of a given classification node having
1846 a name containing "AirServices" .

```

1847
1848 <rs:RequestSlotList>
1849     <rim:Slot
1850         name="urn:oasis:names:tc:ebxml-
1851 regrep:3.0:rs:AdhocQueryRequest:queryId">
1852         <rim:ValueList>
1853             <rim:Value>urn:oasis:names:tc:ebxml-
1854 regrep:query:FindObjectProperties</rim:Value>
1855         </rim:ValueList>
1856     </rim:Slot>
1857     <rim:Slot name="urn:oasis:names:tc:ebxml-
1858 regrep:rs:AdhocQueryRequest:queryId">
1859         <rim:ValueList>
1860             <rim:Value>urn:oasis:names:tc:ebxml-
1861 regrep:query:FindObjectProperties</rim:Value>
1862         </rim:ValueList>
1863     </rim:Slot>
1864     <rim:Slot name="$className">
1865         <rim:ValueList>
1866             <rim:Value>%AirServices%</rim:Value>
1867         </rim:ValueList>

```

```

1868     </rim:Slot>
1869 </rs:RequestSlotList>
1870
1871 <query:ResponseOption returnComposedObjects="true"
1872     returnType="LeafClassWithRepositoryItem"/>
1873
1874 <rim:AdhocQuery id="temporaryId">
1875     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1876     regrep:QueryLanguage:SQL-92">
1877         </rim:QueryExpression>
1878     </rim:AdhocQuery>

```

1879 Example of ObjectProperties Discovery Query

1880 6.12 ImmediateInheritedObjectProperties Discovery Query

1881 The ImmediateInheritedObjectProperties discovery query MUST be implemented by an ebXML Registry
1882 implementing this profile. It allows the discovery of all of the objectProperties of a given classification node
1883 including the ones inherited from its immediate super classes.

1884 6.12.1 Parameter \$className

1885 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1886 value of ClassificationNodes.

1887 6.12.2 Example of ImmediateInheritedObjectProperties Discovery Query

1888 The following example illustrates how to find all the object properties of a given classification node having
1889 a name containing "AirServices" including the ones inherited from its immediate super classes.

```

1890
1891 <rs:RequestSlotList>
1892     <rim:Slot
1893         name="urn:oasis:names:tc:ebxml-
1894     regrep:3.0:rs:AdhocQueryRequest:queryId">
1895         <rim:ValueList>
1896             <rim:Value>urn:oasis:names:tc:ebxml-
1897     regrep:query:FindImmediateInheritedObjectProperties</rim:Value>
1898         </rim:ValueList>
1899     </rim:Slot>
1900     <rim:Slot name="urn:oasis:names:tc:ebxml-
1901     regrep:rs:AdhocQueryRequest:queryId">
1902         <rim:ValueList>
1903             <rim:Value>urn:oasis:names:tc:ebxml-
1904     regrep:query:FindImmediateInheritedObjectProperties</rim:Value>
1905         </rim:ValueList>
1906     </rim:Slot>
1907     <rim:Slot name="$className">
1908         <rim:ValueList>
1909             <rim:Value>%AirServices%</rim:Value>
1910         </rim:ValueList>
1911     </rim:Slot>
1912 </rs:RequestSlotList>
1913
1914 <query:ResponseOption returnComposedObjects="true"
1915     returnType="LeafClassWithRepositoryItem"/>
1916
1917 <rim:AdhocQuery id="temporaryId">
1918     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1919     regrep:QueryLanguage:SQL-92">
1920         </rim:QueryExpression>
1921     </rim:AdhocQuery>

```

1922 Example of ImmediateInheritedObjectProperties Discovery Query

1923 6.13 AllInheritedObjectProperties Discovery Query

1924 It should be noted that, given a class, finding the object properties inherited from immediate super classes
1925 is necessary but not sufficient. Given a class, it should be possible to retrieve all of the object properties
1926 inherited from its super classes. This requires a recursion mechanism in SQL queries. The freebXML
1927 implementation allows various relational database products such as Oracle, PostgreSQL and MS SQL
1928 Server 2005 to be used as the database. These products have different support for recursion in SQL
1929 Queries.

1930 The AllInheritedObjectProperties discovery query MAY be implemented by an ebXML Registry
1931 implementing this profile. It allows the discovery of all inherited ObjectProperties recursively of a given
1932 ClassificationNode in a ClassificationScheme in freebXML Registry implementations using MS SQL
1933 Server 2005 as the database.

1934 6.13.1 Parameter \$className

1935 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1936 value of ClassificationNodes.

1937 6.13.2 Example of AllInheritedObjectProperties Discovery Query

1938 The following example illustrates how to find all the object properties of a given classification node having
1939 a name containing "AirReservationServices" including the ones inherited from all of its super classes
1940 recursively, if the query is implemented as an Adhoc Query.

1941

```
1942 <rs:RequestSlotList>
1943   <rim:Slot
1944     name="urn:oasis:names:tc:ebxml-
1945   regrep:3.0:rs:AdhocQueryRequest:queryId">
1946     <rim:ValueList>
1947       <rim:Value>urn:oasis:names:tc:ebxml-
1948   regrep:query:FindAllInheritedObjectProperties</rim:Value>
1949     </rim:ValueList>
1950   </rim:Slot>
1951   <rim:Slot name="urn:oasis:names:tc:ebxml-
1952   regrep:rs:AdhocQueryRequest:queryId">
1953     <rim:ValueList>
1954       <rim:Value>urn:oasis:names:tc:ebxml-regrep:query:FindAll
1955   InheritedObjectProperties</rim:Value>
1956     </rim:ValueList>
1957   </rim:Slot>
1958   <rim:Slot name="$className">
1959     <rim:ValueList>
1960       <rim:Value>%AirReservationServices%</rim:Value>
1961     </rim:ValueList>
1962   </rim:Slot>
1963 </rs:RequestSlotList>
1964
1965 <query:ResponseOption returnComposedObjects="true"
1966   returnType="LeafClassWithRepositoryItem"/>
1967
1968 <rim:AdhocQuery id="temporaryId">
1969   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1970   regrep:QueryLanguage:SQL-92">
1971     </rim:QueryExpression>
1972 </rim:AdhocQuery>
```

1973 Example of AllInheritedObjectProperties Discovery Query

1974 6.14 DatatypeProperties Discovery Query

1975 The DatatypeProperties discovery query MUST be implemented by an ebXML Registry implementing this
1976 profile. It allows the discovery of all of the datatypeProperties of a given classification node.

1977 6.14.1 Parameter \$className

1978 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1979 value of ClassificationNodes.

1980 6.14.2 Example of DatatypeProperties Discovery Query

1981 The following example illustrates how to find all the datatype properties of a given classification node
1982 having a name containing "AirReservationServices" .

1983

```
1984 <rs:RequestSlotList>  
1985   <rim:Slot  
1986     name="urn:oasis:names:tc:ebxml-  
1987   regrep:3.0:rs:AdhocQueryRequest:queryId">  
1988     <rim:ValueList>  
1989       <rim:Value>urn:oasis:names:tc:ebxml-  
1990   regrep:query:FindDatatypeProperties</rim:Value>  
1991     </rim:ValueList>  
1992   </rim:Slot>  
1993   <rim:Slot name="urn:oasis:names:tc:ebxml-  
1994   regrep:rs:AdhocQueryRequest:queryId">  
1995     <rim:ValueList>  
1996       <rim:Value>urn:oasis:names:tc:ebxml-  
1997   regrep:query:FindDatatypeProperties</rim:Value>  
1998     </rim:ValueList>  
1999   </rim:Slot>  
2000   <rim:Slot name="$className">  
2001     <rim:ValueList>  
2002       <rim:Value>%AirReservationServices%</rim:Value>  
2003     </rim:ValueList>  
2004   </rim:Slot>  
2005 </rs:RequestSlotList>  
  
2006  
2007 <query:ResponseOption returnComposedObjects="true"  
2008   returnType="LeafClassWithRepositoryItem"/>  
2009  
2010 <rim:AdhocQuery id="temporaryId">  
2011   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
2012   regrep:QueryLanguage:SQL-92">  
2013     </rim:QueryExpression>  
2014 </rim:AdhocQuery>
```

2006

Example of DatatypeProperties Discovery Query

2016 6.15 AllInheritedDatatypeProperties Discovery Query

2017 It should be noted that, given a class, finding the datatype properties inherited from immediate super
2018 classes is necessary but not sufficient. Given a class, it should be possible to retrieve all of the datatype
2019 properties inherited from its super classes. This requires a recursion mechanism in SQL queries. The
2020 freebXML implementation allows various relational database products such as Oracle, PostgreSQL and
2021 MS SQL Server 2005 to be used as the database. These products have different support for recursion in
2022 SQL Queries.

2023 The AllInheritedDatatypeProperties discovery query MAY be implemented by an ebXML Registry
2024 implementing this profile. It allows the discovery of all inherited DatatypeProperties recursively of a given
2025 ClassificationNode in a ClassificationScheme in freebXML Registry implementations using MS SQL
2026 Server 2005 as the database.

2027 6.15.1 Parameter \$className

2028 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2029 value of ClassificationNodes.

2030 6.15.2 Example of AllInheritedDatatypeProperties Discovery Query

2031 The following example illustrates how to find all the datatype properties of a given classification node
2032 having a name containing "AirReservationServices" including the ones inherited from all of its super
2033 classes recursively, if the query is implemented as an Adhoc Query.

2034

```
2035 <rs:RequestSlotList>  
2036   <rim:Slot  
2037     name="urn:oasis:names:tc:ebxml-  
2038     regrep:3.0:rs:AdhocQueryRequest:queryId">  
2039     <rim:ValueList>  
2040       <rim:Value>urn:oasis:names:tc:ebxml-  
2041       regrep:query:FindAllInheritedDatatypeProperties</rim:Value>  
2042     </rim:ValueList>  
2043   </rim:Slot>  
2044   <rim:Slot name="urn:oasis:names:tc:ebxml-  
2045   regrep:rs:AdhocQueryRequest:queryId">  
2046     <rim:ValueList>  
2047       <rim:Value>urn:oasis:names:tc:ebxml-  
2048       regrep:query:FindAllInheritedDatatypeProperties</rim:Value>  
2049     </rim:ValueList>  
2050   </rim:Slot>  
2051   <rim:Slot name="$className">  
2052     <rim:ValueList>  
2053       <rim:Value>%AirReservationServices %</rim:Value>  
2054     </rim:ValueList>  
2055   </rim:Slot>  
2056 </rs:RequestSlotList>  
2057  
2058 <query:ResponseOption returnComposedObjects="true"  
2059   returnType="LeafClassWithRepositoryItem"/>  
2060  
2061 <rim:AdhocQuery id="temporaryId">  
2062   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
2063   regrep:QueryLanguage:SQL-92">  
2064     </rim:QueryExpression>  
2065 </rim:AdhocQuery>
```

2066 Example of AllInheritedDatatypeProperties Discovery Query

2067 6.16 TransitiveRelationships Discovery Query

2068 To make any use of the transitive property in ebXML registries, coding is necessary to find out the implied
2069 information. The TransitiveRelationships discovery query MUST be implemented by an ebXML Registry
2070 implementing this profile to handle this semantics.

2071 Given a class which is a source of a transitive property, this discovery query retrieves not only the target
2072 objects of a given transitive property, but if the target objects have the same property, it retrieves their
2073 target objects too.

2074 6.16.1 Parameter \$className

2075 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2076 value of ClassificationNodes.

2077 6.16.2 Parameter \$propertyName

2078 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2079 value of Associations that have associationType of Property

2080 6.16.3 Example of TransitiveRelationships Discovery Query

2081 The following example illustrates how to retrieve all the target objects of the "succeeds" property of the

2082 “AirReservationServices” including the target objects implied by a transitive property relationship.

2083

```
2084 <rs:RequestSlotList>
2085   <rim:Slot
2086     name="urn:oasis:names:tc:ebxml-
2087   regrep:3.0:rs:AdhocQueryRequest:queryId">
2088     <rim:ValueList>
2089       <rim:Value>urn:oasis:names:tc:ebxml-
2090   regrep:query:FindTransitiveRelationships</rim:Value>
2091     </rim:ValueList>
2092   </rim:Slot>
2093   <rim:Slot name="urn:oasis:names:tc:ebxml-
2094   regrep:rs:AdhocQueryRequest:queryId">
2095     <rim:ValueList>
2096       <rim:Value>urn:oasis:names:tc:ebxml-
2097   regrep:query:FindTransitiveRelationships</rim:Value>
2098     </rim:ValueList>
2099   </rim:Slot>
2100   <rim:Slot name="$className">
2101     <rim:ValueList>
2102       <rim:Value>%AirReservationServices%</rim:Value>
2103     </rim:ValueList>
2104   </rim:Slot>
2105   <rim:Slot name="$propertyName">
2106     <rim:ValueList>
2107       <rim:Value>%succeeds%</rim:Value>
2108     </rim:ValueList>
2109   </rim:Slot>
2110 </rs:RequestSlotList>
2111
2112 <query:ResponseOption returnComposedObjects="true"
2113   returnType="LeafClassWithRepositoryItem"/>
2114
2115 <rim:AdhocQuery id="temporaryId">
2116   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2117   regrep:QueryLanguage:SQL-92">
2118     </rim:QueryExpression>
2119 </rim:AdhocQuery>
```

2120

Example of TransitiveRelationships Discovery Query

2121 6.17 TargetObjects Discovery Query

2122 The TargetObjects discovery query MUST be implemented by an ebXML Registry implementing this
2123 profile. It allows the discovery of the targetObjects from the Registry, given a Classification Node
2124 (sourceObject) and a property name (Association Type).

2125 6.17.1 Parameter \$className

2126 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2127 value of ClassificationNodes.

2128 6.17.2 Parameter \$propertyName

2129 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2130 value of Associations that have associationType of Property.

2131 6.17.3 Example of TargetObjects Discovery Query

2132 The following example illustrates how to retrieve all the target objects of the “paymentMethod” property of
2133 the “AirReservationServices”.

2134

```

2135 <rs:RequestSlotList>
2136   <rim:Slot
2137     name="urn:oasis:names:tc:ebxml-
2138   regrep:3.0:rs:AdhocQueryRequest:queryId">
2139     <rim:ValueList>
2140       <rim:Value>urn:oasis:names:tc:ebxml-
2141   regrep:query:FindTargetObjects</rim:Value>
2142     </rim:ValueList>
2143   </rim:Slot>
2144   <rim:Slot name="urn:oasis:names:tc:ebxml-
2145   regrep:rs:AdhocQueryRequest:queryId">
2146     <rim:ValueList>
2147       <rim:Value>urn:oasis:names:tc:ebxml-
2148   regrep:query:FindTargetObjects</rim:Value>
2149     </rim:ValueList>
2150   </rim:Slot>
2151   <rim:Slot name="$className">
2152     <rim:ValueList>
2153       <rim:Value>%AirReservationServices%</rim:Value>
2154     </rim:ValueList>
2155   </rim:Slot>
2156   <rim:Slot name="$propertyName">
2157     <rim:ValueList>
2158       <rim:Value>%paymentMethod%</rim:Value>
2159     </rim:ValueList>
2160   </rim:Slot>
2161 </rs:RequestSlotList>
2162
2163 <query:ResponseOption returnComposedObjects="true"
2164   returnType="LeafClassWithRepositoryItem"/>
2165
2166 <rim:AdhocQuery id="temporaryId">
2167   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2168   regrep:QueryLanguage:SQL-92">
2169     </rim:QueryExpression>
2170 </rim:AdhocQuery>

```

2171 Example of TargetObjects Discovery Query

2172

2173 6.18 TargetObjectsInverseOf Discovery Query

2174 The TargetObjectsInverseOf discovery query MUST be implemented by an ebXML Registry implementing
2175 this profile. Given a Classification Node (sourceObject) and a property name (Association Type), this
2176 query retrieves the source objects of the properties which are stated to be inverseOf the property name
2177 given as a parameter, and considering the Classification Node name as the targetObject of these
2178 properties.

2179 6.18.1 Parameter \$className

2180 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2181 value of ClassificationNodes.

2182 6.18.2 Parameter \$propertyName

2183 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2184 value of Associations that have associationType of Property.

2185 6.18.3 Example of TargetObjectsInverseOf Discovery Query

2186 The following example illustrates how to retrieve all the source objects of the properties which are stated
2187 to the the inverseOf the property "succeeds", considering the "AirReservationServices" as the target object
2188 of these properties.

2189

```
2190 <rs:RequestSlotList>
2191   <rim:Slot
2192     name="urn:oasis:names:tc:ebxml-
2193   regrep:3.0:rs:AdhocQueryRequest:queryId">
2194     <rim:ValueList>
2195       <rim:Value>urn:oasis:names:tc:ebxml-
2196   regrep:query:FindTOinverseOf</rim:Value>
2197     </rim:ValueList>
2198   </rim:Slot>
2199   <rim:Slot name="urn:oasis:names:tc:ebxml-
2200   regrep:rs:AdhocQueryRequest:queryId">
2201     <rim:ValueList>
2202       <rim:Value>urn:oasis:names:tc:ebxml-
2203   regrep:query:FindTOinverseOf</rim:Value>
2204     </rim:ValueList>
2205   </rim:Slot>
2206   <rim:Slot name="$className">
2207     <rim:ValueList>
2208       <rim:Value>%AirReservationServices%</rim:Value>
2209     </rim:ValueList>
2210   </rim:Slot>
2211   <rim:Slot name="$propertyName">
2212     <rim:ValueList>
2213       <rim:Value>%succeeds%</rim:Value>
2214     </rim:ValueList>
2215   </rim:Slot>
2216 </rs:RequestSlotList>
2217
2218 <query:ResponseOption returnComposedObjects="true"
2219   returnType="LeafClassWithRepositoryItem"/>
2220
2221 <rim:AdhocQuery id="temporaryId">
2222   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2223   regrep:QueryLanguage:SQL-92">
2224     </rim:QueryExpression>
2225 </rim:AdhocQuery>
```

2226 Example of TargetObjectsInverseOf Discovery Query

2227

2228 6.19 InverseRanges Discovery Query

2229 The InverseRanges discovery query MUST be implemented by an ebXML Registry implementing this
2230 profile to handle this semantics. Given a Classification Node (sourceObject) and a property name
2231 (Association Type), this query retrieves not only the target objects of this property, but also the source
2232 objects of the properties which are stated to be inverseOf the property name given as a parameter, and
2233 considering the Classification Node name as the targetObject of these properties. This query can be
2234 thought as the union of the queries presented in Sections 6.17 and 6.18.

2235 6.19.1 Parameter \$className

2236 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2237 value of ClassificationNodes.

2238 6.19.2 Parameter \$propertyName

2239 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2240 value of Associations that have associationType of Property

2241 6.19.3 Example of InverseRanges Discovery Query

2242 Consider, for example, the "succeeds" property defined in Section 4.3.3. To denote that a certain Web

2243 service instance precedes another during execution, we may define the "precedes" property as an inverse
2244 of the "succeeds" property as follows:
2245

```
2246 <owl:ObjectProperty rdf:ID="precedes">  
2247   <owl:inverseOf rdf:resource="#succeeds" />  
2248 </owl:ObjectProperty>
```

2249 **Example owl:inverseOf Property**

2250 Assume that we want to find all the Web services which can succeed a given Web service. In such a
2251 case, we need not only find all the Web services which succeeds this given Web service, that is the target
2252 objects of "succeeds" Association instance, but we also need to find all the sourceObjects of the
2253 "precedes" Association instance since "precedes" is declared to be the "inverseOf" succeeds Association
2254 instance.

2255 The following example illustrates how to retrieve all the services that "succeeds" "AirReservationServices"
2256 by also making use of its "precedes" property.

2257

```
2258 <rs:RequestSlotList>  
2259   <rim:Slot  
2260     name="urn:oasis:names:tc:ebxml-  
2261     regrep:3.0:rs:AdhocQueryRequest:queryId">  
2262     <rim:ValueList>  
2263       <rim:Value>urn:oasis:names:tc:ebxml-  
2264       regrep:query:FindInverseRanges</rim:Value>  
2265     </rim:ValueList>  
2266   </rim:Slot>  
2267   <rim:Slot name="urn:oasis:names:tc:ebxml-  
2268   regrep:rs:AdhocQueryRequest:queryId">  
2269     <rim:ValueList>  
2270       <rim:Value>urn:oasis:names:tc:ebxml-  
2271       regrep:query:FindInverseRanges</rim:Value>  
2272     </rim:ValueList>  
2273   </rim:Slot>  
2274   <rim:Slot name="$className">  
2275     <rim:ValueList>  
2276       <rim:Value>%AirReservationServices%</rim:Value>  
2277     </rim:ValueList>  
2278   </rim:Slot>  
2279   <rim:Slot name="$propertyName">  
2280     <rim:ValueList>  
2281       <rim:Value>%succeeds%</rim:Value>  
2282     </rim:ValueList>  
2283   </rim:Slot>  
2284 </rs:RequestSlotList>  
  
2285 <query:ResponseOption returnComposedObjects="true"  
2286   returnType="LeafClassWithRepositoryItem"/>  
2287  
2288 <rim:AdhocQuery id="temporaryId">  
2289   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
2290   regrep:QueryLanguage:SQL-92">  
2291     </rim:QueryExpression>  
2292 </rim:AdhocQuery>
```

2294 **Example of InverseRanges Discovery Query**

2295 **6.20 SymmetricProperties Discovery Query**

2296 The SymmetricProperties discovery query MUST be implemented by an ebXML Registry implementing
2297 this profile. It allows the discovery of all of the Symmetric Properties of a given classification node.

2298 **6.20.1 Parameter \$className**

2299 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2300 value of ClassificationNodes.

2301 **6.20.2 Example of SymmetricProperties Discovery Query**

2302 The following example illustrates how to find all the symmetric properties of a given classification node
2303 having a name containing "AirReservationServices" .

2304

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

2336 Example of SymmetricProperties Discovery Query

2337 **6.21 FunctionalProperties Discovery Query**

2338 The FunctionalProperties discovery query MUST be implemented by an ebXML Registry implementing
2339 this profile. It allows the discovery of all of the Functional Properties of a given classification node.

2340 **6.21.1 Parameter \$className**

2341 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2342 value of ClassificationNodes.

2343 **6.21.2 Example of FunctionalProperties Discovery Query**

2344 The following example illustrates how to find all the functional properties of a given classification node
2345 having a name containing "AirReservationServices" .

2346

```
2347 <rs:RequestSlotList>  
2348   <rim:Slot
```

```

2349     name="urn:oasis:names:tc:ebxml-
2350 regrep:3.0:rs:AdhocQueryRequest:queryId">
2351     <rim:ValueList>
2352         <rim:Value>urn:oasis:names:tc:ebxml-
2353 regrep:query:FindFunctionalProperties</rim:Value>
2354     </rim:ValueList>
2355     </rim:Slot>
2356     <rim:Slot name="urn:oasis:names:tc:ebxml-
2357 regrep:rs:AdhocQueryRequest:queryId">
2358         <rim:ValueList>
2359             <rim:Value>urn:oasis:names:tc:ebxml-
2360 regrep:query:FindFunctionalProperties</rim:Value>
2361         </rim:ValueList>
2362     </rim:Slot>
2363     <rim:Slot name="$className">
2364         <rim:ValueList>
2365             <rim:Value>%AirReservationServices%</rim:Value>
2366         </rim:ValueList>
2367     </rim:Slot>
2368 </rs:RequestSlotList>
2369
2370 <query:ResponseOption returnComposedObjects="true"
2371     returnType="LeafClassWithRepositoryItem"/>
2372
2373 <rim:AdhocQuery id="temporaryId">
2374     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2375 regrep:QueryLanguage:SQL-92">
2376     </rim:QueryExpression>
2377 </rim:AdhocQuery>

```

Example of Functional Properties Discovery Query

2379 6.22 InverseFunctionalProperties Discovery Query

2380 The InverseFunctionalProperties discovery query MUST be implemented by an ebXML Registry
2381 implementing this profile. It allows the discovery of all of the Inverse Functional Properties of a given
2382 classification node.

2383 6.22.1 Parameter \$className

2384 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2385 value of ClassificationNodes.

2386 6.22.2 Example of InverseFunctionalProperties Discovery Query

2387 The following example illustrates how to find all the inverse functional properties of a given classification
2388 node having a name containing "AirReservationServices" .

```

2389
2390 <rs:RequestSlotList>
2391     <rim:Slot
2392         name="urn:oasis:names:tc:ebxml-
2393 regrep:3.0:rs:AdhocQueryRequest:queryId">
2394         <rim:ValueList>
2395             <rim:Value>urn:oasis:names:tc:ebxml-
2396 regrep:query:FindInverseFunctionalProperties</rim:Value>
2397         </rim:ValueList>
2398     </rim:Slot>
2399     <rim:Slot name="urn:oasis:names:tc:ebxml-
2400 regrep:rs:AdhocQueryRequest:queryId">
2401         <rim:ValueList>
2402             <rim:Value>urn:oasis:names:tc:ebxml-
2403 regrep:query:FindInverseFunctionalProperties</rim:Value>
2404         </rim:ValueList>
2405     </rim:Slot>

```



```

2406     <rim:Slot name="$className">
2407         <rim:ValueList>
2408             <rim:Value>%AirReservationServices%</rim:Value>
2409         </rim:ValueList>
2410     </rim:Slot>
2411 </rs:RequestSlotList>
2412
2413 <query:ResponseOption returnComposedObjects="true"
2414     returnType="LeafClassWithRepositoryItem"/>
2415
2416 <rim:AdhocQuery id="temporaryId">
2417     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2418     regrep:QueryLanguage:SQL-92">
2419         </rim:QueryExpression>
2420 </rim:AdhocQuery>

```

2421 Example of InverseFunctional Properties Discovery Query

2422 6.23 Instances Discovery Query

2423 When an intersection definition is used to create a complex class (a new ClassificationNode) in RIM as
 2424 described in Section 4.6, it becomes possible to infer that the objects (instances) classified by both of the
 2425 classes (ClassificationNodes) constituting the intersection are also the instances of this complex class.

2426 The Instances discovery query MUST be implemented by an ebXML Registry implementing this profile. It
 2427 allows the discovery of all of the direct instances of a given classification node and if it is a complex class
 2428 which is an intersection two classes, it also allows to retrieve the intersection of the instances of both of
 2429 the classes involved in the intersection definition.

2430 6.23.1 Parameter \$className

2431 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
 2432 value of ClassificationNodes.

2433 6.23.2 Example of Instances Discovery Query

2434 Consider the "AirReservationServices" definition presented in Section 4.6. The following example
 2435 illustrates how to find all the direct instances of the "AirReservationServices" and also the instances
 2436 classified by both "AirServices" and also the "ReservationServices".

```

2437
2438 <rs:RequestSlotList>
2439     <rim:Slot
2440         name="urn:oasis:names:tc:ebxml-
2441     regrep:3.0:rs:AdhocQueryRequest:queryId">
2442         <rim:ValueList>
2443             <rim:Value>urn:oasis:names:tc:ebxml-
2444     regrep:query:FindInstances</rim:Value>
2445         </rim:ValueList>
2446     </rim:Slot>
2447     <rim:Slot name="urn:oasis:names:tc:ebxml-
2448     regrep:rs:AdhocQueryRequest:queryId">
2449         <rim:ValueList>
2450             <rim:Value>urn:oasis:names:tc:ebxml-
2451     regrep:query:FindInstances</rim:Value>
2452         </rim:ValueList>
2453     </rim:Slot>
2454     <rim:Slot name="$className">
2455         <rim:ValueList>
2456             <rim:Value>%AirReservationServices%</rim:Value>
2457         </rim:ValueList>
2458     </rim:Slot>
2459 </rs:RequestSlotList>
2460
2461 <query:ResponseOption returnComposedObjects="true"

```


2462
2463
2464
2465
2466
2467
2468
2469

```
returnType="LeafClassWithRepositoryItem"/>  
<rim:AdhocQuery id="temporaryId">  
  <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
  regrep:QueryLanguage:SQL-92">  
    </rim:QueryExpression>  
  </rim:AdhocQuery>
```

Example of Instances Discovery Query

2470

7 Canonical Metadata Definitions

2471 This chapter specifies the canonical metadata defined by this profile.

7.1 ObjectType Extensions

2473 The following new extensions to the canonical ObjectType ClassificationScheme are described by this
2474 profile:

2475

```
2476 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2477 regrep:ObjectType:RegistryObject:ExtrinsicObject"  
2478 lid="urn:oasis:names:tc:ebxml-  
2479 regrep:ObjectType:RegistryObject:ExtrinsicObject:OWL" code="OWL"  
2480 id="urn:oasis:names:tc:ebxml-  
2481 regrep:ObjectType:RegistryObject:ExtrinsicObject:OWL">  
2482 <rim:Name>  
2483 <rim:LocalizedString charset="UTF-8" value="label.OWL"/>  
2484 </rim:Name>  
2485 </rim:ClassificationNode>
```

7.2 AssociationType Extensions

2486 The following new extensions to the AssociationType ClassificationScheme are described by this profile:

2487

```
2488  
2489 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2490 regrep:classificationScheme:AssociationType"  
2491 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:ObjectProperty"  
2492 code="ObjectProperty" id="urn:oasis:names:tc:ebxml-  
2493 regrep:AssociationType:ObjectProperty">  
2494 <rim:Name>  
2495 <rim:LocalizedString charset="UTF-8"  
2496 value="ObjectProperty"/>  
2497 </rim:Name>  
2498 </rim:ClassificationNode>  
2499 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2500 regrep:classificationScheme:AssociationType"  
2501 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:Property"  
2502 code="Property" id="urn:oasis:names:tc:ebxml-  
2503 regrep:AssociationType:Property">  
2504 <rim:Name>  
2505 <rim:LocalizedString charset="UTF-8" value="Property"/>  
2506 </rim:Name>  
2507 </rim:ClassificationNode>  
2508 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2509 regrep:classificationScheme:AssociationType"  
2510 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:SubPropertyOf"  
2511 code="SubPropertyOf" id="urn:oasis:names:tc:ebxml-  
2512 regrep:AssociationType:SubPropertyOf">  
2513 <rim:Name>  
2514 <rim:LocalizedString charset="UTF-8" value="SubPropertyOf"/>  
2515 </rim:Name>  
2516 </rim:ClassificationNode>  
2517 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2518 regrep:classificationScheme:AssociationType"  
2519 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:SubClassOf"  
2520 code="SubClassOf" id="urn:oasis:names:tc:ebxml-  
2521 regrep:AssociationType:SubClassOf">  
2522 <rim:Name>  
2523 <rim:LocalizedString charset="UTF-8" value="SubClassOf"/>  
2524 </rim:Name>  
2525 </rim:ClassificationNode>  
2526
```

```

2527 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2528 regrep:classificationScheme:AssociationType"
2529 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:IntersectionOf"
2530 code="IntersectionOf" id="urn:oasis:names:tc:ebxml-
2531 regrep:AssociationType:IntersectionOf">
2532   <rim:Name>
2533     <rim:LocalizedString charset="UTF-8"
2534 value="IntersectionOf"/>
2535   </rim:Name>
2536 </rim:ClassificationNode>
2537
2538 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2539 regrep:classificationScheme:AssociationType"
2540 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:SameAs"
2541 code="SameAs" id="urn:oasis:names:tc:ebxml-
2542 regrep:AssociationType:SameAs">
2543   <rim:Name>
2544     <rim:LocalizedString charset="UTF-8" value="SameAs"/>
2545   </rim:Name>
2546 </rim:ClassificationNode>
2547
2548 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2549 regrep:classificationScheme:AssociationType"
2550 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:DifferentFrom"
2551 code="DifferentFrom" id="urn:oasis:names:tc:ebxml-
2552 regrep:AssociationType:DifferentFrom">
2553   <rim:Name>
2554     <rim:LocalizedString charset="UTF-8" value="DifferentFrom"/>
2555   </rim:Name>
2556 </rim:ClassificationNode>
2557
2558 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2559 regrep:classificationScheme:AssociationType"
2560 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:DatatypeProperty"
2561 code="DatatypeProperty" id="urn:oasis:names:tc:ebxml-
2562 regrep:AssociationType:DatatypeProperty">
2563   <rim:Name>
2564     <rim:LocalizedString charset="UTF-8"
2565 value="DatatypeProperty"/>
2566   </rim:Name>
2567 </rim:ClassificationNode>
2568
2569 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2570 regrep:classificationScheme:AssociationType"
2571 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:TransitiveProperty"
2572 code="TransitiveProperty" id="urn:oasis:names:tc:ebxml-
2573 regrep:AssociationType:TransitiveProperty">
2574   <rim:Name>
2575     <rim:LocalizedString charset="UTF-8"
2576 value="TransitiveProperty"/>
2577   </rim:Name>
2578 </rim:ClassificationNode>
2579
2580 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2581 regrep:classificationScheme:AssociationType"
2582 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:InverseOf"
2583 code="InverseOf" id="urn:oasis:names:tc:ebxml-
2584 regrep:AssociationType:InverseOf">
2585   <rim:Name>
2586     <rim:LocalizedString charset="UTF-8" value="InverseOf"/>
2587   </rim:Name>
2588 </rim:ClassificationNode>
2589

```

```

2590 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2591 regrep:classificationScheme:AssociationType"
2592 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:SymmetricProperty"
2593 code="SymmetricProperty" id="urn:oasis:names:tc:ebxml-
2594 regrep:AssociationType:SymmetricProperty">
2595   <rim:Name>
2596     <rim:LocalizedString charset="UTF-8"
2597 value="SymmetricProperty"/>
2598   </rim:Name>
2599 </rim:ClassificationNode>
2600
2601 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2602 regrep:classificationScheme:AssociationType"
2603 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:FunctionalProperty"
2604 code="FunctionalProperty" id="urn:oasis:names:tc:ebxml-
2605 regrep:AssociationType:FunctionalProperty">
2606   <rim:Name>
2607     <rim:LocalizedString charset="UTF-8"
2608 value="FunctionalProperty"/>
2609   </rim:Name>
2610 </rim:ClassificationNode>
2611
2612 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2613 regrep:classificationScheme:AssociationType"
2614 lid="urn:oasis:names:tc:ebxml-
2615 regrep:AssociationType:InverseFunctionalProperty"
2616 code="InverseFunctionalProperty" id="urn:oasis:names:tc:ebxml-
2617 regrep:AssociationType:InverseFunctionalProperty">
2618   <rim:Name>
2619     <rim:LocalizedString charset="UTF-8"
2620 value="InverseFunctionalProperty"/>
2621   </rim:Name>
2622 </rim:ClassificationNode>

```

2623 Extensions to the AssociationType ClassificationScheme

2624 7.3 Canonical Queries

2625 The following new canonical queries are described by this profile. Note that while these queries are
 2626 complex, the complexity is hidden from clients by exposing only the query parameters to them.

2627 7.3.1 All SuperProperties Discovery Query

2628 Since recursion is not supported by SQL-92, the stored procedure for this query is presented in this
 2629 section.

```

2630 CREATE PROCEDURE findAllSuperProperties
2631   @propertyName varchar(50)
2632 AS
2633 WITH
2634   Parents(superPropertyID) AS
2635   (
2636     SELECT A3.id
2637     FROM Association A1, Association A2, Association A3, Name_ N
2638     WHERE A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
2639 regrep:AssociationType:SubPropertyOf' AND
2640     A1.id = N.parent AND N.value LIKE @propertyName AND
2641     A2.sourceObject = A1.id AND A2.targetObject = A3.id
2642   UNION ALL
2643     SELECT A.targetObject
2644     FROM Association A JOIN Parents P
2645     ON P.superPropertyID = A.sourceObject
2646     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2647 regrep:AssociationType:SubPropertyOf'
2648   )
2649 SELECT * FROM Parents

```

2650

GO

2651 Recursive stored procedure for MS SQL Server 2005 retrieving all super properties of a given 2652 property (Association)

2653 7.3.2 Immediate SuperClass Discovery Query

```

2654 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2655 regrep:query:FindImmediateSuperClasses" id="urn:oasis:names:tc:ebxml-
2656 regrep:query:FindImmediateSuperClasses">
2657 <rim:Name>
2658 <rim:LocalizedString
2659 value="label.FindImmediateSuperClasses"/>
2660 </rim:Name>
2661 <rim:Description>
2662 <rim:LocalizedString
2663 value="label.FindImmediateSuperClasses.desc"/>
2664 </rim:Description>
2665 <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2666 regrep:QueryLanguage:SQL-92">
2667 SELECT C2.*
2668 FROM ClassificationNode C2, Association A, Name_ N,
2669 ClassificationNode C1
2670 WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2671 regrep:AssociationType:SubClassOf' AND
2672 C1.id = N.parent AND
2673 N.value LIKE '$className' AND
2674 A.sourceObject = C1.id AND
2675 A.targetObject = C2.id
2676 </rim:QueryExpression>
2677 </rim:AdhocQuery>
2678

```

2679 **The Adhoc Query retrieving immediate super classes of a given classification node**

2680 7.3.3 Immediate SubClass Discovery Query

```

2681 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2682 regrep:query:FindImmediateSubClasses" id="urn:oasis:names:tc:ebxml-
2683 regrep:query:FindImmediateSubClasses">
2684 <rim:Name>
2685 <rim:LocalizedString value="label.FindImmediateSubClasses"/>
2686 </rim:Name>
2687 <rim:Description>
2688 <rim:LocalizedString
2689 value="label.FindImmediateSubClasses.desc"/>
2690 </rim:Description>
2691 <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2692 regrep:QueryLanguage:SQL-92">
2693 SELECT C2.*
2694 FROM ClassificationNode C2, Association A, Name_ N,
2695 ClassificationNode C1
2696 WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2697 regrep:AssociationType:SubClassOf' AND
2698 C1.id = N.parent AND
2699 N.value LIKE '$className' AND
2700 A.sourceObject = C2.id AND
2701 A.targetObject = C1.id
2702 </rim:QueryExpression>
2703 </rim:AdhocQuery>

```

2704 **The Adhoc Query retrieving immediate subclasses of a given classification node**

2705 7.3.4 All SuperClasses Discovery Query

2706 Since recursion is not supported by SQL-92, the stored procedure for this query is presented in this
2707 section.

```

2708 CREATE PROCEDURE findAllSuperClasses
2709     @className varchar(50)
2710 AS
2711 WITH
2712     Parents(superClassID) AS
2713     (
2714         SELECT C2.id
2715         FROM Association A, Name_ N, ClassificationNode C1,
2716 ClassificationNode C2
2717         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2718 regrep:AssociationType:SubClassOf' AND
2719             C1.id = N.parent AND N.value LIKE @className AND
2720             A.sourceObject = C1.id AND A.targetObject = C2.id
2721     UNION ALL
2722         SELECT A.targetObject
2723         FROM Association A JOIN Parents P
2724         ON P.superClassID = A.sourceObject
2725         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2726 regrep:AssociationType:SubClassOf'
2727     )
2728 SELECT * FROM Parents
2729 GO

```

2730 **Recursive stored procedure for MS SQL Server 2005 database retrieving all super classes of a**
2731 **given classification node**

2732 7.3.5 All SubClasses Discovery Query

2733 Since recursion is not supported by SQL-92, the stored procedure for this query is presented in this
2734 section.

```

2735 CREATE PROCEDURE findAllSubClasses
2736     @className varchar(50)
2737 AS
2738 WITH
2739     Children(subClassID) AS
2740     (
2741         SELECT C1.id
2742         FROM Association A, Name_ N, ClassificationNode C1,
2743 ClassificationNode C2
2744         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2745 regrep:AssociationType:SubClassOf' AND
2746             C2.id = N.parent AND N.value LIKE @className AND
2747             A.sourceObject = C1.id AND A.targetObject = C2.id
2748     UNION ALL
2749         SELECT A.sourceObject
2750         FROM Association A JOIN Children C
2751         ON C.subClassID = A.targetObject
2752         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2753 regrep:AssociationType:SubClassOf'
2754     )
2755 SELECT * FROM Children
2756 GO

```

2757 **Recursive stored procedure for MS SQL Server 2005 database retrieving all subclasses of a**
2758 **given classification node**

2759 7.3.6 EquivalentClasses Discovery Query

```

2760 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2761 regrep:query:FindEquivalentClasses" id="urn:oasis:names:tc:ebxml-
2762 regrep:query:FindEquivalentClasses">
2763     <rim:Name>
2764         <rim:LocalizedString value="label.FindEquivalentClasses"/>
2765     </rim:Name>
2766     <rim:Description>

```

```

2767         <rim:LocalizedString
2768 value="label.FindEquivalentClasses.desc"/>
2769     </rim:Description>
2770     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:exml-
2771 regrep:QueryLanguage:SQL-92">
2772         SELECT C2.*
2773         FROM ClassificationNode C2, Association A, Name_ N,
2774 ClassificationNode C
2775         WHERE A.associationType LIKE 'urn:oasis:names:tc:exml-
2776 regrep:AssociationType:EquivalentTo' AND
2777         C.id = N.parent AND
2778         N.value LIKE '$className' AND
2779         A.sourceObject = C.id AND
2780         A.targetObject = C2.id
2781     </rim:QueryExpression>
2782 </rim:AdhocQuery>

```

2783 **Adhoc Query retrieving all the equivalent classes of a given classification node**

2784 7.3.7 EquivalentProperties Discovery Query

```

2785 <rim:AdhocQuery lid="urn:oasis:names:tc:exml-
2786 regrep:query:FindEquivalentProperties" id="urn:oasis:names:tc:exml-
2787 regrep:query:FindEquivalentProperties">
2788     <rim:Name>
2789         <rim:LocalizedString
2790 value="label.FindEquivalentProperties"/>
2791     </rim:Name>
2792     <rim:Description>
2793         <rim:LocalizedString
2794 value="label.FindEquivalentProperties.desc"/>
2795     </rim:Description>
2796     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:exml-
2797 regrep:QueryLanguage:SQL-92">
2798         SELECT A3.*
2799         FROM Association A3, Association A1, Name_ N, Association
2800 A2
2801         WHERE A1.associationType LIKE 'urn:oasis:names:tc:exml-
2802 regrep:AssociationType:EquivalentTo' AND
2803         A2.id = N.parent AND
2804         N.value LIKE '$propertyName' AND
2805         A1.sourceObject = A2.id AND
2806         A1.targetObject = A3.id
2807     </rim:QueryExpression>
2808 </rim:AdhocQuery>

```

2809 **Adhoc Query retrieving all the equivalent Association Type of a given Association Type**

2810 7.3.8 SameExtrinsicObjects Discovery Query

```

2811 <rim:AdhocQuery lid="urn:oasis:names:tc:exml-
2812 regrep:query:FindTheSameExtrinsicObjects" id="urn:oasis:names:tc:exml-
2813 regrep:query:FindTheSameExtrinsicObjects">
2814     <rim:Name>
2815         <rim:LocalizedString
2816 value="label.FindTheSameExtrinsicObjects"/>
2817     </rim:Name>
2818     <rim:Description>
2819         <rim:LocalizedString
2820 value="label.FindTheSameExtrinsicObjects.desc"/>
2821     </rim:Description>
2822     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:exml-
2823 regrep:QueryLanguage:SQL-92">
2824         SELECT E2.*
2825         FROM ExtrinsicObject E2, Association A, Name_ N,
2826 ExtrinsicObject E

```

```

2827         WHERE A.associationType LIKE 'urn:oasis:names:tc:exml-
2828 regrep:AssociationType:SameAs' AND
2829         E.id = N.parent AND
2830         N.value LIKE '$extrinsicObjectName' AND
2831         A.sourceObject = E.id AND
2832         A.targetObject = E2.id
2833     </rim:QueryExpression>
2834 </rim:AdhocQuery>

```

2835 **Adhoc Query retrieving all the "ExtrinsicObjects" defined to be the same with a given**
2836 **ExtrinsicObject**

2837 7.3.9 DifferentExtrinsicObjects Discovery Query

```

2838 <rim:AdhocQuery lid="urn:oasis:names:tc:exml-
2839 regrep:query:FindDifferentExtrinsicObjects" id="urn:oasis:names:tc:exml-
2840 regrep:query:FindDifferentExtrinsicObjects">
2841     <rim:Name>
2842         <rim:LocalizedString
2843 value="label.FindDifferentExtrinsicObjects"/>
2844     </rim:Name>
2845     <rim:Description>
2846         <rim:LocalizedString
2847 value="label.FindDifferentExtrinsicObjects.desc"/>
2848     </rim:Description>
2849     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:exml-
2850 regrep:QueryLanguage:SQL-92">
2851         SELECT E2.*
2852         FROM ExtrinsicObject E2, Association A, Name_ N,
2853 ExtrinsicObject E
2854         WHERE A.associationType LIKE 'urn:oasis:names:tc:exml-
2855 regrep:AssociationType:DifferentFrom' AND
2856         E.id = N.parent AND
2857         N.value LIKE '$extrinsicObjectName' AND
2858         A.sourceObject = E.id AND
2859         A.targetObject = E2.id
2860     </rim:QueryExpression>
2861 </rim:AdhocQuery>

```

2862 **Adhoc Query retrieving all the "ExtrinsicObjects" defined to be different from a given**
2863 **ExtrinsicObject**

2864 7.3.10 AllDifferentRegistryObject Discovery Query

```

2865 <rim:AdhocQuery lid="urn:oasis:names:tc:exml-
2866 regrep:query:FindAllDifferent" id="urn:oasis:names:tc:exml-
2867 regrep:query:FindAllDifferent">
2868     <rim:Name>
2869         <rim:LocalizedString value="label.FindAllDifferent"/>
2870     </rim:Name>
2871     <rim:Description>
2872         <rim:LocalizedString value="label.FindAllDifferent.desc"/>
2873     </rim:Description>
2874     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:exml-
2875 regrep:QueryLanguage:SQL-92">
2876         SELECT RO2.*
2877         FROM RegistryObject RO2, Association A1, Association A2,
2878 Name_ N, RegistryObject RO,
2879 RegistryPackage RP<!--, Slot S-->
2880         WHERE A1.associationType LIKE 'urn:oasis:names:tc:exml-
2881 regrep:AssociationType:HasMember' AND
2882         RO.id = N.parent AND
2883         N.value LIKE '$registryObjectName' AND
2884         A1.sourceObject = RP.id AND
2885         <!-- S.parent = RP.id AND
2886         S.name_ LIKE 'allDifferent' AND S.value LIKE 'true' AND
2887         -->

```



```

2888         A1.targetObject = RO.id AND
2889         A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
2890 regrep:AssociationType:HasMember' AND
2891         A2.sourceObject = RP.id AND
2892         A2.targetObject != RO.id AND
2893         A2.targetObject = R02.id
2894     </rim:QueryExpression>
2895 </rim:AdhocQuery>

```

2896 **Adhoc Query retrieving all the "RegistryObjects" defined to be different from a given**
2897 **RegistryObject through a "allDifferentFrom" construct**

2898 7.3.11 ObjectProperties Discovery Query

```

2899 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2900 regrep:query:FindObjectProperties" id="urn:oasis:names:tc:ebxml-
2901 regrep:query:FindObjectProperties">
2902     <rim:Name>
2903         <rim:LocalizedString value="label.FindObjectProperties"/>
2904     </rim:Name>
2905     <rim:Description>
2906         <rim:LocalizedString
2907 value="label.FindObjectProperties.desc"/>
2908     </rim:Description>
2909     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2910 regrep:QueryLanguage:SQL-92">
2911         SELECT A.*
2912         FROM Association A, Name_N, ClassificationNode C
2913         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2914 regrep:AssociationType:ObjectProperty' AND
2915         C.id = N.parent AND
2916         N.value LIKE '$className' AND
2917         A.sourceObject = C.id
2918     </rim:QueryExpression>
2919 </rim:AdhocQuery>

```

2920 **Adhoc Query retrieving all the object properties of a given classification node**

2921 7.3.12 ImmediateInheritedObjectProperties Discovery Query

```

2922 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2923 regrep:query:FindImmediateInheritedObjectProperties"
2924 id="urn:oasis:names:tc:ebxml-
2925 regrep:query:FindImmediateInheritedObjectProperties">
2926     <rim:Name>
2927         <rim:LocalizedString
2928 value="label.FindImmediateInheritedObjectProperties"/>
2929     </rim:Name>
2930     <rim:Description>
2931         <rim:LocalizedString
2932 value="label.FindImmediateInheritedObjectProperties.desc"/>
2933     </rim:Description>
2934     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2935 regrep:QueryLanguage:SQL-92">
2936         SELECT A2.*
2937         FROM Association A, Name_N, ClassificationNode C1,
2938 ClassificationNode C2, Association A2
2939         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2940 regrep:AssociationType:SubClassOf' AND
2941         C1.id = N.parent AND
2942         N.value LIKE '$className' AND
2943         A.sourceObject = C1.id AND
2944         A.targetObject = C2.id AND
2945         A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
2946 regrep:AssociationType:ObjectProperty' AND
2947         A2.sourceObject=C2.id
2948     </rim:QueryExpression>

```

2949 </rim:AdhocQuery>

2950 **Adhoc Query retrieving all of the properties of a given classification node including the ones**
2951 **inherited from its immediate super classes**

2952 7.3.13 AllInheritedObjectProperties Discovery Query

2953 Since recursion is not supported by SQL-92, the stored procedure for this query is presented in this
2954 section.

```
2955 CREATE PROCEDURE findAllInheritedObjectProperties
2956     @className varchar(50)
2957 AS
2958 WITH Parents(superClassID) AS (
2959     SELECT C2.id
2960     FROM Association A, Name_ N, ClassificationNode C1,
2961     ClassificationNode C2
2962     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2963     regrep:AssociationType:SubClassOf' AND
2964     C1.id = N.parent AND N.value LIKE @className AND
2965     A.sourceObject = C1.id AND A.targetObject = C2.id
2966
2967     UNION ALL
2968     SELECT A.targetObject
2969     FROM Association A JOIN Parents P
2970     ON P.superClassID = A.sourceObject
2971     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2972     regrep:AssociationType:SubClassOf'
2973 ) SELECT A.id
2974     FROM Association A, Parents P
2975     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2976     regrep:AssociationType:ObjectProperty' AND
2977     A.sourceObject=P.superClassID
2978
2979 UNION
2980
2981 SELECT A.id
2982 FROM Name_ N, ClassificationNode C, Association A
2983 WHERE C.id = N.parent AND N.value LIKE @className AND
2984     A.sourceObject=C.id AND A.associationType LIKE
2985     'urn:oasis:names:tc:ebxml-regrep:AssociationType:ObjectProperty'
2986 GO
```

2985 **Recursive stored procedure for MS SQL Server 2005 database retrieving all inherited Object**
2986 **Properties of a given classification node**

2987 7.3.14 DatatypeProperties Discovery Query

```
2988 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2989     regrep:query:FindDatatypeProperties" id="urn:oasis:names:tc:ebxml-
2990     regrep:query:FindDatatypeProperties">
2991     <rim:Name>
2992         <rim:LocalizedString value="label.FindDatatypeProperties"/>
2993     </rim:Name>
2994     <rim:Description>
2995         <rim:LocalizedString
2996     value="label.FindDatatypeProperties.desc"/>
2997     </rim:Description>
2998     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2999     regrep:QueryLanguage:SQL-92">
3000         SELECT A.*
3001         FROM Association A, Name_ N, ClassificationNode C
3002         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3003     regrep:AssociationType:DatatypeProperty' AND
3004         C.id = N.parent AND
3005         N.value LIKE '$className' AND
3006         A.sourceObject = C.id
3007     </rim:QueryExpression>
3008 </rim:AdhocQuery>
```

3009

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

3010 7.3.15 AllInheritedDatatypeProperties Discovery Query

3011 Since recursion is not supported by SQL-92, the stored procedure for this query is presented in this
3012 section.

```
3013 CREATE PROCEDURE findAllInheritedDatatypeProperties
3014     @className varchar(50)
3015 AS
3016 WITH Parents(superClassID) AS (
3017     SELECT C2.id
3018     FROM Association A, Name_ N, ClassificationNode C1,
3019     ClassificationNode C2
3020     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3021     regrep:AssociationType:SubClassOf' AND
3022     C1.id = N.parent AND N.value LIKE @className AND
3023     A.sourceObject = C1.id AND A.targetObject = C2.id
3024 UNION ALL
3025     SELECT A.targetObject
3026     FROM Association A JOIN Parents P
3027     ON P.superClassID = A.sourceObject
3028     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3029     regrep:AssociationType:SubClassOf'
3030 ) SELECT A.id
3031     FROM Association A, Parents P
3032     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3033     regrep:AssociationType:DatatypeProperty' AND
3034     A.sourceObject=P.superClassID
3035 UNION
3036
3037 SELECT A.id
3038 FROM Name_ N, ClassificationNode C, Association A
3039 WHERE C.id = N.parent AND N.value LIKE @className AND
3040     A.sourceObject=C.id AND A.associationType LIKE
3041     'urn:oasis:names:tc:ebxml-regrep:AssociationType:DatatypeProperty'
3042 GO
```

3043 **Recursive stored procedure for MS SQL Server 2005 database retrieving all inherited Datatype**
3044 **Properties of a given classification node**

3045 7.3.16 TransitiveRelationships Discovery Query

```
3046 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3047     regrep:query:FindTransitiveRelationships" id="urn:oasis:names:tc:ebxml-
3048     regrep:query:FindTransitiveRelationships">
3049     <rim:Name>
3050         <rim:LocalizedString
3051     value="label.FindTransitiveRelationships"/>
3052     </rim:Name>
3053     <rim:Description>
3054         <rim:LocalizedString
3055     value="label.FindTransitiveRelationships.desc"/>
3056     </rim:Description>
3057     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3058     regrep:QueryLanguage:SQL-92">
3059         SELECT C.*
3060         FROM ClassificationNode C, Association A1, Association A2,
3061     Name_ N1, Name_ N2, Name_ N3
3062         WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
3063     regrep:AssociationType:TransitiveProperty' AND
3064         A1.id = N1.parent AND
3065         N1.value LIKE '$propertyName' AND
3066         A1.sourceObject = N3.parent AND
3067         N3.value LIKE '$className' AND
3068         A2.sourceObject = A1.targetObject AND
3069         A2.id = N2.parent AND
```

```

3070         N2.value LIKE '$propertyName' AND
3071         A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3072 regrep:AssociationType:TransitiveProperty' AND
3073         A2.targetObject = C.id
3074         <!-- UNION
3075         SELECT C.*
3076         FROM ClassificationNode C, Association A1, Name_ N1, Name_
3077 N3
3078         WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
3079 regrep:AssociationType:TransitiveProperty' AND
3080         A1.id = N1.parent AND
3081         N1.value LIKE '$propertyName' AND
3082         A1.sourceObject = N3.parent AND
3083         N3.value LIKE '$className' AND
3084         A1.targetObject = C.id -->
3085     </rim:QueryExpression>
3086 </rim:AdhocQuery>

```

3087 **Adhoc Query retrieving the objects in transitive relationship with a given object**

3088 7.3.17 TargetObjects Discovery Query

```

3089 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3090 regrep:query:FindTargetObjects" id="urn:oasis:names:tc:ebxml-
3091 regrep:query:FindTargetObjects">
3092     <rim:Name>
3093         <rim:LocalizedString value="label.FindTargetObjects"/>
3094     </rim:Name>
3095     <rim:Description>
3096         <rim:LocalizedString value="label.FindTargetObjects.desc"/>
3097     </rim:Description>
3098     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3099 regrep:QueryLanguage:SQL-92">
3100         SELECT C2.*
3101         FROM ClassificationNode C2, Association A, Name_ N, Name_
3102 N2, ClassificationNode C1
3103         WHERE A.id=N2.parent AND
3104         N2.value LIKE '$propertyName' AND
3105         C1.id = N.parent AND
3106         N.value LIKE '$className' AND
3107         A.sourceObject = C1.id AND
3108         A.targetObject = C2.id
3109     </rim:QueryExpression>
3110 </rim:AdhocQuery>

```

3111 **Adhoc Query retrieving the Target Objects from the Registry, given a Source Object and an**
3112 **Association**

3113 7.3.18 TargetObjectsInverseOf Discovery Query

```

3114 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3115 regrep:query:FindTOinverseOf" id="urn:oasis:names:tc:ebxml-
3116 regrep:query:FindTOinverseOf">
3117     <rim:Name>
3118         <rim:LocalizedString value="label.FindTOinverseOf"/>
3119     </rim:Name>
3120     <rim:Description>
3121         <rim:LocalizedString value="label.FindTOinverseOf.desc"/>
3122     </rim:Description>
3123     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3124 regrep:QueryLanguage:SQL-92">
3125         SELECT C2.*
3126         FROM ClassificationNode C2, Association A1, Association A2,
3127 Association A3, Name_ N, Name_ N2, ClassificationNode C1
3128         WHERE A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3129 regrep:AssociationType:InverseOf' AND
3130         A1.id = N.parent AND

```

```

3131         N.value LIKE '$propertyName' AND
3132         A2.sourceObject = A1.id AND
3133         A2.targetObject = A3.id AND
3134         C1.id = N2.parent AND
3135         N2.value LIKE '$className' AND
3136         A3.targetObject = C1.id AND
3137         A3.sourceObject = C2.id
3138     </rim:QueryExpression>
3139 </rim:AdhocQuery>

```

3140 **Adhoc query retrieving the Source Objects of an Association which is in "inverseOf"**
3141 **relationship to this Association**

3142 7.3.19 InverseRanges Discovery Query

```

3143 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3144 regrep:query:FindInverseRanges" id="urn:oasis:names:tc:ebxml-
3145 regrep:query:FindInverseRanges">
3146     <rim:Name>
3147         <rim:LocalizedString value="label.FindInverseRanges"/>
3148     </rim:Name>
3149     <rim:Description>
3150         <rim:LocalizedString value="label.FindInverseRanges.desc"/>
3151     </rim:Description>
3152     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3153 regrep:QueryLanguage:SQL-92">
3154         <!-- SELECT C2.*
3155             FROM Association A, Name_ N, Name_ N2, ClassificationNode
3156 C1, ClassificationNode C2
3157             WHERE A.id=N2.parent AND
3158                 N2.value LIKE '$propertyName' AND
3159                 C1.id = N.parent AND
3160                 N.value LIKE '$className' AND
3161                 A.sourceObject = C1.id AND
3162                 A.targetObject = C2.id
3163             UNION -->
3164             SELECT C2.*
3165             FROM ClassificationNode C2, Association A1, Association A2,
3166 Association A3, Name_ N, NAME_ N2, ClassificationNode C1
3167             WHERE A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3168 regrep:AssociationType:InverseOf' AND
3169                 A1.id = N.parent AND
3170                 N.value LIKE '$propertyName' AND
3171                 A2.sourceObject = A1.id AND
3172                 A2.targetObject = A3.id AND
3173                 C1.id = N2.parent AND
3174                 N2.value LIKE '$className' AND
3175                 A1.sourceObject = C1.id AND
3176                 A3.sourceObject = C2.id
3177     </rim:QueryExpression>
3178 </rim:AdhocQuery>

```

3179 **Adhoc Query Retrieving both the Target Objects of a given Association and the Source**
3180 **Objects of an Association which is in "inverseOf" relationship to this Association**

3181 7.3.20 SymmetricProperties Discovery Query

```

3182 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3183 regrep:query:FindSymmetricProperties" id="urn:oasis:names:tc:ebxml-
3184 regrep:query:FindSymmetricProperties">
3185     <rim:Name>
3186         <rim:LocalizedString value="label.FindSymmetricProperties"/>
3187     </rim:Name>
3188     <rim:Description>
3189         <rim:LocalizedString
3190 value="label.FindSymmetricProperties.desc"/>
3191     </rim:Description>

```

```

3192     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3193     regrep:QueryLanguage:SQL-92">
3194         SELECT A.*
3195         FROM Association A, Name_N, ClassificationNode C
3196         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3197     regrep:AssociationType:SymmetricProperty' AND
3198         C.id = N.parent AND
3199         N.value LIKE '$className' AND
3200         A.sourceObject = C.id
3201     </rim:QueryExpression>
3202 </rim:AdhocQuery>

```

3203 **Adhoc Query retrieving all the Symmetric properties of a given classification node**

3204 7.3.21 FunctionalProperties Discovery Query

```

3205 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3206     regrep:query:FindFunctionalProperties" id="urn:oasis:names:tc:ebxml-
3207     regrep:query:FindFunctionalProperties">
3208     <rim:Name>
3209         <rim:LocalizedString
3210     value="label.FindFunctionalProperties"/>
3211     </rim:Name>
3212     <rim:Description>
3213         <rim:LocalizedString
3214     value="label.FindFunctionalProperties.desc"/>
3215     </rim:Description>
3216     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3217     regrep:QueryLanguage:SQL-92">
3218         SELECT A.*
3219         FROM Association A, Name_N, ClassificationNode C
3220         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3221     regrep:AssociationType:FunctionalProperty' AND
3222         C.id = N.parent AND
3223         N.value LIKE '$className' AND
3224         A.sourceObject = C.id
3225     </rim:QueryExpression>
3226 </rim:AdhocQuery>

```

3227 **Adhoc Query retrieving all the Functional properties of a given classification node**

3228 7.3.22 InverseFunctionalProperties Discovery Query

```

3229 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3230     regrep:query:FindInverseFunctionalProperties"
3231     id="urn:oasis:names:tc:ebxml-
3232     regrep:query:FindInverseFunctionalProperties">
3233     <rim:Name>
3234         <rim:LocalizedString
3235     value="label.FindInverseFunctionalProperties"/>
3236     </rim:Name>
3237     <rim:Description>
3238         <rim:LocalizedString
3239     value="label.FindInverseFunctionalProperties.desc"/>
3240     </rim:Description>
3241     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3242     regrep:QueryLanguage:SQL-92">
3243         SELECT A.*
3244         FROM Association A, Name_N, ClassificationNode C
3245         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3246     regrep:AssociationType:InverseFunctionalProperty' AND
3247         C.id = N.parent AND
3248         N.value LIKE '$className' AND
3249         A.sourceObject = C.id
3250     </rim:QueryExpression>
3251 </rim:AdhocQuery>

```

3252 **Adhoc Query retrieving all the Inverse Functional properties of a given classification node**

7.3.23 Instances Discovery Query Discovery Query

```

3254 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-regrep:query:FindInstances"
3255 id="urn:oasis:names:tc:ebxml-regrep:query:FindInstances">
3256   <rim:Name>
3257     <rim:LocalizedString value="label.FindInstances"/>
3258   </rim:Name>
3259   <rim:Description>
3260     <rim:LocalizedString value="label.FindInstances.desc"/>
3261   </rim:Description>
3262   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3263 regrep:QueryLanguage:SQL-92">
3264     <!-- SELECT S.* FROM Service S, (
3265       SELECT A.targetObject AS id
3266       FROM RegistryPackage R, Association A
3267       WHERE R.id=A.sourceObject AND
3268       A.associationType = 'urn:oasis:names:tc:ebxml-
3269 regrep:AssociationType:HasMember' AND
3270       R.id IN (
3271         SELECT A.targetObject
3272         FROM Association A, Name_ N, ClassificationNode C
3273         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3274 regrep:AssociationType:IntersectionOf' AND
3275         C.id = N.parent AND
3276         N.value LIKE '$className' AND
3277         A.sourceObject = C.id
3278       )
3279       ) AS T1, (
3280       SELECT A.targetObject AS id
3281       FROM RegistryPackage R, Association A
3282       WHERE R.id=A.sourceObject AND
3283       A.associationType = 'urn:oasis:names:tc:ebxml-
3284 regrep:AssociationType:HasMember' AND
3285       R.id IN (
3286         SELECT A.targetObject
3287         FROM Association A, Name_ N, ClassificationNode C
3288         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3289 regrep:AssociationType:IntersectionOf' AND
3290         C.id = N.parent AND
3291         N.value LIKE '$className' AND
3292         A.sourceObject = C.id
3293       )
3294       ) AS T2
3295       WHERE S.id IN (
3296         SELECT classifiedObject
3297         FROM Classification
3298         WHERE classificationNode=T1.id
3299         INTERSECT
3300         SELECT classifiedObject
3301         FROM Classification
3302         WHERE classificationNode=T2.id
3303         ) AND T1.id!=T2.id
3304       UNION -->
3305       SELECT S.*
3306       FROM Service S, Classification C, ClassificationNode CN,
3307       Name_ N
3308       WHERE S.id = C.classifiedObject AND
3309       C.classificationNode = CN.id AND
3310       N.value LIKE '$className' AND
3311       N.parent = CN.id
3312   </rim:QueryExpression>
3313 </rim:AdhocQuery>

```

Adhoc Query Retrieving the instances of intersected classes

3315 8 OWL Profile References

3316 8.1 Normative References

- 3317 [Brickley, Guha] Brickley, D., Guha, R.V., RDF Vocabulary Description Language 1.0: RDF Schema
3318 W3C Recommendation 10 February 2004
3319 <http://www.w3.org/TR/rdf-schema/>
3320
- 3321 [DAML+OIL] <http://www.daml.org/>
3322 [ebRIM] ebXML Registry Information Model version 3.0
3323 <http://docs.oasis-open.org/regrep/regrep-rim/v3.0/regrep-rim-3.0-os.pdf>
3324
- 3325 [ebRS] ebXML Registry Services Specification version 3.0
3326 <http://docs.oasis-open.org/regrep/regrep-rs/v3.0/regrep-rs-3.0-os.pdf>
- 3327 [UML] Unified Modeling Language version 1.5
3328 <http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.pdf>
- 3329 [ebRR-DPT] Deployment Profile Template For ebXML Registry 3.0 OASIS Specifications V_0.1.2
- 3330 [ebMS-DPT] Deployment Profile Template For OASIS Specification ebXML Message Service 2.0
- 3331 [OWL] Web Ontology Language (OWL), <http://www.w3.org/2004/OWL/>
- 3332 [RDF] Resource Description Framework, <http://www.w3.org/RDF/>
- 3333 [SQL 92] SQL ISO/IEC 9075:1992 Information technology - Database languages - SQL.
- 3334 [SQL 99] ISO/IEC 9075:1999(E) Information technology - Database languages - SQL.
- 3335 [WSDL] WSDL Specification
3336 <http://www.w3.org/TR/wsdl>

3337 8.2 Informative References

- 3338 [Dogac, et. al.] Dogac A., Kabak Y., Laleci G. C. Mattocks, F. Najmi, J. Pollock
3339 Enhancing ebXML Registries to Make them OWL Aware
3340 Distributed and Parallel Databases Journal, Springer-Verlag, Vol. 18, No. 1, July 2005, pp. 9-36.
3341
- 3342 [IMPL] ebXML Registry 3.0 Implementations
3343 freebXML Registry: A royalty free, open source ebXML Registry Implementation
3344 <http://ebxmlrr.sourceforge.net>
3345
- 3346 [LeeHendler]
3347 Berners-Lee, T., Hendler, J., Lassila, O., "The Semantic Web", Scientific American, May 2001.
3348
- 3349 [McGuinness, Harmelen] OWL Web Ontology Language Overview,

- 3350 W3C Recommendation 10 February 2004, <http://www.w3.org/TR/owl-features/>
3351
3352 [StaabStuder] Staab, S., Studer, R., Handbook on Ontologies, Springer, 2004.