



ebXML Registry Profile for Web Ontology

Language (OWL)

Version 1.0 Draft 5

Draft OASIS Profile, April 19,2006

Document identifier:

regrep-owl-profile-1.0-draft 5

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
Evan Wallace	NIST, USA
Dave RR Webber	XML Consultant, USA

Abstract:

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

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
1.3 Recommended Enhancements.....	11
2 OWL Overview.....	12
2.1 OWL Lite Constructs.....	12
2.1.1 RDF Schema Features.....	12
2.1.2 (In)Equality.....	12
2.1.3 Property Characteristics	12
2.1.4 Property Restrictions.....	13
2.1.5 Restricted Cardinality.....	13
2.1.6 Class Intersection.....	13
2.1.7 Versioning.....	13
2.1.8 Annotation Properties	13
2.1.9 Datatypes	13
2.2 OWL DL Constructs.....	13
2.2.1 Class Axioms.....	13
2.2.2 Boolean Combinations of Class Expressions	14
2.2.3 Arbitrary Cardinality	14
2.2.4 Filler Information.....	14
3 ebXML Registry Overview.....	15
3.1 Overview of [ebRIM].....	15
3.1.1 RegistryObject.....	16
3.1.2 Object Identification.....	16
3.1.3 Object Naming and Description.....	17
3.1.4 Object Attributes.....	17
3.1.4.1 Slot Attributes.....	17
3.1.5 Object Classification.....	18
3.1.6 Object Association.....	18
3.1.7 Object References To Web Content.....	19
3.1.8 Object Packaging.....	19
3.1.9 ExtrinsicObject	20
3.1.10 Service Description.....	20
3.2 Overview of [ebRS].....	20
4 Representing OWL Lite Constructs in ebRIM	21
4.1 Representing RDF Schema Features in ebRIM.....	21
4.1.1 owl:Class → rim:ClassificationNode.....	21
4.1.2 rdf:Property → rim:Association Type Property.....	21
4.1.3 rdfs:subPropertyOf → rim:Association Type subPropertyOf.....	22
4.1.4 rdfs:subClassOf → rim:Association Type subClassOf.....	22
4.1.5 owl:Individual → rim:ExtrinsicObject.....	23
4.2 Representing OWL (In)Equality Constructs in ebXML RIM.....	23
4.2.1 owl:equivalentClass, owl:equivalentProperty → rim:Association Type EquivalentTo	23

78	4.2.2 owl:sameAs → rim:Association Type sameAs.....	23
79	4.2.3 owl:differentFrom → rim:Association Type differentFrom.....	24
80	4.2.4 owl:AllDifferent.....	24
81	4.3 Representing OWL Property Characteristics in ebRIM.....	24
82	4.3.1 owl:ObjectProperty → rim:Association Type objectProperty.....	24
83	4.3.2 owl:DatatypeProperty → rim:Association Type DatatypeProperty.....	25
84	4.3.3 owl:TransitiveProperty → rim:Association Type transitiveProperty.....	25
85	4.3.4 owl:inverseOf → rim:Association Type inverseOf.....	26
86	4.3.5 owl:SymmetricProperty→ rim:Association Type SymmetricProperty.....	26
87	4.3.6 owl:FunctionalProperty→ rim:Association Type FunctionalProperty.....	26
88	4.3.7 owl:InverseFunctionalProperty→ rim:Association Type InverseFunctionalProperty.....	27
89	4.4 OWL Property Restrictions in ebXML RIM.....	27
90	4.5 Representing OWL Restricted Cardinality in ebXML RIM.....	28
91	4.5.1 owl:minCardinality (only 0 or 1).....	28
92	4.5.2 owl:maxCardinality (only 0 or 1).....	29
93	4.5.3 owl:cardinality (only 0 or 1).....	30
94	4.6 Representing OWL Class Intersection in ebXML RIM.....	30
95	4.7 Representing OWL Versioning in ebXML RIM.....	31
96	4.7.1 owl:versionInfo, owl:priorVersion.....	31
97	4.8 Representing OWL Annotation Properties in ebXML RIM.....	32
98	4.8.1 rdfs:label.....	32
99	4.8.2 rdfs:comment.....	32
100	4.8.3 rdfs:seeAlso.....	32
101	4.9 OWL Datatypes in ebXML RIM.....	33
102	5 Cataloging Service Profile.....	34
103	5.1 Invocation Control File.....	34
104	5.2 Input Metadata.....	34
105	5.3 Input Content.....	34
106	5.4 Output Metadata.....	35
107	5.4.1 owl:Class → rim:ClassificationNode.....	35
108	5.4.2 rdf:Property → rim:Association Type Property.....	35
109	5.4.3 rdfs:subPropertyOf → rim:Association Type subPropertyOf.....	35
110	5.4.4 rdfs:subClassOf → rim:Association Type subClassOf.....	35
111	5.4.5 owl:Individual → rim:ExtrinsicObject.....	35
112	5.4.6 owl:equivalentClass, owl:equivalentProperty → rim:Association Type EquivalentTo	35
113	5.4.7 owl:sameAs → rim:Association Type sameAs	35
114	5.4.8 owl:differentFrom → rim:Association Type differentFrom.....	35
115	5.4.9 owl:AllDifferent → rim:RegistryPackage.....	35
116	5.4.10 owl:ObjectProperty → rim:Association Type objectProperty.....	36
117	5.4.11 owl:DatatypeProperty → rim:Association Type DatatypeProperty.....	36
118	5.4.12 owl:TransitiveProperty → rim:Association Type transitiveProperty.....	36
119	5.4.13 owl:inverseOf → rim:Association Type inverseOf.....	36
120	5.4.14 owl:SymmetricProperty→ rim:Association Type SymetricProperty.....	36
121	5.4.15 owl:FunctionalProperty→ rim:Association Type FunctionalProperty.....	36
122	5.4.16 owl:InverseFunctionalProperty→ rim:Association Type InverseFunctionalProperty.....	36
123	5.4.17 owl:minCardinality (only 0 or 1).....	36

124	5.4.18 owl:maxCardinality (only 0 or 1).....	37
125	5.4.19 owl:cardinality.....	37
126	5.4.20 owl:intersectionOf.....	37
127	5.4.21 rdfs:label.....	37
128	5.4.22 rdfs:comment.....	37
129	5.4.23 rdfs:seeAlso.....	37
130	6 Discovery Profile.....	38
131	6.1 All SuperProperties Discovery Query.....	38
132	6.1.1 Parameter \$propertyName.....	38
133	6.1.2 Example of All SuperProperties Discovery Query.....	38
134	6.2 Immediate SuperClass Discovery Query.....	39
135	6.2.1 Parameter \$className.....	39
136	6.2.2 Example of Immediate SuperClass Discovery Query.....	39
137	6.3 Immediate SubClass Discovery Query.....	40
138	6.3.1 Parameter \$className.....	40
139	6.3.2 Example of Immediate SubClass Discovery Query.....	40
140	6.4 All SuperClasses Discovery Query.....	40
141	6.4.1 Parameter \$className.....	41
142	6.4.2 Example of All SuperClasses Discovery Query.....	41
143	6.5 All SubClasses Discovery Query.....	41
144	6.5.1 Parameter \$className.....	41
145	6.5.2 Example of All SubClasses Discovery Query.....	42
146	6.6 EquivalentClasses Discovery Query.....	42
147	6.6.1 Parameter \$className.....	42
148	6.6.2 Example of EquivalentClasses Discovery Query.....	42
149	6.7 EquivalentProperties Discovery Query.....	43
150	6.7.1 Parameter \$propertyName.....	43
151	6.7.2 Example of EquivalentProperties Discovery Query.....	43
152	6.8 SameExtrinsicObjects Discovery Query.....	44
153	6.8.1 Parameter \$extrinsicObjectName.....	44
154	6.8.2 Example of SameExtrinsicObjects Discovery Query.....	44
155	6.9 DifferentExtrinsicObjects Discovery Query.....	44
156	6.9.1 Parameter \$extrinsicObjectName.....	45
157	6.9.2 Example of DifferentExtrinsicObjects Discovery Query.....	45
158	6.10 AllDifferentRegistryObject Discovery Query.....	45
159	6.10.1 Parameter \$registryObjectName.....	45
160	6.10.2 Example of AllDifferentRegistryObjects Discovery Query.....	45
161	6.11 ObjectProperties Discovery Query.....	46
162	6.11.1 Parameter \$className.....	46
163	6.11.2 Example of ObjectProperties Discovery Query.....	46
164	6.12 ImmediateInheritedObjectProperties Discovery Query.....	47
165	6.12.1 Parameter \$className.....	47
166	6.12.2 Example of ImmediateInheritedObjectProperties Discovery Query.....	47
167	6.13 AllInheritedObjectProperties Discovery Query.....	48
168	6.13.1 Parameter \$className.....	48
169	6.13.2 Example of AllInheritedObjectProperties Discovery Query.....	48

170	6.14 DatatypeProperties Discovery Query.....	49
171	6.14.1 Parameter \$className.....	49
172	6.14.2 Example of DatatypeProperties Discovery Query.....	49
173	6.15 AllInheritedDatatypeProperties Discovery Query.....	49
174	6.15.1 Parameter \$className.....	50
175	6.15.2 Example of AllInheritedDatatypeProperties Discovery Query.....	50
176	6.16 TransitiveRelationships Discovery Query.....	50
177	6.16.1 Parameter \$className.....	50
178	6.16.2 Parameter \$propertyName.....	50
179	6.16.3 Example of TransitiveRelationships Discovery Query.....	51
180	6.17 TargetObjects Discovery Query.....	51
181	6.17.1 Parameter \$className.....	51
182	6.17.2 Parameter \$propertyName.....	51
183	6.17.3 Example of TargetObjects Discovery Query.....	52
184	6.18 TargetObjectsInverseOf Discovery Query.....	52
185	6.18.1 Parameter \$className.....	52
186	6.18.2 Parameter \$propertyName.....	52
187	6.18.3 Example of TargetObjectsInverseOf Discovery Query.....	53
188	6.19 InverseRanges Discovery Query.....	53
189	6.19.1 Parameter \$className.....	53
190	6.19.2 Parameter \$propertyName.....	53
191	6.19.3 Example of InverseRanges Discovery Query.....	54
192	6.20 SymmetricProperties Discovery Query.....	55
193	6.20.1 Parameter \$className.....	55
194	6.20.2 Example of SymmetricProperties Discovery Query.....	55
195	6.21 FunctionalProperties Discovery Query.....	55
196	6.21.1 Parameter \$className.....	55
197	6.21.2 Example of FunctionalProperties Discovery Query.....	55
198	6.22 InverseFunctionalProperties Discovery Query.....	56
199	6.22.1 Parameter \$className.....	56
200	6.22.2 Example of InverseFunctionalProperties Discovery Query.....	56
201	6.23 Instances Discovery Query.....	57
202	6.23.1 Parameter \$className.....	57
203	6.23.2 Example of Instances Discovery Query.....	57
204	7 Canonical Metadata Definitions.....	59
205	7.1 ObjectType Extensions.....	59
206	7.2 AssociationType Extensions.....	59
207	7.3 Canonical Queries.....	61
208	7.3.1 All SuperProperties Discovery Query.....	61
209	7.3.2 Immediate SuperClass Discovery Query.....	62
210	7.3.3 Immediate SubClass Discovery Query.....	62
211	7.3.4 All SuperClasses Discovery Query.....	63
212	7.3.5 All SubClasses Discovery Query.....	63
213	7.3.6 EquivalentClasses Discovery Query.....	64
214	7.3.7 EquivalentProperties Discovery Query.....	64
215	7.3.8 SameExtrinsicObjects Discovery Query.....	64

216	7.3.9 DifferentExtrinsicObjects Discovery Query.....	65
217	7.3.10 AllDifferentRegistryObject Discovery Query.....	65
218	7.3.11 ObjectProperties Discovery Query.....	66
219	7.3.12 ImmediateInheritedObjectProperties Discovery Query.....	66
220	7.3.13 AllInheritedObjectProperties Discovery Query.....	67
221	7.3.14 DatatypeProperties Discovery Query.....	67
222	7.3.15 AllInheritedDatatypeProperties Discovery Query.....	68
223	7.3.16 TransitiveRelationships Discovery Query.....	68
224	7.3.17 TargetObjects Discovery Query.....	69
225	7.3.18 TargetObjectsInverseOf Discovery Query.....	69
226	7.3.19 InverseRanges Discovery Query.....	70
227	7.3.20 SymmetricProperties Discovery Query.....	71
228	7.3.21 FunctionalProperties Discovery Query.....	71
229	7.3.22 InverseFunctionalProperties Discovery Query.....	71
230	7.3.23 Instances Discovery Query Discovery Query.....	72
231	8 OWL Profile References.....	74
232	8.1 Normative References.....	74
233	8.2 Informative References.....	74
234		

Illustration Index

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

235

Index of Tables

236

1 Introduction

237

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

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

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

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

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

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

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

271 The document is organized as follows:

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

1.1 Terminology

280

281 The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT,
282 RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in IETF RFC
283 2119 [RFC2119].

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

287 1.2 Conventions

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

- 290 • UML Diagrams

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

- 293 • Identifier Placeholders

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

299

300

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

301 1.3 Recommended Enhancements

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

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

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

2 OWL Overview

311

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

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

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

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

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

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

2.1 OWL Lite Constructs

330

2.1.1 RDF Schema Features

331

- 332 • Class (Thing, Nothing)
- 333 • rdfs:subClassOf
- 334 • rdf:Property
- 335 • rdfs:subPropertyOf
- 336 • rdfs:domain
- 337 • rdfs:range
- 338 • Individual

2.1.2 (In)Equality

339

- 340 • equivalentClass
- 341 • equivalentProperty
- 342 • sameAs
- 343 • differentFrom
- 344 • AllDifferent
- 345 • distinctMembers

2.1.3 Property Characteristics

346

- 347 • ObjectProperty
- 348 • DatatypeProperty
- 349 • inverseOf
- 350 • TransitiveProperty

- 351 • SymmetricProperty
- 352 • FunctionalProperty
- 353 • InverseFunctionalProperty

354 2.1.4 Property Restrictions

- 355 • Restriction
- 356 • onProperty
- 357 • allValuesFrom
- 358 • someValuesFrom

359 2.1.5 Restricted Cardinality

- 360 • minCardinality (only 0 or 1)
- 361 • maxCardinality (only 0 or 1)
- 362 • cardinality (only 0 or 1)

363 2.1.6 Class Intersection

- 364 • intersectionOf

365 2.1.7 Versioning

- 366 • versionInfo
- 367 • priorVersion
- 368 • backwardCompatibleWith
- 369 • incompatibleWith
- 370 • DeprecatedClass
- 371 • DeprecatedProperty

372 2.1.8 Annotation Properties

- 373 • rdfs:label
- 374 • rdfs:comment
- 375 • rdfs:seeAlso
- 376 • rdfs:isDefinedBy
- 377 • AnnotationProperty
- 378 • OntologyProperty

379 2.1.9 Datatypes

- 380 • xsd datatypes

381 2.2 OWL DL Constructs

382 2.2.1 Class Axioms

- 383 • oneOf, dataRange
- 384 • disjointWith
- 385 • equivalentClass (applied to class expressions)

386 • rdfs:subClassOf (applied to class expressions)

387 2.2.2 Boolean Combinations of Class Expressions

- 388 • unionOf
- 389 • complementOf
- 390 • intersectionOf

391 2.2.3 Arbitrary Cardinality

- 392 • minCardinality
- 393 • maxCardinality
- 394 • cardinality

395 2.2.4 Filler Information

- 396 • hasValue

397

398

3 ebXML Registry Overview

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

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

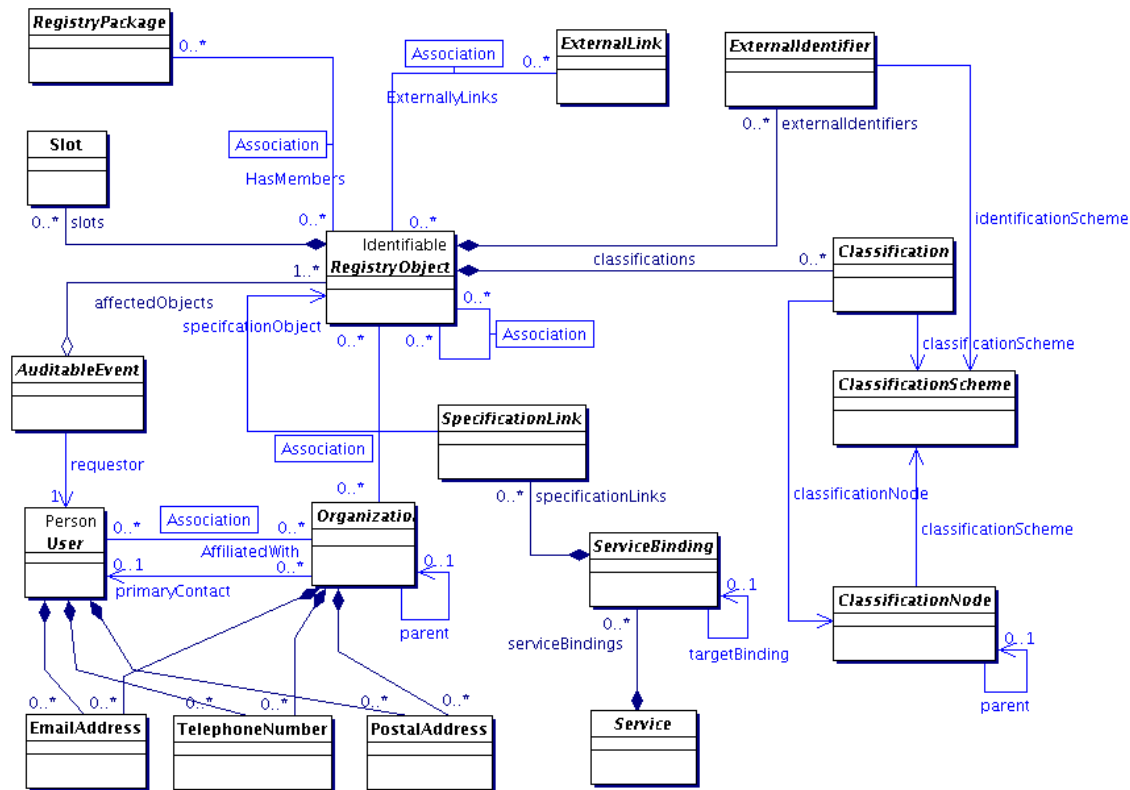
402 The information presented is informative and is not intended to replace the normative information defined
403 by ebXML Registry.

3.1 Overview of [ebRIM]

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

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

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

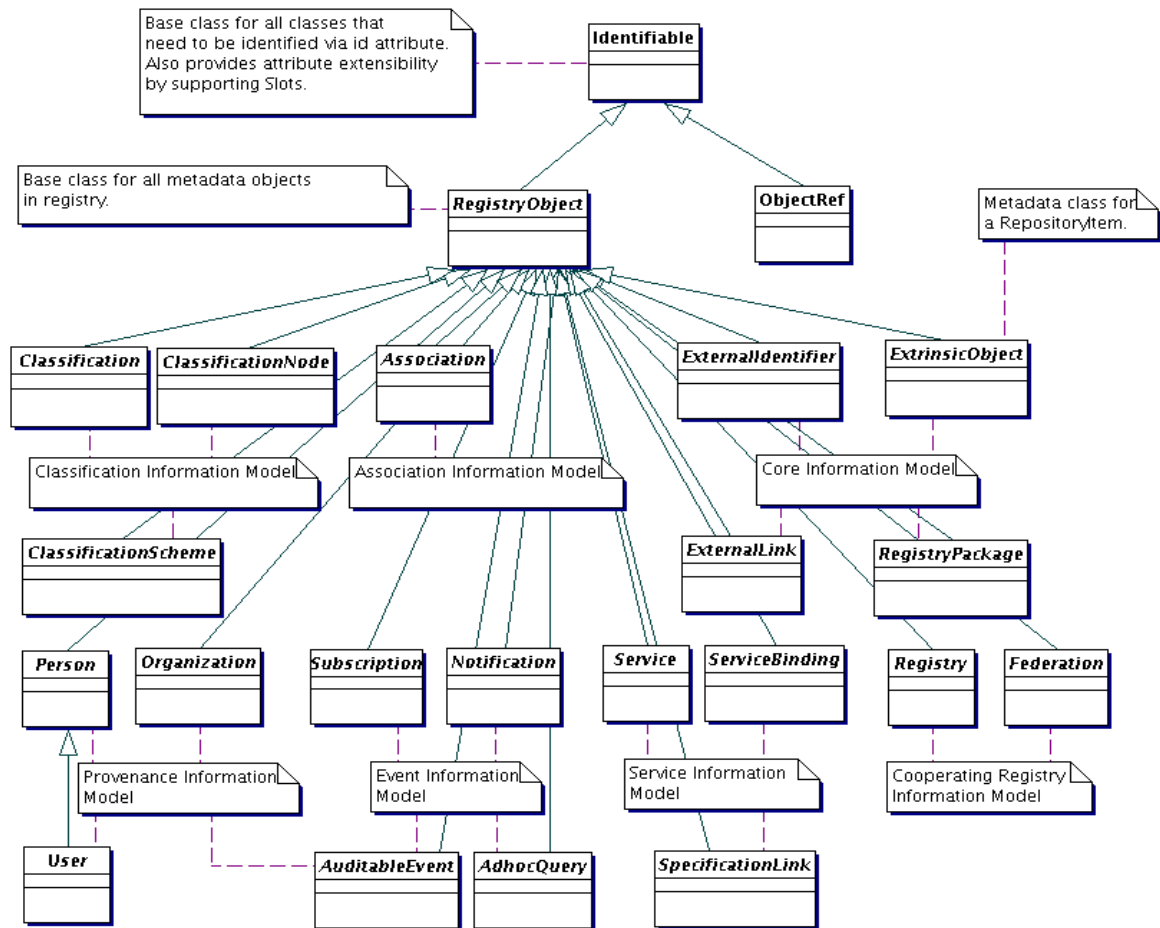


412 **Figure 1: ebXML Registry Information Model, High Level Public View**

413

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

418



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

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

422 3.1.1 RegistryObject

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

426 3.1.2 Object Identification

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

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

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

430
 431 The id attribute value MAY potentially be human friendly but MUST be a unique ID value within the
 432 registry.

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

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

435
 436 Since a RegistryObject MAY have several versions, a logical id (called lid) is also defined which is unique
 437 for different logical objects. However the lid attribute value MUST be the same for all versions of the same

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

440

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

443

Listing 3: Example of lid Attribute

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

446

```
447 <rim:Organization id=${ACME_ORG_ID}  
448   lid="urn:acme:ACMEOrganization">  
449  
450   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}  
451     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}  
452     value="ACME"/>  
453 </rim:ExternalIdentifier>  
454  
455 </rim:Organization>
```

456

Listing 4: Example of ExternalIdentifier

457 3.1.3 Object Naming and Description

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

460

```
461 <rim:Organization id=${ACME_ORG_ID}  
462   lid="urn:acme:ACMEOrganization">  
463  
464   <rim:Name>  
465     <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>  
466   </rim:Name>  
467   <rim:Description>  
468     <rim:LocalizedString value="ACME is a provider of Java software."  
469       xml:lang="en-US"/>  
470   </rim:Description>  
471  
472   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}  
473     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}  
474     value="ACME"/>  
475 </rim:ExternalIdentifier>  
476 </rim:Organization>
```

477

Listing 5: Example of Name and Description

478

479 3.1.4 Object Attributes

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

482 3.1.4.1 Slot Attributes

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

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

489

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

```

492 <rim:Slot name="urn:acme:slot:NASDAQSymbol">
493   <rim:ValueList>
494     <rim:Value>ACME</rim:Value>
495   </rim:ValueList>
496 </rim:Slot>
497
498   <rim:Name>
499     <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
500   </rim:Name>
501   <rim:Description>
502     <rim:LocalizedString value="ACME makes Java. Provider of free Java
503 software."
504     xml:lang="en-US"/>
505   </rim:Description>
506   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
507     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
508     value="ACME"/>
509   </rim:ExternalIdentifier>
510 </rim:Organization>
511

```

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.

```

519 <rim:Organization id=${ACME_ORG_ID}
520   lid="urn:acme:ACMEOrganization">
521   <rim:Slot name="urn:acme:slot:NASDAQSymbol">
522     <rim:ValueList>
523       <rim:Value>ACME</rim:Value>
524     </rim:ValueList>
525   </rim:Slot>
526   <rim:Name>
527     <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
528   </rim:Name>
529   <rim:Description>
530     <rim:LocalizedString value="ACME makes Java. Provider of free Java
531 software." xml:lang="en-US"/>
532   </rim:Description>
533   <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
534     identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
535     value="ACME"/>
536   </rim:ExternalIdentifier>
537
538   <!--Classify Organization as a Software Publisher using NAICS Taxonomy-->
539   <rim:Classification id=${CLASSIFICATION_ID}
540     classificationNode=${NAICS_SOFTWARE_PUBLISHER_NODE_ID}
541     classifiedObject=${ACME_ORG_ID}>
542
543 </rim:Organization>

```

Listing 7: Example of Object Classification

3.1.6 Object Association

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

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

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

554 ml

555 [ebRIM] allows this scheme to be extensible.

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

559

```
560 <rim:Association  
561     id=${ASSOCIATION_ID}  
562     associationType=${CANONICAL_ASSOCIATION_TYPE_OFFERS_SERVICE_ID}  
563     sourceObject=${ACME_ORG_ID}  
564     targetObject=${ACME_SERVICE1_ID}/>
```

565 **Listing 8: Example of Object Association**

566 3.1.7 Object References To Web Content

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

572

```
573 <rim:ExternalLink externalURI="http://www.acme.com"  
574     id=${ACME_WEBSITE_EXTERNAL_ID}>  
575 <rim:Association  
576     id=${EXTERNALLYLINKS_ASSOCIATION_ID}  
577     associationType=${CANONICAL_ASSOCIATION_TYPE_EXTERNALLY_LINKS_ID}  
578     sourceObject=${ACME_WEBSITE_EXTERNAL_ID}  
579     targetObject=${ACME_ORG_ID}/>
```

580 **Listing 9: Example of Reference to Web Content Using ExternalLink**

581 3.1.8 Object Packaging

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

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

589

```
590 <rim:RegistryPackage  
591     id=${ACME_SERVICES_PACKAGE_ID}>  
592     <rim:RegistryObjectList>  
593         <rim:ObjectRef id=${ACME_SERVICE1_ID}>  
594             <rim:RegistryPackage  
595                 id=${ACME_PURCHASING_SERVICES_PACKAGE_ID}>  
596                 <rim:ObjectRef id=${ACME_PURCHASING_SERVICE1_ID}>  
597                 <rim:ObjectRef id=${ACME_PURCHASING_SERVICE2_ID}>  
598             </rim:RegistryPackage>  
599             <rim:RegistryPackage  
600                 id=${ACME_HR_SERVICES_PACKAGE_ID}>  
601                 <rim:ObjectRef id=${ACME_HR_SERVICE1_ID}>  
602                 <rim:ObjectRef id=${ACME_HR_SERVICE2_ID}>  
603             </rim:RegistryPackage>  
604         </rim:RegistryObjectList>  
605     </rim:RegistryPackage>
```

606 **Listing 10: Example of Object Packaging Using RegistryPackages**

607 3.1.9 ExtrinsicObject

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

612 3.1.10 Service Description

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

616 3.2 Overview of [ebRS]

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

4 Representing OWL Lite Constructs in ebRIM

619

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

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

628

```
629 ClassScheme (id, home, lid, objectType, status, versionName, comment_,...);  
630  
631 ClassificationNode(accessControlPolicy, id, lid, home, objectType, code, parent,  
632 path,versionName, comment_...)  
633  
634 Association(accessControlPolicy, id, lid, home, objectType, associationType,  
635 sourceObject, targetObject, isConfirmedBySourceOwner,versionName, comment_  
636 isConfirmedByTargetOwner,...)  
637  
638 Name_(charset, lang, value, parent,...)  
639  
640 Classification (id, objectType, lid, home, classificationNode, versionName,  
641 comment_, classificationScheme, classifiedObject, nodeRepresentation,...);  
642  
643 ExtrinsicObject (id, lid, home, objectType,...)
```

644

ebXML Registry Relations

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

4.1 Representing RDF Schema Features in ebRIM

648

4.1.1 owl:Class → rim:ClassificationNode

649

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

651

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

654

Example owl:Class

655

```
656 <rim:ClassificationNode id='City' code='City'>  
657 </rim:ClassificationNode>
```

658

Example Corresponding ebRIM construct ClassificationNode

4.1.2 rdf:Property → rim:Association Type Property

659

660 A new ebRIM Association Type called "Property" MUST be defined. The domain of an rdf:Property,
661 rdfs:domain, is the sourceObject in this Association Type and the range of an rdf:Property which is
662 rdfs:range, is the targetObject of the Association Type. Consider the following example which defines an
663 rdf:Property instance called "hasAirport" whose domain is "City" and whose range is "Airport" classes:

664

```
665 <rdf:Property rdf:ID="hasAirport">  
666 <rdfs:domain rdf:resource="#City"/>  
667 <rdfs:range rdf:resource="#AirPort"/>  
668 </rdf:Property>
```

669

Example rdf:Property

670
671
672
673
674
675

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

676 **Example: ebRIM construct Association corresponding to rdf:Property**

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

679 4.1.3 rdfs:subPropertyOf → rim:Association Type subPropertyOf

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

683

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

684 **Example rdfs:subPropertyOf**

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

691 4.1.4 rdfs:subClassOf → rim:Association Type subclassOf

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

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

702

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

708 **Example rdfs:subClassOf**

709 To express this semantics through ebXML RIM constructs, "AirReservationServices" ClassificationNode is
710 associated both with the "OWL-S Profile" and "AirServices" ClassificationNodes through the "targetObject"
711 and "sourceObject" attributes of the two instances of the newly created "subclassOf" ebXML Association
712 Type as shown in the following:

713

```
<rim:Association id='subclassOf1' associationType='urn:oasis:names:tc:ebxml-
  regrep:AssociationType:SubClassOf'
  sourceObject= 'AirReservationServices' targetObject='OWL-S Profile' >
</rim:Association>
<rim:Association id='subclassOf2' associationType='urn:oasis:names:tc:ebxml-
  regrep:AssociationType:SubClassOf'
  sourceObject= 'AirReservationServices' targetObject='AirServices' >
</rim:Association>
```

722 Once such a semantics is defined, there is a need to process the objects in the registry according to the

723 semantics implied; that is, given a class, it should be possible to retrieve all of its subclasses and/or all of
724 its super classes. By making the required adhoc queries available in the registry, this need can be readily
725 served as described in Sections 6.2, 6.3, 6.4 and 6.5.

726 4.1.5 owl:Individual → rim:ExtrinsicObject

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

730

```
731 <owl:Class rdf:ID="TravelWebService">  
732 </owl:Class>  
733  
734 <owl:ObjectProperty rdf:ID="paymentMethod">  
735 <rdfs:domain rdf:resource="#TravelWebService"/>  
736 <rdfs:range rdf:resource="#PossiblePaymentMethods"/>  
737 </owl:ObjectProperty >
```

738 Example owl:Class example

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

743

```
744 <TravelWebService rdf:ID="MyTravelWebService">  
745 <paymentMethod> Cash </paymentMethod>  
746 </TravelWebService>
```

747 Example owl:Individual example

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

752 4.2 Representing OWL (In)Equality Constructs in ebXML RIM

753 4.2.1 owl:equivalentClass, owl:equivalentProperty → rim:Association Type 754 EquivalentTo

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

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

762 4.2.2 owl:sameAs → rim:Association Type sameAs

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

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

767

```
768 <rdf:Description rdf:about="#MyAirReservationService">  
769 <owl:sameAs rdf:resource="#THYAirReservationService"/>  
770 </rdf:Description>
```


771 **Example owl:sameAs**

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

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

776 **4.2.3 owl:differentFrom → rim:Association Type differentFrom**

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

781

```
782 <rdf:Description rdf:about="#MyAirReservationService">  
783   <owl:differentFrom rdf:resource="#THYAirReservationService"/>  
784 </rdf:Description>
```

785 **Example owl:differentFrom**

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

790 **4.2.4 owl:AllDifferent**

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

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

```
797 <owl:AllDifferent>  
798   <owl:distinctMembers rdf:parseType="Collection">  
799     <WebService rdf:about="#MyCarService"/>  
800     <WebService rdf:about="#MyFlightService"/>  
801     <WebService rdf:about="#MyHotelService"/>  
802   </owl:distinctMembers>  
803 </owl:AllDifferent>
```

804 **Example owl:AllDifferentFrom**

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

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

810 **4.3 Representing OWL Property Characteristics in ebRIM**

811 **4.3.1 owl:ObjectProperty → rim:Association Type objectProperty**

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

815

```
816 <owl:ObjectProperty rdf:ID="hasAirport">  
817   <rdfs:domain rdf:resource="#City"/>
```



```
818 <rdfs:range rdf:resource="#AirPort"/>
819 </owl:ObjectProperty>
```

Example owl:ObjectProperty

```
821
822 <rim:Association id='hasAirport' associationType='urn:oasis:names:tc:ebxml-
823 regrep:AssociationType:ObjectProperty'
824 sourceObject= 'City' targetObject='Airport' >
825 </rim:Association>
```

Example Corresponding ebRIM construct Association

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

830 4.3.2 owl:DatatypeProperty → rim:Association Type DatatypeProperty

831 Similarly, to represent OWL DatatypeProperty in ebXML, a new Association Type called
832 "DatatypeProperty" MUST be defined. Consider the following example which defines an datatype property
833 "hasPrice" whose domain is the "AirReservationServices" and whose range is "XMLSchema
834 nonNegativeInteger". How OWL XML Schema types are handled in ebXML RIM is described in Section
835 4.9.

```
836 <owl:DatatypeProperty rdf:ID="hasPrice">
837 <rdfs:domain rdf:resource="#AirReservationServices"/>
838 <rdfs:range
839 rdf:resource="http://www.w3.org/2000/10/XMLSchema/nonNegativeInteger"/>
840 </owl:DatatypeProperty>
```

Example owl:DatatypeProperty

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

844 4.3.3 owl:TransitiveProperty → rim:Association Type transitiveProperty

845 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)
846 [McGuinness, Harmelen]. Transitive property is a subproperty of ObjectProperty and MUST be defined as
847 a new Association Type called "transitiveProperty" in ebRIM.

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

```
850
851 <owl:ObjectProperty rdf:ID="succeeds">
852 <rdf:type rdf:resource="&owl;TransitiveProperty" />
853 <rdfs:domain rdf:resource="#TravelWebService" />
854 <rdfs:range rdf:resource="#TravelWebService" />
855 </owl:ObjectProperty>
```

Example owl:TransitiveProperty

857 Assume the following two definitions which declare three Web service instances from TravelWebService
858 class where "MyHotelAvailabilityService" service succeeds "MyAirReservationService" and
859 "MyInsuranceService" succeeds "MyHotelAvailabilityService". Since "succeeds" is a transitive property, it
860 follows that "MyInsuranceService" succeeds "MyAirReservationService" although this fact is not explicitly
861 stated.

```
862
863 <TravelWebService rdf:ID="MyHotelAvailabilityService">
864 <succeeds rdf:resource="#MyAirReservationService" />
865 </TravelWebService>
866
867 <TravelWebService rdf:ID="MyInsuranceService">
868 <succeeds rdf:resource="#MyHotelAvailabilityService" />
869 </TravelWebService>
```

870 **Example owl:TransitiveProperty instances**

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

874 **4.3.4 owl:inverseOf → rim:Association Type inverseOf**

875 In OWL, one property may be stated to be the inverse of another property. If the property P1 is stated to
876 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
877 P1 property [McGuinness, Harmelen].

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

881

```
882 <owl:ObjectProperty rdf:ID="precedes">  
883   <owl:inverseOf rdf:resource="#succeeds" />  
884 </owl:ObjectProperty>
```

885 **Example owl:inverseOf Property**

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

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

896 **4.3.5 owl:SymmetricProperty → rim:Association Type SymmetricProperty**

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

```
901 <owl:Class rdf:ID="WebService">  
902   <rdfs:subClassOf  
903     rdf:resource="http://www.w3.org/2000/01/rdfschema#Resource"/>  
904 </owl:Class>  
905 <owl:SymmetricProperty rdf:ID="complements">  
906   <rdfs:domain rdf:resource="#WebService"/>  
907   <rdfs:range rdf:resource="#WebService"/>  
908 </owl:SymmetricProperty>
```

909 **Example owl:SymmetricProperty**

910 Given that HotelReservationWebService complements AirReservationWebService, it is possible to
911 deduce that AirReservationWebService complements HotelReservationWebService.

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

915 **4.3.6 owl:FunctionalProperty → rim:Association Type FunctionalProperty**

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

```
920 <owl:DatatypeProperty rdf:ID="hasPrice">
921   <rdf:type rdf:resource="&owl;FunctionalProperty" />
922   <rdfs:domain rdf:resource="#AirReservationServices"/>
923   <rdfs:range
924   rdf:resource="http://www.w3.org/2000/10/XMLSchema/nonNegativeInteger"/>
925 </owl:DatatypeProperty>
```

926 **Example owl:FunctionalProperty**

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

930 **4.3.7 owl:InverseFunctionalProperty → rim:Association Type** 931 **InverseFunctionalProperty**

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

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

```
940 <owl:ObjectProperty rdf:ID="finalDestination">
941   <rdf:type rdf:resource="&owl;InverseFunctionalProperty" />
942   <rdfs:domain rdf:resource="#Airport"/>
943   <rdfs:range rdf:resource="#Flight"/>
944 </owl:ObjectProperty>
```

945 **Example owl:InverseFunctionalProperty**

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

949 **4.4 OWL Property Restrictions in ebXML RIM**

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

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

```
959
960 <owl:ObjectProperty rdf:ID="paymentMethod">
961   <rdfs:domain rdf:resource="#TravelWebService"/>
962   <rdfs:range rdf:resource="#PossiblePaymentMethods"/>
963 </owl:ObjectProperty >
```

964 **Example owl:ObjectProperty "paymentMethod"**

```
965
966 <owl:Class rdf:ID="AirReservationServices">
967   <rdfs:subClassOf>
968     <owl:Restriction>
969       <owl:onProperty rdf:resource="#paymentMethod"/>
970       <owl:allValuesFrom rdf:resource="&#CreditCard"/>
971     </owl:Restriction>
```

```
972 </rdfs:subClassOf>
973 </owl:Class>
```

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

975 A new Association Type of “restriction” SHOULD be defined to represent OWL restriction. A slot of this
976 Association Type SHOULD indicate the whether the restriction is “allValuesFrom” or “someValuesFrom”.
977 When such restriction is submitted to the system, the registry MUST create a new Association instance,
978 say, “paymentMethod_1” of AssociationType “ObjectProperty” is created whose sourceObject is
979 “AirReservationServices” and the targetObject is “CreditCard”. “paymentMethod_1” Association instance
980 is related with the “paymentMethod” Association instance by using an instance of the Association Type
981 “restriction” as shown in the following example:

```
982
983 <rim:Association id = "paymentMethod_1"
984     associationType =
985     "urn:oasis:names:tc:ebxml-regrep:AssociationType:ObjectProperty"
986     sourceObject = "AirReservationServices"
987     targetObject = "CreditCard" />
988
989     <rim:Association id = "paymentMethodRestriction"
990         associationType =
991         "urn:oasis:names:tc:ebxml-regrep:AssociationType:restriction"
992         sourceObject = "paymentMethod"
993         targetObject = "paymentMethod_1">
994         <rim:Slot name="restrictionType">
995             <rim:ValueList>
996                 <rim:Value>allValuesFrom</rim:Value>
997             </rim:ValueList>
998         </rim:Slot>
999     </rim:Association>
```

1000 **Example Handling owl:Restriction in ebXML Registry**

1001 Obviously, this serves the purpose of reusing the "paymentMethod" property. Otherwise, a new property
1002 "paymentMethodCC" can be defined between "AirReservationServices" and the "CreditCard" classes as
1003 shown in the following:

```
1004
1005 <owl:ObjectProperty rdf:ID="paymentMethodCC">
1006     <rdfs:domain rdf:resource="#AirReservationServices"/>
1007     <rdfs:range rdf:resource="#CreditCard"/>
1008 </owl:ObjectProperty >
```

1009 **Example owl:ObjectProperty “paymentMethodCC”**

1010 **4.5 Representing OWL Restricted Cardinality in ebXML RIM**

1011 **4.5.1 owl:minCardinality (only 0 or 1)**

1012 In OWL, cardinality is stated on a property with respect to a particular class. If a minCardinality of 1 is
1013 stated on a property with respect to a class, then then any instance of the class will have at least one
1014 value for the restricted property. This restriction is another way of saying that the property is required to
1015 have a value for all instances of the class. In OWL Lite, the only minimum cardinalities allowed are 0 or 1.
1016 A minimum cardinality of zero on a property just states (in the absence of any more specific information)
1017 that the property is optional with respect to a class [McGuinness, Harmelen].

1018 Consider for example the following OWL code which states that each instance of a “WebService” class
1019 must have at least one price:

```
1020 <owl:Class rdf:ID="WebService">
1021     <rdfs:subClassOf>
1022         <owl:Restriction>
1023             <owl:onProperty rdf:resource="#hasPrice"/>
1024             <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">
1025 1 </owl:minCardinality>
1026         </owl:Restriction>
```

```
1027 </rdfs:subClassOf>
1028 </owl:Class>
```

1029 **Example owl:minCardinality**

1030 In ebXML RIM, cardinalities of Association Types MUST be defined by associating a minCardinality slot
1031 with the Association Types as shown in the following example:

1032

```
1033 <rim:Association id = "hasPriceMinCardinalityRestriction"
1034 associationType = "urn:oasis:names:tc:ebxml-
1035 regrep:AssociationType:ObjectProperty" sourceObject = "WebService"
1036 targetObject = "Price">
1037   <rim:Name>
1038     <rim:LocalizedString value = 'hasPrice' />
1039   </rim:Name>
1040   <rim:Slot name="minCardinality">
1041     <rim:ValueList>
1042       <rim:Value>1</rim:Value>
1043     </rim:ValueList>
1044   </rim:Slot>
1045 </rim:Association>
```

1046 **Example Representing owl:minCardinality in ebRIM**

1047 **4.5.2 owl:maxCardinality (only 0 or 1)**

1048 In OWL, cardinality is stated on a property with respect to a particular class. If a maxCardinality of 1 is
1049 stated on a property with respect to a class, then any instance of that class will be related to at most one
1050 individual by that property. A maxCardinality 1 restriction is sometimes called a functional or unique
1051 property. It may be useful to state that certain classes have no values for a particular property. This
1052 situation is represented by a maximum cardinality of zero on the property [McGuinness, Harmelen].

1053 Consider for example the following OWL code which states that each instance of a "WebService" class
1054 can have at most one price:

```
1055 <owl:Class rdf:ID="WebService">
1056   <rdfs:subClassOf>
1057     <owl:Restriction>
1058       <owl:onProperty rdf:resource="#hasPrice"/>
1059       <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
1060 1 </owl:maxCardinality>
1061     </owl:Restriction>
1062   </rdfs:subClassOf>
1063 </owl:Class>
```

1064 **Example owl:maxCardinality**

1065 In ebXML RIM, cardinalities of Association Types MUST be defined by associating a maxCardinality slot
1066 with the Association Types as shown in the following example:

1067

```
1068 <rim:Association id = "hasPriceMaxCardinalityRestriction"
1069 associationType = "urn:oasis:names:tc:ebxml-
1070 regrep:AssociationType:ObjectProperty" sourceObject = "WebService"
1071 targetObject = "Price">
1072   <rim:Name>
1073     <rim:LocalizedString value = 'hasPrice' />
1074   </rim:Name>
1075   <rim:Slot name="maxCardinality">
1076     <rim:ValueList>
1077       <rim:Value>1</rim:Value>
1078     </rim:ValueList>
1079   </rim:Slot>
1080 </rim:Association>
```

1081 **Example Representing owl:maxCardinality in ebRIM**

1082 4.5.3 owl:cardinality (only 0 or 1)

1083 In OWL Lite, cardinality is provided as a convenience when it is useful to state that a property on a class
1084 has both minCardinality 0 and maxCardinality 0 or both minCardinality 1 and maxCardinality 1
1085 [McGuinness, Harmelen].

1086 Consider for example the following OWL code which states that each instance of a “WebService” class
1087 must have exactly one price:

```
1088 <owl:Class rdf:ID="WebService">  
1089   <rdfs:subClassOf>  
1090     <owl:Restriction>  
1091       <owl:onProperty rdf:resource="#hasPrice"/>  
1092       <owl:Cardinality rdf:datatype="&xsd;nonNegativeInteger"> 1  
1093     </owl:Cardinality>  
1094   </owl:Restriction>  
1095 </rdfs:subClassOf>  
1096 </owl:Class>
```

1097 Example owl:Cardinality

1098 In ebXML RIM, cardinalities of Association Types MUST be defined by associating a Cardinality slot with
1099 the Association Types as shown in the following example:

```
1100  
1101 <rim:Association id = "hasPriceCardinalityRestriction"  
1102 associationType = "urn:oasis:names:tc:ebxml-  
1103 regrep:AssociationType:ObjectProperty" sourceObject = "WebService"  
1104 targetObject = "Price">  
1105   <rim:Name>  
1106     <rim:LocalizedString value = 'hasPrice' />  
1107   </rim:Name>  
1108   <rim:Slot name="cardinality">  
1109     <rim:ValueList>  
1110       <rim:Value>1</rim:Value>  
1111     </rim:ValueList>  
1112   </rim:Slot>  
1113 </rim:Association>
```

1114 Example Representing owl:Cardinality in ebRIM

1115 4.6 Representing OWL Class Intersection in ebXML RIM

1116 OWL provides the means to manipulate class extensions using basic set operators. In OWL lite, only
1117 “owl:intersectionOf” is available which defines a class that consists of exactly all objects that belong to
1118 both of the classes. In the following example, "AirReservationServices" is defined as the intersection of
1119 "AirServices" and "ReservationServices":

```
1120  
1121 <owl:Class rdf:ID="AirReservationServices">  
1122   <owl:intersectionOf rdf:parseType="Collection">  
1123     <owl:Class rdf:about="#AirServices" />  
1124     <owl:Class rdf:about="#ReservationServices" />  
1125   </owl:intersectionOf>  
1126 </owl:Class>
```

1127 Example owl:intersectionOf

1128 In ebXML RIM "owl:intersectionOf" set operator MUST be represented as follows:

- 1129 • A new Association Type called "intersectionOf" MUST be created.
- 1130 • A new ClassificationNode to denote the intersection of the classes MUST be created. For the
1131 example, this could be “AirReservationServices” ClassificationNode.
- 1132 • Each of the intersected classes MUST be represented as members of a new RegistryPackage.
1133 For the example, the RegistryPackage should contain “AirServices” and the
1134 “RegistrationServices”.

- The new ClassificationNode denoting the intersection MUST be assigned as the sourceObject of the "intersectionOf" association. For the example, "AirReservationServices" must be the the sourceObject of the "intersectionOf" association.
- The target class of the "intersectionOf" association MUST be set to the newly created RegistryPackage. For the example given above, the RegistryPackage containing "AirServices" and the "RegistrationServices" should be the target class of the "intersectionOf" association.

1141

```

1142 <rim:ClassificationNode id = "AirReservationServices" code =
1143 "AirReservationServices">
1144   <rim:Name>
1145     <rim:LocalizedString value = "AirReservationServices" />
1146   </rim:Name>
1147 </rim:ClassificationNode>

1148
1149
1150 <rim:RegistryPackage id = "IntersectionOfRegistryPackage" >
1151   <rim:Name>
1152     <rim:LocalizedString value =
1153 "IntersectionOfRegistryPackage"/>
1154   </rim:Name>
1155 </rim:RegistryPackage>

1156
1157 <rim:Association id = "HasMemberRegistryPackageAssoc1"
1158 associationType = "urn:oasis:names:tc:ebxml-
1159 regrep:AssociationType:HasMember" sourceObject =
1160 "IntersectionOfRegistryPackage"
1161 targetObject = "AirServices" />

1162
1163 <rim:Association id = "HasMemberRegistryPackageAssoc2"
1164 associationType = "urn:oasis:names:tc:ebxml-
1165 regrep:AssociationType:HasMember" sourceObject =
1166 "IntersectionOfRegistryPackage"
1167 targetObject = "ReservationServices" />

1168
1169 <rim:Association id = "IntersectionOfRegistryPackageAssoc"
1170 associationType = "urn:oasis:names:tc:ebxml-
1171 regrep:AssociationType:IntersectionOf" sourceObject =
1172 "AirReservationServices"
1173 targetObject = " IntersectionOfRegistryPackage " />
1174

```

1175 Example Defining Intersection of ClassificationNodes in ebRIM

1176 When such a representation is used to create a complex class (a new ClassificationNode) in RIM, it
 1177 becomes possible to infer that the objects (instances) classified by both of the classes
 1178 (ClassificationNodes) constituting the intersection are also the instances of this complex class. The adhoc
 1179 query presented in Section 6.23 MUST be available in the ebXML Registry to retrieve the direct instances
 1180 of the complex class and also the instances of the intersection of the classes.

1181

1182 4.7 Representing OWL Versioning in ebXML RIM

1183 4.7.1 owl:versionInfo, owl:priorVersion

1184 An owl:versionInfo statement generally has as its object a string giving information about this version, for
 1185 example RCS/ CVS keywords. This statement does not contribute to the logical meaning of the ontology
 1186 other than that given by the RDF(S) model theory [McGuinness, Harmelen].

1187 An owl:priorVersion statement contains a reference to another ontology. This identifies the specified
 1188 ontology as a prior version of the containing ontology [McGuinness, Harmelen].

1189 In ebXML, since a RegistryObject MAY have several versions, a logical id (called lid) is also defined which
 1190 is unique for different logical objects. However the lid attribute value MUST be the same for all versions of

1191 the same logical object. Therefore, almost all the underlying ebXML relational tables keep version
1192 information through “versionName” and “comment_” attributes.
1193 “owl:version” information MUST be stored in the “versionName” and “comment_” attributes of the table
1194 ClassScheme in the Registry.

1195 4.8 Representing OWL Annotation Properties in ebXML RIM

1196 4.8.1 rdfs:label

1197 rdfs:label is an instance of rdf:Property that may be used to provide a human-readable version of a
1198 resource's name [Brickley, Guha].

1199 In ebXML RIM, human readable names of resources are provided through rim:Name. rdfs:label MUST be
1200 expressed through rim:Name.

1201

```
1202 <owl:Class rdf:ID="AirReservationServices">  
1203   <rdfs:label>Air Reservation Services</rdfs:label>  
1204 </owl:Class>
```

1205 Example rdfs:label

1206

```
1207 <rim:ClassificationNode id = 'AirReservationServices' code =  
1208 'AirReservationServices'>  
1209   <rim:Name>  
1210     <rim:LocalizedString value = 'Air Reservation Services' />  
1211   </rim:Name>  
1212 </rim:ClassificationNode>
```

1213 Example rim:Name

1214 4.8.2 rdfs:comment

1215 rdfs:comment is an instance of rdf:Property that may be used to provide a human-readable description of
1216 a resource [Brickley, Guha].

1217 In ebXML RIM, this construct MUST be expressed through rim:Description.

1218

```
1219 <owl:Class rdf:ID="AirReservationServices">  
1220   <rdfs:comment>Open Travel Alliance Air Reservation Services  
1221   </rdfs:comment>  
1222 </owl:Class>
```

1223 Example rdfs:comment

1224

```
1225 <rim:ClassificationNode id = 'AirReservationServices' code =  
1226 'AirReservationServices'>  
1227   <rim:Description>  
1228     <rim:LocalizedString value = 'Open Travel Alliance Air  
1229     Reservation Services' />  
1230   </rim:Description>  
1231 </rim:ClassificationNode>
```

1232 Example: rim:Description

1233 4.8.3 rdfs:seeAlso

1234 rdfs:seeAlso is an instance of rdf:Property that is used to indicate a resource that might provide additional
1235 information about the subject resource [Brickley, Guha].

1236 This construct MUST be expressed in ebXML RIM by defining an ExternalLink, called,
1237 "seeAlsoExternalLink".

1238

```
1239 <owl:Class rdf:ID="AirReservationServices">  
1240   <rdfs:seeAlso rdf:resource="http://www.opentravel.org" />  
1241 </owl:Class>
```

1242 **Example rdfs:seeAlso**

```
1243 <rim:ClassificationNode id = 'AirReservationServices' code =  
1244 'AirReservationServices'>  
1245 </rim:ClassificationNode>  
1246  
1247 <rim:ExternalLink id = "seeAlsoExternalLink"  
1248   externalURI= "http://www.opentravel.org" >  
1249 </rim:ExternalLink>  
1250  
1251 <rim:Association id = 'seeAlsoAssociation'  
1252   associationType = 'urn:oasis:names:tc:ebxml-  
1253   regrep:AssociationType:ExternallyLinks '  
1254   sourceObject = 'AirReservationServices'  
1255   targetObject = 'seeAlsoExternalLink' />
```

1256 **Example rim:seeAlsoExternalLink**

1257 **4.9 OWL Datatypes in ebXML RIM**

1258 OWL allows the use of XML Schema datatypes to describe part of the datatype domain by simply
1259 including their URIs within an OWL ontology [McGuinness, Harmelen]. In ebXML, XML Schema datatypes
1260 SHOULD be used by providing an external link from the registry.

1261 The following example demonstrates how XML Schema datatype "integer" can be referenced through an
1262 ExternalLink called 'integer' and how to define a DatatypeProperty, namely, "hasPrice", whose target
1263 object is the defined to be ExternalLink 'integer':

1264

```
1265 <rim:ExternalLink id = "integer"  
1266   externalURI="http://www.w3.org/2001/XMLSchema#integer" >  
1267   <rim:Name> <rim:LocalizedString value = "XML Schema integer"/>  
1268   </rim:Name>  
1269 </rim:ExternalLink>  
1270 <rim:Association id = 'hasPrice' associationType = 'urn:oasis:names:tc:ebxml-  
1271   regrep:AssociationType:DatatypeProperty'  
1272   sourceObject = 'AirReservationServices'  
1273   targetObject = 'integer' >  
1274   <rim:Name> <rim:LocalizedString value = "hasPrice"/></rim:Name>  
1275 </rim:Association>
```

1277 **Example Corresponding ebRIM construct Association**

1278

1279 5 Cataloging Service Profile

1280 The ebXML Registry provides the ability for a content cataloging service to be configured for any type of
1281 content. The cataloging service serves the following purposes:

- 1282 • Automates the mapping from the source information model (in this case OWL) to ebRIM. This
1283 hides the complexity of the mapping from the OWL publisher and eliminates the need for any
1284 special UI tools to be provided by the registry implementor for publishing OWL documents.
- 1285 • Selectively converts content into ebRIM compatible metadata when the content is cataloged after
1286 being published. The generated metadata enables the selected content to be used as
1287 parameter(s) in content specific parameterized queries.

1288 This section describes the cataloging service for cataloging OWL content.

1289 An OWL document, when published to an ebXML Registry implementing the OWL Profile, **MUST** be
1290 cataloged as specified in this section using a OWL Content Cataloging Service as defined by [ebRS].

1291 5.1 Invocation Control File

1292 The OWL cataloging service **MAY** optionally support an invocation control file that declaratively specifies
1293 the transforms necessary to catalog published OWL documents.

1294 5.2 Input Metadata

1295 The OWL cataloging service **MUST** be pre-configured to be automatically invoked when the following
1296 types of metadata are published, as defined by the [ebRS] specifications.

1297 These are the only types of metadata that **MAY** describe a OWL document being published:

- 1298 • An `ExtrinsicObject` whose `ObjectType` references the canonical OWL `ClassificationNode`
1299 specified in Section 7. The `ExtrinsicObject` **MUST** have an OWL document as its `RepositoryItem`.
- 1300 • An `ExternalLink` whose `ObjectType` references the canonical OWL `ClassificationNode` specified in
1301 Section 7. In case of `ExternalLink` the OWL document **MUST** be resolvable via a URL described
1302 by the value of the `externalURI` attribute of the `ExternalLink`. Recall that, in the `ExternalLink` case
1303 the OWL document is not be stored in the repository.

1304

```
1305 <rim:ExtrinsicObject id="urn:acmeinc:ebxml:registry:3.0:owl">  
1306 ...  
1307 </rim:ExtrinsicObject>
```

1308 Example of ExtrinsicObject Input Metadata

1309

```
1310 <rim:ExternalLink  
1311 id="urn:acmeinc:ebxml:registry:3.0:owl"  
1312 externalURI="http://www.acme.com/owl/ebXMLRegistryService.owl"  
1313 >  
1314 ...  
1315 </rim:ExternalLink>
```

1316 Example of ExternalLink Input Metadata

1317 5.3 Input Content

1318 The OWL cataloging service expects an OWL document as its input content. The input content **MUST** be
1319 processed by the OWL cataloging service regardless of whether it is a `RepositoryItem` for an
1320 `ExtrinsicObject` or whether it is content external to repository that is referenced by an `ExternalLink`.

1321

1322 5.4 Output Metadata

1323 This section describes the metadata produced by the OWL cataloging service produces as output.

1324 5.4.1 owl:Class → rim:ClassificationNode

1325 The OWL Cataloging service MUST automatically produce a rim:ClassificationNode instance for each
1326 owl:class element within the input OWL or its imports, as specified in the owl:Class →
1327 rim:ClassificationNode mapping earlier in this document.

1328 5.4.2 rdf:Property → rim:Association Type Property

1329 The OWL Cataloging service MUST automatically produce an rim:Association instance with
1330 associationType Property for each rdf:Property element within the input OWL or its imports, as specified
1331 in the rdf:Property → rim:Association Type Property mapping earlier in this document.

1332 5.4.3 rdfs:subPropertyOf → rim:Association Type subPropertyOf

1333 The OWL Cataloging service MUST automatically produce an rim:Association instance with
1334 associationType subPropertyOf for each rdfs:subPropertyOf element within the input OWL or its imports,
1335 as specified in the rdfs:subPropertyOf → rim:Association Type subPropertyOf mapping earlier in this
1336 document.

1337 5.4.4 rdfs:subClassOf → rim:Association Type subClassOf

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

1341 5.4.5 owl:Individual → rim:ExtrinsicObject

1342 The OWL Cataloging service MUST automatically produce rim:ExtrinsicObject instances for each
1343 owl:Individual element within the input OWL or its imports, as specified in the owl:Individual →
1344 rim:ExtrinsicObject mapping earlier in this document.

1345 5.4.6 owl:equivalentClass, owl:equivalentProperty → rim:Association Type 1346 EquivalentTo

1347 The OWL Cataloging service MUST automatically produce rim:Association instances with
1348 associationType EquivalentTo for each owl:equivalentClass or owl:equivalentProperty element within the
1349 input OWL or its imports, as specified in the owl:equivalentClass, owl:equivalentProperty →
1350 rim:Association Type EquivalentTo mapping earlier in this document.

1351 5.4.7 owl:sameAs → rim:Association Type sameAs

1352 The OWL Cataloging service MUST automatically produce rim:Association instances with
1353 associationType sameAs for each owl:sameAs element within the input OWL or its imports, as specified
1354 in the owl:sameAs → rim:Association Type sameAs mapping earlier in this document.

1355 5.4.8 owl:differentFrom → rim:Association Type differentFrom

1356 The OWL Cataloging service MUST automatically produce rim:Association instances with
1357 associationType differentFrom for each owl:differentFrom element within the input OWL or its imports, as
1358 specified in the owl:differentFrom → rim:Association Type differentFrom mapping earlier in this document.

1359 5.4.9 owl:AllDifferent → rim:RegistryPackage

1360 The OWL Cataloging service MUST automatically produce rim:RegistryPackage instances for each
1361 owl:AllDifferent element within the input OWL or its imports, as specified in the owl:AllDifferent →

1362 rim:RegistryPackage mapping earlier in this document.

1363 **5.4.10 owl:ObjectProperty → rim:Association Type objectProperty**

1364 The OWL Cataloging service MUST automatically produce rim:Association instances with
1365 associationType objectProperty for each owl:ObjectProperty element within the input OWL or its imports,
1366 as specified in the owl:ObjectProperty → rim:Association Type objectProperty mapping earlier in this
1367 document.

1368 **5.4.11 owl:DatatypeProperty → rim:Association Type DatatypeProperty**

1369 The OWL Cataloging service MUST automatically produce rim:Association instances with
1370 associationType datatypeProperty for each owl:DatatypeProperty element within the input OWL or its
1371 imports, as specified in the owl:DatatypeProperty → rim:Association Type datatypeProperty mapping
1372 earlier in this document.

1373 **5.4.12 owl:TransitiveProperty → rim:Association Type transitiveProperty**

1374 The OWL Cataloging service MUST automatically produce rim:Association instances with
1375 associationType transitiveProperty for each owl:TransitiveProperty element within the input OWL or its
1376 imports, as specified in the owl:TransitiveProperty → rim:Association Type transitiveProperty mapping
1377 earlier in this document.

1378 **5.4.13 owl:inverseOf → rim:Association Type inverseOf**

1379 The OWL Cataloging service MUST automatically produce rim:Association instances with
1380 associationType inverseOf for each owl:inverseOf element within the input OWL or its imports, as
1381 specified in the owl:inverseOf → rim:Association Type inverseOf mapping earlier in this document.

1382 **5.4.14 owl:SymmetricProperty → rim:Association Type SymetricProperty**

1383 The OWL Cataloging service MUST automatically produce rim:Association instances with
1384 associationType SymetricProperty for each owl:SymetricProperty element within the input OWL or its
1385 imports, as specified in the owl:SymetricProperty → rim:Association Type SymetricProperty mapping
1386 earlier in this document.

1387 **5.4.15 owl:FunctionalProperty → rim:Association Type FunctionalProperty**

1388 The OWL Cataloging service MUST automatically produce rim:Association instances with
1389 associationType FunctionalProperty for each owl:FunctionalProperty element within the input OWL or its
1390 imports, as specified in the owl:FunctionalProperty → rim:Association Type FunctionalProperty mapping
1391 earlier in this document.

1392 **5.4.16 owl:InverseFunctionalProperty → rim:Association Type 1393 InverseFunctionalProperty**

1394 The OWL Cataloging service MUST automatically produce rim:Association instances with
1395 associationType InverseFunctionalProperty for each owl:InverseFunctionalProperty element within the
1396 input OWL or its imports, as specified in the owl:InverseFunctionalProperty → rim:Association Type
1397 InverseFunctionalProperty mapping earlier in this document.

1398 **5.4.17 owl:minCardinality (only 0 or 1)**

1399 The OWL Cataloging service MUST automatically add a slot with name minCardinality to the relevant
1400 rim:Association instances for each owl:minCardinality element within the input OWL or its imports, as
1401 specified in section 4.5.1 where how to represent owl:minCardinality is described.

1402 **5.4.18 owl:maxCardinality (only 0 or 1)**

1403 The OWL Cataloging service MUST automatically add a slot with name maxCardinality to the relevant
1404 rim:Association instances for each owl:maxCardinality element within the input OWL or its imports, as
1405 specified in section 4.5.2 where how to represent owl:maxCardinality is described.

1406 **5.4.19 owl:cardinality**

1407 The OWL Cataloging service MUST automatically add a slot with name cardinality to the relevant
1408 rim:Association instances for each owl:cardinality element within the input OWL or its imports, as
1409 specified in section 4.5.3 where how to represent owl:cardinality is described.

1410 **5.4.20 owl:intersectionOf**

1411 The OWL Cataloging service MUST automatically produce a rim:RegistryPackage and a rim:Association
1412 instances with type IntersectionOf for each owl:intersectionOf element within the input OWL or its imports,
1413 as specified in section 4.6 where how to represent owl:intersectionOf is described.

1414 **5.4.21 rdfs:label**

1415 The OWL Cataloging service MUST automatically produce a rim:Name instance for each rdfs:label
1416 element within the input OWL or its imports, as specified in section 4.8.1 where how to represent
1417 rdfs:label is described.

1418 **5.4.22 rdfs:comment**

1419 The OWL Cataloging service MUST automatically produce a rim:Description instance for each
1420 rdfs:comment element within the input OWL or its imports, as specified in section 4.8.2 where how to
1421 represent rdfs:comment is described.

1422 **5.4.23 rdfs:seeAlso**

1423 The OWL Cataloging service MUST automatically produce a rim:ExternalLink and a rim:Association with
1424 type ExternallyLinks instances for each rdfs:seeAlso element within the input OWL or its imports, as
1425 specified in section 4.8.3 where how to represent rdfs:seeAlso is described.

6 Discovery Profile

1426

1427 The ebXML Registry provides the ability for a user defined parameterized queries to be configured for
1428 each type of content. The queries may be as complex or simple as the discovery use case requires. The
1429 complexity of the parameterized queries may hidden from the registry client by storing them within the
1430 ebXML Registry as instances of the AdhocQuery class, and being invoked by simply providing their
1431 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].

1433 An ebXML Registry SHOULD provide a graphical user interface that displays any configured
1434 parameterized query as a form which contains an appropriate field for entering each query parameter.

1435 This chapter defines the queries that MUST be support by an ebXML Registry implementing the OWL
1436 Profile for processing the semantics provided in the OWL content. An implementation MAY also support
1437 additional discovery queries for OWL content, some of which have already identified in this section.

1438 The queries defined in this chapter are parameterized queries stored in the Registry as instances of the
1439 AdhocQuery type, in the same manner as any other RegistryObject.

1440 In the subsequent section each query is described simply in terms of its supported parameters that serve
1441 as its search criteria. The actual AdhocQuery instances are much more complex in comparison but they
1442 are not exposed to the client making the query. Details on these queries are specified canonically in
1443 section 7.3 .

1444 Some of the queries that are necessary to process the semantics involved in OWL documents requires
1445 SQL recursion mechanism. Since SQL 92, does not support recursion mechanism, those queries are
1446 stated to be implemented optionally. Additionally for these types of discovery queries, the “stored
1447 procedures” are presented in Section 7.3.

1448 6.1 All SuperProperties Discovery Query

1449 As presented in Section 4.1.3, a new ebXML RIM Association Type called “SubPropertyOf” MUST be
1450 defined to represent rdfs:subPropertyOf in ebRIM. Such a semantic enhancement brings the following
1451 processing need: given a property, it should be possible to retrieve all of its super properties. This requires
1452 a recursion mechanism in SQL queries.

1453 The freebXML implementation allows various relational database products such as Oracle, PostgreSQL
1454 and MS SQL Server 2005 to be used as the database. These products have different support for
1455 recursion mechanism in SQL Queries.

1456 The AllSuperProperties discovery query MAY be implemented by an ebXML Registry implementing this
1457 profile. It allows the discovery of all super properties of a given property instance (Association instance in
1458 ebXML terminology) recursively in a property hierarchy (hierarchy of Association Types) in freebXML
1459 Registry implementations using MS SQL Server 2005 as the database.

1460 6.1.1 Parameter \$propertyName

1461 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1462 value of Associations that have associationType of Property.

1463 6.1.2 Example of All SuperProperties Discovery Query

1464 The following example illustrates how to find all the super properties of a given property having a name
1465 containing “creditCardPayment” if the query is implemented as an AdHoc Query.

1466

```
1467 <<rs:RequestSlotList>  
1468   <rim:Slot  
1469     name="urn:oasis:names:tc:ebxml-  
1470     regrep:3.0:rs:AdhocQueryRequest:queryId">  
1471     <rim:ValueList>  
1472       <rim:Value>urn:oasis:names:tc:ebxml-  
1473       regrep:query:FindAllSuperProperties</rim:Value>
```

```

1474         </rim:ValueList>
1475     </rim:Slot>
1476     <rim:Slot name="urn:oasis:names:tc:ebxml-
1477 regrep:rs:AdhocQueryRequest:queryId">
1478         <rim:ValueList>
1479             <rim:Value>urn:oasis:names:tc:ebxml-
1480 regrep:query:FindAllSuperProperties</rim:Value>
1481         </rim:ValueList>
1482     </rim:Slot>
1483     <rim:Slot name="$propertyName">
1484         <rim:ValueList>
1485             <rim:Value>%creditCardPayment%</rim:Value>
1486         </rim:ValueList>
1487     </rim:Slot>
1488 </rs:RequestSlotList>
1489
1490 <query:ResponseOption returnComposedObjects="true"
1491     returnType="LeafClassWithRepositoryItem"/>
1492
1493 <rim:AdhocQuery id="temporaryId">
1494     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1495 regrep:QueryLanguage:SQL-92">
1496         </rim:QueryExpression>
1497 </rim:AdhocQuery>

```

1498 Example of All SuperProperties Discovery Query

1499 6.2 Immediate SuperClass Discovery Query

1500 The Immediate SuperClass discovery query MUST be implemented by an ebXML Registry implementing
1501 this profile. It allows the discovery of all of the immediate super classes of a given class.

1502 6.2.1 Parameter \$className

1503 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1504 value of ClassificationNodes.

1505 6.2.2 Example of Immediate SuperClass Discovery Query

1506 The following example illustrates how to find all the immediate super classes of a given class that have a
1507 name containing the string "AirReservationServices".

```

1508 <rs:RequestSlotList>
1509     <rim:Slot
1510         name="urn:oasis:names:tc:ebxml-
1511 regrep:3.0:rs:AdhocQueryRequest:queryId">
1512         <rim:ValueList>
1513             <rim:Value>urn:oasis:names:tc:ebxml-
1514 regrep:query:FindImmediateSuperClasses</rim:Value>
1515         </rim:ValueList>
1516     </rim:Slot>
1517     <rim:Slot name="urn:oasis:names:tc:ebxml-
1518 regrep:rs:AdhocQueryRequest:queryId">
1519         <rim:ValueList>
1520             <rim:Value>urn:oasis:names:tc:ebxml-
1521 regrep:query:FindImmediateSuperClasses</rim:Value>
1522         </rim:ValueList>
1523     </rim:Slot>
1524     <rim:Slot name="$className">
1525         <rim:ValueList>
1526             <rim:Value>%AirReservationServices%</rim:Value>
1527         </rim:ValueList>
1528     </rim:Slot>
1529 </rs:RequestSlotList>
1530
1531 <query:ResponseOption returnComposedObjects="true"

```



```

1532         returnType="LeafClassWithRepositoryItem"/>
1533
1534 <rim:AdhocQuery id="temporaryId">
1535     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1536     regrep:QueryLanguage:SQL-92">
1537         </rim:QueryExpression>
1538     </rim:AdhocQuery>

```

1539 Example of Immediate SuperClass Discovery Query

1540 6.3 Immediate SubClass Discovery Query

1541 The Immediate SubClass discovery query MUST be implemented by an ebXML Registry implementing
 1542 this profile. It allows the discovery of all of the immediate subclasses of a given class.

1543 6.3.1 Parameter \$className

1544 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
 1545 value of ClassificationNode.

1546 6.3.2 Example of Immediate SubClass Discovery Query

1547 The following example illustrates how to find all the immediate subclasses of a given class that have a
 1548 name containing the string "AirServices" .

```

1549 <rs:RequestSlotList>
1550     <rim:Slot
1551         name="urn:oasis:names:tc:ebxml-
1552     regrep:3.0:rs:AdhocQueryRequest:queryId">
1553         <rim:ValueList>
1554             <rim:Value>urn:oasis:names:tc:ebxml-
1555     regrep:query:FindImmediateSubClasses</rim:Value>
1556         </rim:ValueList>
1557     </rim:Slot>
1558     <rim:Slot name="urn:oasis:names:tc:ebxml-
1559     regrep:rs:AdhocQueryRequest:queryId">
1560         <rim:ValueList>
1561             <rim:Value>urn:oasis:names:tc:ebxml-
1562     regrep:query:FindImmediateSubClasses</rim:Value>
1563         </rim:ValueList>
1564     </rim:Slot>
1565     <rim:Slot name="$className">
1566         <rim:ValueList>
1567             <rim:Value>%AirServices%</rim:Value>
1568         </rim:ValueList>
1569     </rim:Slot>
1570 </rs:RequestSlotList>
1571
1572 <query:ResponseOption returnComposedObjects="true"
1573     returnType="LeafClassWithRepositoryItem"/>
1574
1575 <rim:AdhocQuery id="temporaryId">
1576     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1577     regrep:QueryLanguage:SQL-92">
1578         </rim:QueryExpression>
1579     </rim:AdhocQuery>

```

1580 Example of Immediate SubClass Discovery Query

1581 6.4 All SuperClasses Discovery Query

1582 It should be noted that, given a class, finding its immediate subclasses, super classes is necessary but
 1583 not sufficient. Given a class, it should be possible to retrieve all of its subclasses, and all of its super
 1584 classes. This requires a recursion mechanism in SQL queries. The freebXML implementation allows
 1585 various relational database products such as Oracle, PostgreSQL and MS SQL Server 2005 to be used

1586 as the database. These products have different support for recursion mechanisms in SQL Queries.

1587 The All SuperClasses discovery query MAY be implemented by an ebXML Registry implementing this
1588 profile. It allows the discovery of all super classes of a given ClassificationNode recursively in freebXML
1589 Registry implementations using MS SQL Server 2005 as the database.

1590 6.4.1 Parameter \$className

1591 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1592 value of ClassificationNode.

1593 6.4.2 Example of All SuperClasses Discovery Query

1594 The following example illustrates how to find all the super classes of a given class recursively that have a
1595 name containing the string "AirReservationServices" if the query is implemented as an Adhoc Query .

```
1596 <rs:RequestSlotList>  
1597   <rim:Slot  
1598     name="urn:oasis:names:tc:ebxml-  
1599   regrep:3.0:rs:AdhocQueryRequest:queryId">  
1600     <rim:ValueList>  
1601       <rim:Value>urn:oasis:names:tc:ebxml-  
1602   regrep:query:FindAllSuperClasses</rim:Value>  
1603     </rim:ValueList>  
1604   </rim:Slot>  
1605   <rim:Slot name="urn:oasis:names:tc:ebxml-  
1606   regrep:rs:AdhocQueryRequest:queryId">  
1607     <rim:ValueList>  
1608       <rim:Value>urn:oasis:names:tc:ebxml-  
1609   regrep:query:FindAllSuperClasses</rim:Value>  
1610     </rim:ValueList>  
1611   </rim:Slot>  
1612   <rim:Slot name="$className">  
1613     <rim:ValueList>  
1614       <rim:Value>%AirReservationServices%</rim:Value>  
1615     </rim:ValueList>  
1616   </rim:Slot>  
1617 </rs:RequestSlotList>  
  
1618  
1619 <query:ResponseOption returnComposedObjects="true"  
1620   returnType="LeafClassWithRepositoryItem"/>  
1621  
1622 <rim:AdhocQuery id="temporaryId">  
1623   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
1624   regrep:QueryLanguage:SQL-92">  
1625     </rim:QueryExpression>  
1626 </rim:AdhocQuery>
```

1627 Example of All SuperClasses Discovery Query

1628 6.5 All SubClasses Discovery Query

1629 The All SubClasses discovery query MAY be implemented by an ebXML Registry implementing this
1630 profile. It allows the discovery of all subclasses of a given ClassificationNode recursively in a freebXML
1631 Registry implementations supporting recursion.

1632 6.5.1 Parameter \$className

1633 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1634 value of ClassificationNode.

1635 6.5.2 Example of All SubClasses Discovery Query

1636 The following example illustrates how to find all the subclasses of a given class recursively that have a
1637 name containing the string "AirServices", if the query is implemented as an Adhoc Query.

```
1638 <rs:RequestSlotList>
1639   <rim:Slot
1640     name="urn:oasis:names:tc:ebxml-
1641   regrep:3.0:rs:AdhocQueryRequest:queryId">
1642     <rim:ValueList>
1643       <rim:Value>urn:oasis:names:tc:ebxml-
1644   regrep:query:FindAllSubClasses</rim:Value>
1645     </rim:ValueList>
1646   </rim:Slot>
1647   <rim:Slot name="urn:oasis:names:tc:ebxml-
1648   regrep:rs:AdhocQueryRequest:queryId">
1649     <rim:ValueList>
1650       <rim:Value>urn:oasis:names:tc:ebxml-
1651   regrep:query:FindAllSubClasses</rim:Value>
1652     </rim:ValueList>
1653   </rim:Slot>
1654   <rim:Slot name="$className">
1655     <rim:ValueList>
1656       <rim:Value>%AirServices%</rim:Value>
1657     </rim:ValueList>
1658   </rim:Slot>
1659 </rs:RequestSlotList>
1660
1661 <query:ResponseOption returnComposedObjects="true"
1662   returnType="LeafClassWithRepositoryItem"/>
1663
1664 <rim:AdhocQuery id="temporaryId">
1665   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1666   regrep:QueryLanguage:SQL-92">
1667     </rim:QueryExpression>
1668 </rim:AdhocQuery>
```

1669 Example of All SubClasses Discovery Query

1670 6.6 EquivalentClasses Discovery Query

1671 The EquivalentClasses discovery query MUST be implemented by an ebXML Registry implementing this
1672 profile. It allows the discovery of all the equivalent classes of a given ClassificationNode.

1673 6.6.1 Parameter \$className

1674 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1675 value of ClassificationNodes.

1676 6.6.2 Example of EquivalentClasses Discovery Query

1677 The following example illustrates how to find all the equivalent classes of a given class that have a name
1678 containing the string "AirServices".

```
1679 <rs:RequestSlotList>
1680   <rim:Slot
1681     name="urn:oasis:names:tc:ebxml-
1682   regrep:3.0:rs:AdhocQueryRequest:queryId">
1683     <rim:ValueList>
1684       <rim:Value>urn:oasis:names:tc:ebxml-
1685   regrep:query:FindEquivalentClasses</rim:Value>
1686     </rim:ValueList>
1687   </rim:Slot>
1688   <rim:Slot name="urn:oasis:names:tc:ebxml-
1689   regrep:rs:AdhocQueryRequest:queryId">
1690     <rim:ValueList>
```

```

1691         <rim:Value>urn:oasis:names:tc:ebxml-
1692 regrep:query:FindEquivalentClasses</rim:Value>
1693     </rim:ValueList>
1694 </rim:Slot>
1695 <rim:Slot name="$className">
1696     <rim:ValueList>
1697         <rim:Value>%AirServices%</rim:Value>
1698     </rim:ValueList>
1699 </rim:Slot>
1700 </rs:RequestSlotList>
1701
1702 <query:ResponseOption returnComposedObjects="true"
1703     returnType="LeafClassWithRepositoryItem"/>
1704
1705 <rim:AdhocQuery id="temporaryId">
1706     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1707 regrep:QueryLanguage:SQL-92">
1708     </rim:QueryExpression>
1709 </rim:AdhocQuery>

```

1710 Example of Equivalent Classes Discovery Query

1711 6.7 EquivalentProperties Discovery Query

1712 The EquivalentProperties discovery query MUST be implemented by an ebXML Registry implementing
1713 this profile. It allows the discovery of all the equivalent properties of a given Association that have
1714 associationType of Property.

1715 6.7.1 Parameter \$propertyName

1716 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1717 value of Associations that have associationType of Property

1718 6.7.2 Example of EquivalentProperties Discovery Query

1719 The following example illustrates how to find all the equivalent properties (Association Type) of a given
1720 property (Association Type) that have a name containing the string "paymentMethods".

```

1721 <rs:RequestSlotList>
1722     <rim:Slot
1723         name="urn:oasis:names:tc:ebxml-
1724 regrep:3.0:rs:AdhocQueryRequest:queryId">
1725         <rim:ValueList>
1726             <rim:Value>urn:oasis:names:tc:ebxml-
1727 regrep:query:FindEquivalentProperties</rim:Value>
1728         </rim:ValueList>
1729     </rim:Slot>
1730     <rim:Slot name="urn:oasis:names:tc:ebxml-
1731 regrep:rs:AdhocQueryRequest:queryId">
1732         <rim:ValueList>
1733             <rim:Value>urn:oasis:names:tc:ebxml-
1734 regrep:query:FindEquivalentProperties</rim:Value>
1735         </rim:ValueList>
1736     </rim:Slot>
1737     <rim:Slot name="$propertyName">
1738         <rim:ValueList>
1739             <rim:Value>%paymentMethods%</rim:Value>
1740         </rim:ValueList>
1741     </rim:Slot>
1742 </rs:RequestSlotList>
1743
1744 <query:ResponseOption returnComposedObjects="true"
1745     returnType="LeafClassWithRepositoryItem"/>
1746
1747 <rim:AdhocQuery id="temporaryId">

```

```
1748     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1749     regrep:QueryLanguage:SQL-92">
1750         </rim:QueryExpression>
1751 </rim:AdhocQuery>
```

1752 Example of Equivalent Properties Discovery Query

1753 6.8 SameExtrinsicObjects Discovery Query

1754 The SameExtrinsicObjects discovery query MUST be implemented by an ebXML Registry implementing
1755 this profile. It allows the discovery of all the "ExtrinsicObjects" defined to be the same with a given
1756 ExtrinsicObject.

1757 6.8.1 Parameter \$extrinsicObjectName

1758 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1759 value of ExtrinsicObjects.

1760 6.8.2 Example of SameExtrinsicObjects Discovery Query

1761 The following example illustrates how to find all the ExtrinsicObjects that are defined to be the same as
1762 the ExtrinsicObject that have a name containing the string "MyDocument" .

```
1763 <rs:RequestSlotList>
1764     <rim:Slot
1765         name="urn:oasis:names:tc:ebxml-
1766     regrep:3.0:rs:AdhocQueryRequest:queryId">
1767         <rim:ValueList>
1768             <rim:Value>urn:oasis:names:tc:ebxml-
1769     regrep:query:FindTheSameExtrinsicObjects</rim:Value>
1770         </rim:ValueList>
1771     </rim:Slot>
1772     <rim:Slot name="urn:oasis:names:tc:ebxml-
1773     regrep:rs:AdhocQueryRequest:queryId">
1774         <rim:ValueList>
1775             <rim:Value>urn:oasis:names:tc:ebxml-
1776     regrep:query:FindTheSameExtrinsicObjects</rim:Value>
1777         </rim:ValueList>
1778     </rim:Slot>
1779     <rim:Slot name="$extrinsicObjectName">
1780         <rim:ValueList>
1781             <rim:Value>%MyDocument%</rim:Value>
1782         </rim:ValueList>
1783     </rim:Slot>
1784 </rs:RequestSlotList>
1785
1786 <query:ResponseOption returnComposedObjects="true"
1787     returnType="LeafClassWithRepositoryItem"/>
1788
1789 <rim:AdhocQuery id="temporaryId">
1790     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1791     regrep:QueryLanguage:SQL-92">
1792         </rim:QueryExpression>
1793 </rim:AdhocQuery>
```

1795 Example of SameExtrinsicObjects Discovery Query

1796 6.9 DifferentExtrinsicObjects Discovery Query

1797 The DifferentExtrinsicObjects discovery query MUST be implemented by an ebXML Registry
1798 implementing this profile. It allows the discovery of all the "ExtrinsicObjects" defined to be the different
1799 from a given ExtrinsicObject.

1800 6.9.1 Parameter \$extrinsicObjectName

1801 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1802 value of ExtrinsicObjects.

1803 6.9.2 Example of DifferentExtrinsicObjects Discovery Query

1804 The following example illustrates how to find all the ExtrinsicObjects that are defined to be different from
1805 the ExtrinsicObject that have a name containing the string "MyDocument" .

```
1806 <rs:RequestSlotList>
1807   <rim:Slot
1808     name="urn:oasis:names:tc:ebxml-
1809   regrep:3.0:rs:AdhocQueryRequest:queryId">
1810     <rim:ValueList>
1811       <rim:Value>urn:oasis:names:tc:ebxml-
1812   regrep:query:FindDifferentExtrinsicObjects</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:FindDifferentExtrinsicObjects</rim:Value>
1820     </rim:ValueList>
1821   </rim:Slot>
1822   <rim:Slot name="$extrinsicObjectName">
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
```

1838 Example of DifferentExtrinsicObjects Discovery Query

1839 6.10 AllDifferentRegistryObject Discovery Query

1840 The AllDifferentRegistryObjects discovery query MUST be implemented by an ebXML Registry
1841 implementing this profile. Given a RegistryObject, it allows the discovery of all the other member
1842 "RegistryObjects" of a Registry package that are defined to be the different from each other through a
1843 allDifferent slot.

1844 6.10.1 Parameter \$registryObjectName

1845 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1846 value of RegistryObjects.

1847 6.10.2 Example of AllDifferentRegistryObjects Discovery Query

1848 The following example illustrates how to find all the RegistryObjects that are defined to be different from
1849 the RegistryObject that have a name containing the string "MyDocument" .

```
1850
1851 <rs:RequestSlotList>
1852   <rim:Slot
```

```

1853         name="urn:oasis:names:tc:ebxml-
1854 regrep:3.0:rs:AdhocQueryRequest:queryId">
1855         <rim:ValueList>
1856             <rim:Value>urn:oasis:names:tc:ebxml-
1857 regrep:query:FindAllDifferent</rim:Value>
1858         </rim:ValueList>
1859     </rim:Slot>
1860     <rim:Slot name="urn:oasis:names:tc:ebxml-
1861 regrep:rs:AdhocQueryRequest:queryId">
1862         <rim:ValueList>
1863             <rim:Value>urn:oasis:names:tc:ebxml-
1864 regrep:query:FindAllDifferent</rim:Value>
1865         </rim:ValueList>
1866     </rim:Slot>
1867     <rim:Slot name="$registryObjectName">
1868         <rim:ValueList>
1869             <rim:Value>%MyDocument%</rim:Value>
1870         </rim:ValueList>
1871     </rim:Slot>
1872 </rs:RequestSlotList>
1873
1874 <query:ResponseOption returnComposedObjects="true"
1875     returnType="LeafClassWithRepositoryItem"/>
1876
1877 <rim:AdhocQuery id="temporaryId">
1878     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1879 regrep:QueryLanguage:SQL-92">
1880     </rim:QueryExpression>
1881 </rim:AdhocQuery>

```

1882 Example of AllDifferentRegistryObjects Discovery Query

1883 6.11 ObjectProperties Discovery Query

1884 The ObjectProperties discovery query MUST be implemented by an ebXML Registry implementing this
1885 profile. It allows the discovery of all of the objectProperties of a given classification node.

1886 6.11.1 Parameter \$className

1887 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1888 value of ClassificationNodes.

1889 6.11.2 Example of ObjectProperties Discovery Query

1890 The following example illustrates how to find all the object properties of a given classification node having
1891 a name containing "AirServices" .

```

1892
1893 <rs:RequestSlotList>
1894     <rim:Slot
1895         name="urn:oasis:names:tc:ebxml-
1896 regrep:3.0:rs:AdhocQueryRequest:queryId">
1897         <rim:ValueList>
1898             <rim:Value>urn:oasis:names:tc:ebxml-
1899 regrep:query:FindObjectProperties</rim:Value>
1900         </rim:ValueList>
1901     </rim:Slot>
1902     <rim:Slot name="urn:oasis:names:tc:ebxml-
1903 regrep:rs:AdhocQueryRequest:queryId">
1904         <rim:ValueList>
1905             <rim:Value>urn:oasis:names:tc:ebxml-
1906 regrep:query:FindObjectProperties</rim:Value>
1907         </rim:ValueList>
1908     </rim:Slot>
1909     <rim:Slot name="$className">

```

```

1910     <rim:ValueList>
1911         <rim:Value>%AirServices%</rim:Value>
1912     </rim:ValueList>
1913 </rim:Slot>
1914 </rs:RequestSlotList>
1915
1916 <query:ResponseOption returnComposedObjects="true"
1917     returnType="LeafClassWithRepositoryItem"/>
1918
1919 <rim:AdhocQuery id="temporaryId">
1920     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1921     regrep:QueryLanguage:SQL-92">
1922     </rim:QueryExpression>
1923 </rim:AdhocQuery>

```

1924 Example of ObjectProperties Discovery Query

1925 6.12 ImmediateInheritedObjectProperties Discovery Query

1926 The ImmediateInheritedObjectProperties discovery query MUST be implemented by an ebXML Registry
 1927 implementing this profile. It allows the discovery of all of the objectProperties of a given classification node
 1928 including the ones inherited from its immediate super classes.

1929 6.12.1 Parameter \$className

1930 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
 1931 value of ClassificationNodes.

1932 6.12.2 Example of ImmediateInheritedObjectProperties Discovery Query

1933 The following example illustrates how to find all the object properties of a given classification node having
 1934 a name containing "AirServices" including the ones inherited from its immediate super classes.

```

1935
1936 <rs:RequestSlotList>
1937     <rim:Slot
1938         name="urn:oasis:names:tc:ebxml-
1939     regrep:3.0:rs:AdhocQueryRequest:queryId">
1940         <rim:ValueList>
1941             <rim:Value>urn:oasis:names:tc:ebxml-
1942     regrep:query:FindImmediateInheritedObjectProperties</rim:Value>
1943         </rim:ValueList>
1944     </rim:Slot>
1945     <rim:Slot name="urn:oasis:names:tc:ebxml-
1946     regrep:rs:AdhocQueryRequest:queryId">
1947         <rim:ValueList>
1948             <rim:Value>urn:oasis:names:tc:ebxml-
1949     regrep:query:FindImmediateInheritedObjectProperties</rim:Value>
1950         </rim:ValueList>
1951     </rim:Slot>
1952     <rim:Slot name="$className">
1953         <rim:ValueList>
1954             <rim:Value>%AirServices%</rim:Value>
1955         </rim:ValueList>
1956     </rim:Slot>
1957 </rs:RequestSlotList>
1958
1959 <query:ResponseOption returnComposedObjects="true"
1960     returnType="LeafClassWithRepositoryItem"/>
1961
1962 <rim:AdhocQuery id="temporaryId">
1963     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1964     regrep:QueryLanguage:SQL-92">
1965     </rim:QueryExpression>
1966 </rim:AdhocQuery>

```


1967

Example of ImmediateInheritedObjectProperties Discovery Query

1968 6.13 AllInheritedObjectProperties Discovery Query

1969 It should be noted that, given a class, finding the object properties inherited from immediate super classes
1970 is necessary but not sufficient. Given a class, it should be possible to retrieve all of the object properties
1971 inherited from its super classes. This requires a recursion mechanism in SQL queries. The freebXML
1972 implementation allows various relational database products such as Oracle, PostgreSQL and MS SQL
1973 Server 2005 to be used as the database. These products have different support for recursion in SQL
1974 Queries.

1975 The AllInheritedObjectProperties discovery query MAY be implemented by an ebXML Registry
1976 implementing this profile. It allows the discovery of all inherited ObjectProperties recursively of a given
1977 ClassificationNode in a ClassificationScheme in freebXML Registry implementations using MS SQL
1978 Server 2005 as the database.

1979 6.13.1 Parameter \$className

1980 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
1981 value of ClassificationNodes.

1982 6.13.2 Example of AllInheritedObjectProperties Discovery Query

1983 The following example illustrates how to find all the object properties of a given classification node having
1984 a name containing "AirReservationServices" including the ones inherited from all of its super classes
1985 recursively, if the query is implemented as an Adhoc Query.

1986

```

1987 <rs:RequestSlotList>
1988   <rim:Slot
1989     name="urn:oasis:names:tc:ebxml-
1990   regrep:3.0:rs:AdhocQueryRequest:queryId">
1991     <rim:ValueList>
1992       <rim:Value>urn:oasis:names:tc:ebxml-
1993   regrep:query:FindAllInheritedObjectProperties</rim:Value>
1994     </rim:ValueList>
1995   </rim:Slot>
1996   <rim:Slot name="urn:oasis:names:tc:ebxml-
1997   regrep:rs:AdhocQueryRequest:queryId">
1998     <rim:ValueList>
1999       <rim:Value>urn:oasis:names:tc:ebxml-regrep:query:FindAll
2000   InheritedObjectProperties</rim:Value>
2001     </rim:ValueList>
2002   </rim:Slot>
2003   <rim:Slot name="$className">
2004     <rim:ValueList>
2005       <rim:Value>%AirReservationServices%</rim:Value>
2006     </rim:ValueList>
2007   </rim:Slot>
2008 </rs:RequestSlotList>
2009
2010 <query:ResponseOption returnComposedObjects="true"
2011   returnType="LeafClassWithRepositoryItem"/>
2012
2013 <rim:AdhocQuery id="temporaryId">
2014   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2015   regrep:QueryLanguage:SQL-92">
2016     </rim:QueryExpression>
2017 </rim:AdhocQuery>

```

2018

Example of AllInheritedObjectProperties Discovery Query

2019 6.14 DatatypeProperties Discovery Query

2020 The DatatypeProperties discovery query MUST be implemented by an ebXML Registry implementing this
2021 profile. It allows the discovery of all of the datatypeProperties of a given classification node.

2022 6.14.1 Parameter \$className

2023 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2024 value of ClassificationNodes.

2025 6.14.2 Example of DatatypeProperties Discovery Query

2026 The following example illustrates how to find all the datatype properties of a given classification node
2027 having a name containing "AirReservationServices" .

2028

```
2029 <rs:RequestSlotList>  
2030   <rim:Slot  
2031     name="urn:oasis:names:tc:ebxml-  
2032     regrep:3.0:rs:AdhocQueryRequest:queryId">  
2033     <rim:ValueList>  
2034       <rim:Value>urn:oasis:names:tc:ebxml-  
2035       regrep:query:FindDatatypeProperties</rim:Value>  
2036     </rim:ValueList>  
2037   </rim:Slot>  
2038   <rim:Slot name="urn:oasis:names:tc:ebxml-  
2039   regrep:rs:AdhocQueryRequest:queryId">  
2040     <rim:ValueList>  
2041       <rim:Value>urn:oasis:names:tc:ebxml-  
2042       regrep:query:FindDatatypeProperties</rim:Value>  
2043     </rim:ValueList>  
2044   </rim:Slot>  
2045   <rim:Slot name="$className">  
2046     <rim:ValueList>  
2047       <rim:Value>%AirReservationServices%</rim:Value>  
2048     </rim:ValueList>  
2049   </rim:Slot>  
2050 </rs:RequestSlotList>  
2051  
2052 <query:ResponseOption returnComposedObjects="true"  
2053   returnType="LeafClassWithRepositoryItem"/>  
2054  
2055 <rim:AdhocQuery id="temporaryId">  
2056   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
2057   regrep:QueryLanguage:SQL-92">  
2058     </rim:QueryExpression>  
2059 </rim:AdhocQuery>
```

2060

Example of DatatypeProperties Discovery Query

2061 6.15 AllInheritedDatatypeProperties Discovery Query

2062 It should be noted that, given a class, finding the datatype properties inherited from immediate super
2063 classes is necessary but not sufficient. Given a class, it should be possible to retrieve all of the datatype
2064 properties inherited from its super classes. This requires a recursion mechanism in SQL queries. The
2065 freebXML implementation allows various relational database products such as Oracle, PostgreSQL and
2066 MS SQL Server 2005 to be used as the database. These products have different support for recursion in
2067 SQL Queries.

2068 The AllInheritedDatatypeProperties discovery query MAY be implemented by an ebXML Registry
2069 implementing this profile. It allows the discovery of all inherited DatatypeProperties recursively of a given
2070 ClassificationNode in a ClassificationScheme in freebXML Registry implementations using MS SQL
2071 Server 2005 as the database.

2072 6.15.1 Parameter \$className

2073 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2074 value of ClassificationNodes.

2075 6.15.2 Example of AllInheritedDatatypeProperties Discovery Query

2076 The following example illustrates how to find all the datatype properties of a given classification node
2077 having a name containing "AirReservationServices" including the ones inherited from all of its super
2078 classes recursively, if the query is implemented as an Adhoc Query.

2079

```
2080 <rs:RequestSlotList>  
2081   <rim:Slot  
2082     name="urn:oasis:names:tc:ebxml-  
2083   regrep:3.0:rs:AdhocQueryRequest:queryId">  
2084     <rim:ValueList>  
2085       <rim:Value>urn:oasis:names:tc:ebxml-  
2086   regrep:query:FindAllInheritedDatatypeProperties</rim:Value>  
2087     </rim:ValueList>  
2088   </rim:Slot>  
2089   <rim:Slot name="urn:oasis:names:tc:ebxml-  
2090   regrep:rs:AdhocQueryRequest:queryId">  
2091     <rim:ValueList>  
2092       <rim:Value>urn:oasis:names:tc:ebxml-  
2093   regrep:query:FindAllInheritedDatatypeProperties</rim:Value>  
2094     </rim:ValueList>  
2095   </rim:Slot>  
2096   <rim:Slot name="$className">  
2097     <rim:ValueList>  
2098       <rim:Value>%AirReservationServices %</rim:Value>  
2099     </rim:ValueList>  
2100   </rim:Slot>  
2101 </rs:RequestSlotList>  
  
2102  
2103 <query:ResponseOption returnComposedObjects="true"  
2104   returnType="LeafClassWithRepositoryItem"/>  
2105  
2106 <rim:AdhocQuery id="temporaryId">  
2107   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
2108   regrep:QueryLanguage:SQL-92">  
2109     </rim:QueryExpression>  
2110 </rim:AdhocQuery>
```

2111 Example of AllInheritedDatatypeProperties Discovery Query

2112 6.16 TransitiveRelationships Discovery Query

2113 To make any use of the transitive property in ebXML registries, coding is necessary to find out the implied
2114 information. The TransitiveRelationships discovery query MUST be implemented by an ebXML Registry
2115 implementing this profile to handle this semantics.

2116 Given a class which is a source of a transitive property, this discovery query retrieves not only the target
2117 objects of a given transitive property, but if the target objects have the same property, it retrieves their
2118 target objects too.

2119 6.16.1 Parameter \$className

2120 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2121 value of ClassificationNodes.

2122 6.16.2 Parameter \$propertyName

2123 This parameter's value SHALL specify a string containing a pattern match against the name attribute

2124 value of Associations that have associationType of Property

2125 6.16.3 Example of TransitiveRelationships Discovery Query

2126 The following example illustrates how to retrieve all the target objects of the “succeeds” property of the
2127 “AirReservationServices” including the target objects implied by a transitive property relationship.

2128

```
2129 <rs:RequestSlotList>
2130   <rim:Slot
2131     name="urn:oasis:names:tc:ebxml-
2132   regrep:3.0:rs:AdhocQueryRequest:queryId">
2133     <rim:ValueList>
2134       <rim:Value>urn:oasis:names:tc:ebxml-
2135   regrep:query:FindTransitiveRelationships</rim:Value>
2136     </rim:ValueList>
2137   </rim:Slot>
2138   <rim:Slot name="urn:oasis:names:tc:ebxml-
2139   regrep:rs:AdhocQueryRequest:queryId">
2140     <rim:ValueList>
2141       <rim:Value>urn:oasis:names:tc:ebxml-
2142   regrep:query:FindTransitiveRelationships</rim:Value>
2143     </rim:ValueList>
2144   </rim:Slot>
2145   <rim:Slot name="$className">
2146     <rim:ValueList>
2147       <rim:Value>%AirReservationServices%</rim:Value>
2148     </rim:ValueList>
2149   </rim:Slot>
2150   <rim:Slot name="$propertyName">
2151     <rim:ValueList>
2152       <rim:Value>%succeeds%</rim:Value>
2153     </rim:ValueList>
2154   </rim:Slot>
2155 </rs:RequestSlotList>
2156
2157 <query:ResponseOption returnComposedObjects="true"
2158   returnType="LeafClassWithRepositoryItem"/>
2159
2160 <rim:AdhocQuery id="temporaryId">
2161   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2162   regrep:QueryLanguage:SQL-92">
2163     </rim:QueryExpression>
2164 </rim:AdhocQuery>
```

2165 Example of TransitiveRelationships Discovery Query

2166 6.17 TargetObjects Discovery Query

2167 The TargetObjects discovery query MUST be implemented by an ebXML Registry implementing this
2168 profile. It allows the discovery of the targetObjects from the Registry, given a Classification Node
2169 (sourceObject) and a property name (Association Type).

2170 6.17.1 Parameter \$className

2171 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2172 value of ClassificationNodes.

2173 6.17.2 Parameter \$propertyName

2174 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2175 value of Associations that have associationType of Property.

2176 6.17.3 Example of TargetObjects Discovery Query

2177 The following example illustrates how to retrieve all the target objects of the "paymentMethod" property of
2178 the "AirReservationServices".

2179

```
2180 <rs:RequestSlotList>
2181   <rim:Slot
2182     name="urn:oasis:names:tc:ebxml-
2183     regrep:3.0:rs:AdhocQueryRequest:queryId">
2184     <rim:ValueList>
2185       <rim:Value>urn:oasis:names:tc:ebxml-
2186       regrep:query:FindTargetObjects</rim:Value>
2187     </rim:ValueList>
2188   </rim:Slot>
2189   <rim:Slot name="urn:oasis:names:tc:ebxml-
2190   regrep:rs:AdhocQueryRequest:queryId">
2191     <rim:ValueList>
2192       <rim:Value>urn:oasis:names:tc:ebxml-
2193       regrep:query:FindTargetObjects</rim:Value>
2194     </rim:ValueList>
2195   </rim:Slot>
2196   <rim:Slot name="$className">
2197     <rim:ValueList>
2198       <rim:Value>%AirReservationServices%</rim:Value>
2199     </rim:ValueList>
2200   </rim:Slot>
2201   <rim:Slot name="$propertyName">
2202     <rim:ValueList>
2203       <rim:Value>%paymentMethod%</rim:Value>
2204     </rim:ValueList>
2205   </rim:Slot>
2206 </rs:RequestSlotList>
2207
2208 <query:ResponseOption returnComposedObjects="true"
2209   returnType="LeafClassWithRepositoryItem"/>
2210
2211 <rim:AdhocQuery id="temporaryId">
2212   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2213   regrep:QueryLanguage:SQL-92">
2214     </rim:QueryExpression>
2215 </rim:AdhocQuery>
```

2216 Example of TargetObjects Discovery Query

2217

2218 6.18 TargetObjectsInverseOf Discovery Query

2219 The TargetObjectsInverseOf discovery query MUST be implemented by an ebXML Registry implementing
2220 this profile. Given a Classification Node (sourceObject) and a property name (Association Type), this
2221 query retrieves the source objects of the properties which are stated to be inverseOf the property name
2222 given as a parameter, and considering the Classification Node name as the targetObject of these
2223 properties.

2224 6.18.1 Parameter \$className

2225 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2226 value of ClassificationNodes.

2227 6.18.2 Parameter \$propertyName

2228 This parameter's value SHALL specify a string containing a pattern match against the name attribute
2229 value of Associations that have associationType of Property.

2230 6.18.3 Example of TargetObjectsInverseOf Discovery Query

2231 The following example illustrates how to retrieve all the source objects of the properties which are stated
2232 to the the inverseOf the property “succeeds”, considering the “AirReservationServices” as the target object
2233 of these properties.

2234

```
2235 <rs:RequestSlotList>
2236   <rim:Slot
2237     name="urn:oasis:names:tc:ebxml-
2238   regrep:3.0:rs:AdhocQueryRequest:queryId">
2239     <rim:ValueList>
2240       <rim:Value>urn:oasis:names:tc:ebxml-
2241   regrep:query:FindTOinverseOf</rim:Value>
2242     </rim:ValueList>
2243   </rim:Slot>
2244   <rim:Slot name="urn:oasis:names:tc:ebxml-
2245   regrep:rs:AdhocQueryRequest:queryId">
2246     <rim:ValueList>
2247       <rim:Value>urn:oasis:names:tc:ebxml-
2248   regrep:query:FindTOinverseOf</rim:Value>
2249     </rim:ValueList>
2250   </rim:Slot>
2251   <rim:Slot name="$className">
2252     <rim:ValueList>
2253       <rim:Value>%AirReservationServices%</rim:Value>
2254     </rim:ValueList>
2255   </rim:Slot>
2256   <rim:Slot name="$propertyName">
2257     <rim:ValueList>
2258       <rim:Value>%succeeds%</rim:Value>
2259     </rim:ValueList>
2260   </rim:Slot>
2261 </rs:RequestSlotList>
2262
2263 <query:ResponseOption returnComposedObjects="true"
2264   returnType="LeafClassWithRepositoryItem"/>
2265
2266 <rim:AdhocQuery id="temporaryId">
2267   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2268   regrep:QueryLanguage:SQL-92">
2269     </rim:QueryExpression>
2270 </rim:AdhocQuery>
```

2271 Example of TargetObjectsInverseOf Discovery Query

2272

2273 6.19 InverseRanges Discovery Query

2274 The InverseRanges discovery query MUST be implemented by an ebXML Registry implementing this
2275 profile to handle this semantics. Given a Classification Node (sourceObject) and a property name
2276 (Association Type), this query retrieves not only the target objects of this property, but also the source
2277 objects of the properties which are stated to be inverseOf the property name given as a parameter, and
2278 considering the Classification Node name as the targetObject of these properties. This query can be
2279 thought as the union of the queries presented in Sections 6.17 and 6.18.

2280 6.19.1 Parameter \$className

2281 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2282 value of ClassificationNodes.

2283 6.19.2 Parameter \$propertyName

2284 This parameter's value SHALL specify a string containing a pattern match against the name attribute

2285 value of Associations that have associationType of Property

2286 6.19.3 Example of InverseRanges Discovery Query

2287 Consider, for example, the "succeeds" property defined in Section 4.3.3. To denote that a certain Web
2288 service instance precedes another during execution, we may define the "precedes" property as an inverse
2289 of the "succeeds" property as follows:

2290

```
2291 <owl:ObjectProperty rdf:ID="precedes">  
2292   <owl:inverseOf rdf:resource="#succeeds" />  
2293 </owl:ObjectProperty>
```

2294 **Example owl:inverseOf Property**

2295 Assume that we want to find all the Web services which can succeed a given Web service. In such a
2296 case, we need not only find all the Web services which succeeds this given Web service, that is the target
2297 objects of "succeeds" Association instance, but we also need to find all the sourceObjects of the
2298 "precedes" Association instance since "precedes" is declared to be the "inverseOf" succeeds Association
2299 instance.

2300 The following example illustrates how to retrieve all the services that "succeeds" "AirReservationServices"
2301 by also making use of its "precedes" property.

2302

```
2303 <rs:RequestSlotList>  
2304   <rim:Slot  
2305     name="urn:oasis:names:tc:ebxml-  
2306   regrep:3.0:rs:AdhocQueryRequest:queryId">  
2307     <rim:ValueList>  
2308       <rim:Value>urn:oasis:names:tc:ebxml-  
2309   regrep:query:FindInverseRanges</rim:Value>  
2310     </rim:ValueList>  
2311   </rim:Slot>  
2312   <rim:Slot name="urn:oasis:names:tc:ebxml-  
2313   regrep:rs:AdhocQueryRequest:queryId">  
2314     <rim:ValueList>  
2315       <rim:Value>urn:oasis:names:tc:ebxml-  
2316   regrep:query:FindInverseRanges</rim:Value>  
2317     </rim:ValueList>  
2318   </rim:Slot>  
2319   <rim:Slot name="$className">  
2320     <rim:ValueList>  
2321       <rim:Value>%AirReservationServices%</rim:Value>  
2322     </rim:ValueList>  
2323   </rim:Slot>  
2324   <rim:Slot name="$propertyName">  
2325     <rim:ValueList>  
2326       <rim:Value>%succeeds%</rim:Value>  
2327     </rim:ValueList>  
2328   </rim:Slot>  
2329 </rs:RequestSlotList>  
2330  
2331 <query:ResponseOption returnComposedObjects="true"  
2332   returnType="LeafClassWithRepositoryItem"/>  
2333  
2334 <rim:AdhocQuery id="temporaryId">  
2335   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
2336   regrep:QueryLanguage:SQL-92">  
2337     </rim:QueryExpression>  
2338 </rim:AdhocQuery>
```

2339

Example of InverseRanges Discovery Query

2340 6.20 SymmetricProperties Discovery Query

2341 The SymmetricProperties discovery query MUST be implemented by an ebXML Registry implementing
2342 this profile. It allows the discovery of all of the Symmetric Properties of a given classification node.

2343 6.20.1 Parameter \$className

2344 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2345 value of ClassificationNodes.

2346 6.20.2 Example of SymmetricProperties Discovery Query

2347 The following example illustrates how to find all the symmetric properties of a given classification node
2348 having a name containing "AirReservationServices" .

2349

```
2350 <rs:RequestSlotList>
2351   <rim:Slot
2352     name="urn:oasis:names:tc:ebxml-
2353   regrep:3.0:rs:AdhocQueryRequest:queryId">
2354     <rim:ValueList>
2355       <rim:Value>urn:oasis:names:tc:ebxml-
2356   regrep:query:FindSymmetricProperties</rim:Value>
2357     </rim:ValueList>
2358   </rim:Slot>
2359   <rim:Slot name="urn:oasis:names:tc:ebxml-
2360   regrep:rs:AdhocQueryRequest:queryId">
2361     <rim:ValueList>
2362       <rim:Value>urn:oasis:names:tc:ebxml-
2363   regrep:query:FindSymmetricProperties</rim:Value>
2364     </rim:ValueList>
2365   </rim:Slot>
2366   <rim:Slot name="$className">
2367     <rim:ValueList>
2368       <rim:Value>%AirReservationServices%</rim:Value>
2369     </rim:ValueList>
2370   </rim:Slot>
2371 </rs:RequestSlotList>
2372
2373 <query:ResponseOption returnComposedObjects="true"
2374   returnType="LeafClassWithRepositoryItem"/>
2375
2376 <rim:AdhocQuery id="temporaryId">
2377   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2378   regrep:QueryLanguage:SQL-92">
2379     </rim:QueryExpression>
2380 </rim:AdhocQuery>
```

2381 Example of SymmetricProperties Discovery Query

2382 6.21 FunctionalProperties Discovery Query

2383 The FunctionalProperties discovery query MUST be implemented by an ebXML Registry implementing
2384 this profile. It allows the discovery of all of the Functional Properties of a given classification node.

2385 6.21.1 Parameter \$className

2386 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2387 value of ClassificationNodes.

2388 6.21.2 Example of FunctionalProperties Discovery Query

2389 The following example illustrates how to find all the functional properties of a given classification node
2390 having a name containing "AirReservationServices" .

2391

```
2392 <rs:RequestSlotList>
2393   <rim:Slot
2394     name="urn:oasis:names:tc:ebxml-
2395   regrep:3.0:rs:AdhocQueryRequest:queryId">
2396     <rim:ValueList>
2397       <rim:Value>urn:oasis:names:tc:ebxml-
2398   regrep:query:FindFunctionalProperties</rim:Value>
2399     </rim:ValueList>
2400   </rim:Slot>
2401   <rim:Slot name="urn:oasis:names:tc:ebxml-
2402   regrep:rs:AdhocQueryRequest:queryId">
2403     <rim:ValueList>
2404       <rim:Value>urn:oasis:names:tc:ebxml-
2405   regrep:query:FindFunctionalProperties</rim:Value>
2406     </rim:ValueList>
2407   </rim:Slot>
2408   <rim:Slot name="$className">
2409     <rim:ValueList>
2410       <rim:Value>%AirReservationServices%</rim:Value>
2411     </rim:ValueList>
2412   </rim:Slot>
2413 </rs:RequestSlotList>
2414
2415 <query:ResponseOption returnComposedObjects="true"
2416   returnType="LeafClassWithRepositoryItem"/>
2417
2418 <rim:AdhocQuery id="temporaryId">
2419   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2420   regrep:QueryLanguage:SQL-92">
2421     </rim:QueryExpression>
2422 </rim:AdhocQuery>
```

2423

Example of Functional Properties Discovery Query

2424 6.22 InverseFunctionalProperties Discovery Query

2425 The InverseFunctionalProperties discovery query MUST be implemented by an ebXML Registry
2426 implementing this profile. It allows the discovery of all of the Inverse Functional Properties of a given
2427 classification node.

2428 6.22.1 Parameter \$className

2429 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2430 value of ClassificationNodes.

2431 6.22.2 Example of InverseFunctionalProperties Discovery Query

2432 The following example illustrates how to find all the inverse functional properties of a given classification
2433 node having a name containing "AirReservationServices".

2434

```
2435 <rs:RequestSlotList>
2436   <rim:Slot
2437     name="urn:oasis:names:tc:ebxml-
2438   regrep:3.0:rs:AdhocQueryRequest:queryId">
2439     <rim:ValueList>
2440       <rim:Value>urn:oasis:names:tc:ebxml-
2441   regrep:query:FindInverseFunctionalProperties</rim:Value>
2442     </rim:ValueList>
2443   </rim:Slot>
2444   <rim:Slot name="urn:oasis:names:tc:ebxml-
2445   regrep:rs:AdhocQueryRequest:queryId">
2446     <rim:ValueList>
```



```

2447         <rim:Value>urn:oasis:names:tc:ebxml-
2448 regrep:query:FindInverseFunctionalProperties</rim:Value>
2449     </rim:ValueList>
2450 </rim:Slot>
2451 <rim:Slot name="$className">
2452     <rim:ValueList>
2453         <rim:Value>%AirReservationServices%</rim:Value>
2454     </rim:ValueList>
2455 </rim:Slot>
2456 </rs:RequestSlotList>
2457
2458 <query:ResponseOption returnComposedObjects="true"
2459     returnType="LeafClassWithRepositoryItem"/>
2460
2461 <rim:AdhocQuery id="temporaryId">
2462     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2463 regrep:QueryLanguage:SQL-92">
2464     </rim:QueryExpression>
2465 </rim:AdhocQuery>

```

2466 Example of InverseFunctional Properties Discovery Query

2467 6.23 Instances Discovery Query

2468 When an intersection definition is used to create a complex class (a new ClassificationNode) in RIM as
2469 described in Section 4.6, it becomes possible to infer that the objects (instances) classified by both of the
2470 classes (ClassificationNodes) constituting the intersection are also the instances of this complex class.

2471 The Instances discovery query MUST be implemented by an ebXML Registry implementing this profile. It
2472 allows the discovery of all of the direct instances of a given classification node and if it is a complex class
2473 which is an intersection two classes, it also allows to retrieve the intersection of the instances of both of
2474 the classes involved in the intersection definition.

2475 6.23.1 Parameter \$className

2476 This parameter's value SHALL specify a string containing a pattern to match against the name attribute
2477 value of ClassificationNodes.

2478 6.23.2 Example of Instances Discovery Query

2479 Consider the "AirReservationServices" definition presented in Section 4.6. The following example
2480 illustrates how to find all the direct instances of the "AirReservationServices" and also the instances
2481 classified by both "AirServices" and also the "ReservationServices".

```

2482
2483 <rs:RequestSlotList>
2484     <rim:Slot
2485         name="urn:oasis:names:tc:ebxml-
2486 regrep:3.0:rs:AdhocQueryRequest:queryId">
2487         <rim:ValueList>
2488             <rim:Value>urn:oasis:names:tc:ebxml-
2489 regrep:query:FindInstances</rim:Value>
2490         </rim:ValueList>
2491     </rim:Slot>
2492     <rim:Slot name="urn:oasis:names:tc:ebxml-
2493 regrep:rs:AdhocQueryRequest:queryId">
2494         <rim:ValueList>
2495             <rim:Value>urn:oasis:names:tc:ebxml-
2496 regrep:query:FindInstances</rim:Value>
2497         </rim:ValueList>
2498     </rim:Slot>
2499     <rim:Slot name="$className">
2500         <rim:ValueList>
2501             <rim:Value>%AirReservationServices%</rim:Value>
2502         </rim:ValueList>

```

```
2503     </rim:Slot>
2504 </rs:RequestSlotList>
2505
2506 <query:ResponseOption returnComposedObjects="true"
2507     returnType="LeafClassWithRepositoryItem"/>
2508
2509 <rim:AdhocQuery id="temporaryId">
2510     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2511     regrep:QueryLanguage:SQL-92">
2512         </rim:QueryExpression>
2513     </rim:AdhocQuery>
```

2514

Example of Instances Discovery Query

2515 7 Canonical Metadata Definitions

2516 This chapter specifies the canonical metadata defined by this profile.

2517 7.1 ObjectType Extensions

2518 The following new extensions to the canonical ObjectType ClassificationScheme are described by this
2519 profile:

2520

```
2521 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2522 regrep:ObjectType:RegistryObject:ExtrinsicObject"  
2523 lid="urn:oasis:names:tc:ebxml-  
2524 regrep:ObjectType:RegistryObject:ExtrinsicObject:OWL" code="OWL"  
2525 id="urn:oasis:names:tc:ebxml-  
2526 regrep:ObjectType:RegistryObject:ExtrinsicObject:OWL">  
2527 <rim:Name>  
2528 <rim:LocalizedString charset="UTF-8" value="label.OWL"/>  
2529 </rim:Name>  
2530 </rim:ClassificationNode>
```

2531 7.2 AssociationType Extensions

2532 The following new extensions to the AssociationType ClassificationScheme are described by this profile:

2533

```
2534 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2535 regrep:classificationScheme:AssociationType"  
2536 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:ObjectProperty"  
2537 code="ObjectProperty" id="urn:oasis:names:tc:ebxml-  
2538 regrep:AssociationType:ObjectProperty">  
2539 <rim:Name>  
2540 <rim:LocalizedString charset="UTF-8"  
2541 value="ObjectProperty"/>  
2542 </rim:Name>  
2543 </rim:ClassificationNode>  
2544 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2545 regrep:classificationScheme:AssociationType"  
2546 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:Property"  
2547 code="Property" id="urn:oasis:names:tc:ebxml-  
2548 regrep:AssociationType:Property">  
2549 <rim:Name>  
2550 <rim:LocalizedString charset="UTF-8" value="Property"/>  
2551 </rim:Name>  
2552 </rim:ClassificationNode>  
2553 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2554 regrep:classificationScheme:AssociationType"  
2555 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:SubPropertyOf"  
2556 code="SubPropertyOf" id="urn:oasis:names:tc:ebxml-  
2557 regrep:AssociationType:SubPropertyOf">  
2558 <rim:Name>  
2559 <rim:LocalizedString charset="UTF-8" value="SubPropertyOf"/>  
2560 </rim:Name>  
2561 </rim:ClassificationNode>  
2562 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2563 regrep:classificationScheme:AssociationType"  
2564 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:SubClassOf"  
2565 code="SubClassOf" id="urn:oasis:names:tc:ebxml-  
2566 regrep:AssociationType:SubClassOf">  
2567 <rim:Name>  
2568 <rim:LocalizedString charset="UTF-8" value="SubClassOf"/>  
2569 </rim:Name>  
2570 </rim:ClassificationNode>  
2571
```

```

2572 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2573 regrep:classificationScheme:AssociationType"
2574 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:IntersectionOf"
2575 code="IntersectionOf" id="urn:oasis:names:tc:ebxml-
2576 regrep:AssociationType:IntersectionOf">
2577   <rim:Name>
2578     <rim:LocalizedString charset="UTF-8"
2579 value="IntersectionOf"/>
2580   </rim:Name>
2581 </rim:ClassificationNode>
2582
2583 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2584 regrep:classificationScheme:AssociationType"
2585 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:SameAs"
2586 code="SameAs" id="urn:oasis:names:tc:ebxml-
2587 regrep:AssociationType:SameAs">
2588   <rim:Name>
2589     <rim:LocalizedString charset="UTF-8" value="SameAs"/>
2590   </rim:Name>
2591 </rim:ClassificationNode>
2592
2593 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2594 regrep:classificationScheme:AssociationType"
2595 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:restriction"
2596 code="restriction" id="urn:oasis:names:tc:ebxml-
2597 regrep:AssociationType:restriction">
2598   <rim:Name>
2599     <rim:LocalizedString charset="UTF-8" value="restriction"/>
2600   </rim:Name>
2601 </rim:ClassificationNode>
2602
2603 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2604 regrep:classificationScheme:AssociationType"
2605 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:DifferentFrom"
2606 code="DifferentFrom" id="urn:oasis:names:tc:ebxml-
2607 regrep:AssociationType:DifferentFrom">
2608   <rim:Name>
2609     <rim:LocalizedString charset="UTF-8" value="DifferentFrom"/>
2610   </rim:Name>
2611 </rim:ClassificationNode>
2612
2613 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2614 regrep:classificationScheme:AssociationType"
2615 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:DatatypeProperty"
2616 code="DatatypeProperty" id="urn:oasis:names:tc:ebxml-
2617 regrep:AssociationType:DatatypeProperty">
2618   <rim:Name>
2619     <rim:LocalizedString charset="UTF-8"
2620 value="DatatypeProperty"/>
2621   </rim:Name>
2622 </rim:ClassificationNode>
2623
2624 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2625 regrep:classificationScheme:AssociationType"
2626 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:TransitiveProperty"
2627 code="TransitiveProperty" id="urn:oasis:names:tc:ebxml-
2628 regrep:AssociationType:TransitiveProperty">
2629   <rim:Name>
2630     <rim:LocalizedString charset="UTF-8"
2631 value="TransitiveProperty"/>
2632   </rim:Name>
2633 </rim:ClassificationNode>
2634

```

```

2635 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2636 regrep:classificationScheme:AssociationType"
2637 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:InverseOf"
2638 code="InverseOf" id="urn:oasis:names:tc:ebxml-
2639 regrep:AssociationType:InverseOf">
2640   <rim:Name>
2641     <rim:LocalizedString charset="UTF-8" value="InverseOf"/>
2642   </rim:Name>
2643 </rim:ClassificationNode>
2644
2645 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2646 regrep:classificationScheme:AssociationType"
2647 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:SymmetricProperty"
2648 code="SymmetricProperty" id="urn:oasis:names:tc:ebxml-
2649 regrep:AssociationType:SymmetricProperty">
2650   <rim:Name>
2651     <rim:LocalizedString charset="UTF-8"
2652 value="SymmetricProperty"/>
2653   </rim:Name>
2654 </rim:ClassificationNode>
2655
2656 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2657 regrep:classificationScheme:AssociationType"
2658 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:FunctionalProperty"
2659 code="FunctionalProperty" id="urn:oasis:names:tc:ebxml-
2660 regrep:AssociationType:FunctionalProperty">
2661   <rim:Name>
2662     <rim:LocalizedString charset="UTF-8"
2663 value="FunctionalProperty"/>
2664   </rim:Name>
2665 </rim:ClassificationNode>
2666
2667 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2668 regrep:classificationScheme:AssociationType"
2669 lid="urn:oasis:names:tc:ebxml-
2670 regrep:AssociationType:InverseFunctionalProperty"
2671 code="InverseFunctionalProperty" id="urn:oasis:names:tc:ebxml-
2672 regrep:AssociationType:InverseFunctionalProperty">
2673   <rim:Name>
2674     <rim:LocalizedString charset="UTF-8"
2675 value="InverseFunctionalProperty"/>
2676   </rim:Name>
2677 </rim:ClassificationNode>

```

2678 Extensions to the AssociationType ClassificationScheme

2679 7.3 Canonical Queries

2680 The following new canonical queries are described by this profile. Note that while these queries are
2681 complex, the complexity is hidden from clients by exposing only the query parameters to them.

2682 7.3.1 All SuperProperties Discovery Query

2683 Since recursion is not supported by SQL-92, the stored procedure for this query is presented in this
2684 section.

```

2685 CREATE PROCEDURE findAllSuperProperties
2686   @propertyName varchar(50)
2687 AS
2688 WITH
2689   Parents(superPropertyID) AS
2690   (
2691     SELECT A3.id
2692     FROM Association A1, Association A2, Association A3, Name_ N
2693     WHERE A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
2694 regrep:AssociationType:SubPropertyOf' AND

```

```

2695         A1.id = N.parent AND N.value LIKE @propertyName AND
2696         A2.sourceObject = A1.id AND A2.targetObject = A3.id
2697     UNION ALL
2698         SELECT A.targetObject
2699         FROM Association A JOIN Parents P
2700         ON P.superPropertyID = A.sourceObject
2701         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2702 regrep:AssociationType:SubPropertyOf'
2703     )
2704     SELECT * FROM Parents
2705     GO

```

2706 **Recursive stored procedure for MS SQL Server 2005 retrieving all super properties of a given**
2707 **property (Association)**

2708 7.3.2 Immediate SuperClass Discovery Query

```

2709     <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2710 regrep:query:FindImmediateSuperClasses" id="urn:oasis:names:tc:ebxml-
2711 regrep:query:FindImmediateSuperClasses">
2712     <rim:Name>
2713         <rim:LocalizedString
2714 value="label.FindImmediateSuperClasses"/>
2715     </rim:Name>
2716     <rim:Description>
2717         <rim:LocalizedString
2718 value="label.FindImmediateSuperClasses.desc"/>
2719     </rim:Description>
2720     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2721 regrep:QueryLanguage:SQL-92">
2722         SELECT C2.*
2723         FROM ClassificationNode C2, Association A, Name_ N,
2724 ClassificationNode C1
2725         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2726 regrep:AssociationType:SubClassOf' AND
2727         C1.id = N.parent AND
2728         N.value LIKE '$className' AND
2729         A.sourceObject = C1.id AND
2730         A.targetObject = C2.id
2731     </rim:QueryExpression>
2732 </rim:AdhocQuery>
2733

```

2734 **The Adhoc Query retrieving immediate super classes of a given classification node**

2735 7.3.3 Immediate SubClass Discovery Query

```

2736 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2737 regrep:query:FindImmediateSubClasses" id="urn:oasis:names:tc:ebxml-
2738 regrep:query:FindImmediateSubClasses">
2739     <rim:Name>
2740         <rim:LocalizedString value="label.FindImmediateSubClasses"/>
2741     </rim:Name>
2742     <rim:Description>
2743         <rim:LocalizedString
2744 value="label.FindImmediateSubClasses.desc"/>
2745     </rim:Description>
2746     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2747 regrep:QueryLanguage:SQL-92">
2748         SELECT C2.*
2749         FROM ClassificationNode C2, Association A, Name_ N,
2750 ClassificationNode C1
2751         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2752 regrep:AssociationType:SubClassOf' AND
2753         C1.id = N.parent AND
2754         N.value LIKE '$className' AND
2755         A.sourceObject = C2.id AND

```

```
2756         A.targetObject = C1.id
2757     </rim:QueryExpression>
2758 </rim:AdhocQuery>
```

2759 **The Adhoc Query retrieving immediate subclasses of a given classification node**

2760 7.3.4 All SuperClasses Discovery Query

2761 Since recursion is not supported by SQL-92, the stored procedure for this query is presented in this
2762 section.

```
2763     CREATE PROCEDURE findAllSuperClasses
2764         @className varchar(50)
2765     AS
2766     WITH
2767         Parents(superClassID) AS
2768         (
2769             SELECT C2.id
2770             FROM Association A, Name_ N, ClassificationNode C1,
2771             ClassificationNode C2
2772             WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2773             regrep:AssociationType:SubClassOf' AND
2774             C1.id = N.parent AND N.value LIKE @className AND
2775             A.sourceObject = C1.id AND A.targetObject = C2.id
2776         UNION ALL
2777             SELECT A.targetObject
2778             FROM Association A JOIN Parents P
2779             ON P.superClassID = A.sourceObject
2780             WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2781             regrep:AssociationType:SubClassOf'
2782         )
2783     SELECT * FROM Parents
2784     GO
```

2785 **Recursive stored procedure for MS SQL Server 2005 database retrieving all super classes of a**
2786 **given classification node**

2787 7.3.5 All SubClasses Discovery Query

2788 Since recursion is not supported by SQL-92, the stored procedure for this query is presented in this
2789 section.

```
2790     CREATE PROCEDURE findAllSubClasses
2791         @className varchar(50)
2792     AS
2793     WITH
2794         Children(subClassID) AS
2795         (
2796             SELECT C1.id
2797             FROM Association A, Name_ N, ClassificationNode C1,
2798             ClassificationNode C2
2799             WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2800             regrep:AssociationType:SubClassOf' AND
2801             C2.id = N.parent AND N.value LIKE @className AND
2802             A.sourceObject = C1.id AND A.targetObject = C2.id
2803         UNION ALL
2804             SELECT A.sourceObject
2805             FROM Association A JOIN Children C
2806             ON C.subClassID = A.targetObject
2807             WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2808             regrep:AssociationType:SubClassOf'
2809         )
2810     SELECT * FROM Children
2811     GO
```

2812 **Recursive stored procedure for MS SQL Server 2005 database retrieving all subclasses of a**
2813 **given classification node**

2814 7.3.6 EquivalentClasses Discovery Query

```
2815 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2816 regrep:query:FindEquivalentClasses" id="urn:oasis:names:tc:ebxml-
2817 regrep:query:FindEquivalentClasses">
2818   <rim:Name>
2819     <rim:LocalizedString value="label.FindEquivalentClasses"/>
2820   </rim:Name>
2821   <rim:Description>
2822     <rim:LocalizedString
2823 value="label.FindEquivalentClasses.desc"/>
2824   </rim:Description>
2825   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2826 regrep:QueryLanguage:SQL-92">
2827     SELECT C2.*
2828     FROM ClassificationNode C2, Association A, Name_ N,
2829 ClassificationNode C
2830     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2831 regrep:AssociationType:EquivalentTo' AND
2832     C.id = N.parent AND
2833     N.value LIKE '$className' AND
2834     A.sourceObject = C.id AND
2835     A.targetObject = C2.id
2836   </rim:QueryExpression>
2837 </rim:AdhocQuery>
```

2838 **Adhoc Query retrieving all the equivalent classes of a given classification node**

2839 7.3.7 EquivalentProperties Discovery Query

```
2840 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2841 regrep:query:FindEquivalentProperties" id="urn:oasis:names:tc:ebxml-
2842 regrep:query:FindEquivalentProperties">
2843   <rim:Name>
2844     <rim:LocalizedString
2845 value="label.FindEquivalentProperties"/>
2846   </rim:Name>
2847   <rim:Description>
2848     <rim:LocalizedString
2849 value="label.FindEquivalentProperties.desc"/>
2850   </rim:Description>
2851   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2852 regrep:QueryLanguage:SQL-92">
2853     SELECT A3.*
2854     FROM Association A3, Association A1, Name_ N, Association
2855 A2
2856     WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
2857 regrep:AssociationType:EquivalentTo' AND
2858     A2.id = N.parent AND
2859     N.value LIKE '$propertyName' AND
2860     A1.sourceObject = A2.id AND
2861     A1.targetObject = A3.id
2862   </rim:QueryExpression>
2863 </rim:AdhocQuery>
```

2864 **Adhoc Query retrieving all the equivalent Association Type of a given Association Type**

2865 7.3.8 SameExtrinsicObjects Discovery Query

```
2866 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2867 regrep:query:FindTheSameExtrinsicObjects" id="urn:oasis:names:tc:ebxml-
2868 regrep:query:FindTheSameExtrinsicObjects">
2869   <rim:Name>
2870     <rim:LocalizedString
2871 value="label.FindTheSameExtrinsicObjects"/>
2872   </rim:Name>
2873   <rim:Description>
```



```

2874         <rim:LocalizedString
2875 value="label.FindTheSameExtrinsicObjects.desc"/>
2876     </rim:Description>
2877     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2878 regrep:QueryLanguage:SQL-92">
2879         SELECT E2.*
2880         FROM ExtrinsicObject E2, Association A, Name_ N,
2881 ExtrinsicObject E
2882         WHERE A.associationType LIKE ''urn:oasis:names:tc:ebxml-
2883 regrep:AssociationType:SameAs'' AND
2884         E.id = N.parent AND
2885         N.value LIKE ''$extrinsicObjectName'' AND
2886         A.sourceObject = E.id AND
2887         A.targetObject = E2.id
2888     </rim:QueryExpression>
2889 </rim:AdhocQuery>

```

2890 **Adhoc Query retrieving all the "ExtrinsicObjects" defined to be the same with a given**
2891 **ExtrinsicObject**

2892 7.3.9 DifferentExtrinsicObjects Discovery Query

```

2893 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2894 regrep:query:FindDifferentExtrinsicObjects" id="urn:oasis:names:tc:ebxml-
2895 regrep:query:FindDifferentExtrinsicObjects">
2896     <rim:Name>
2897         <rim:LocalizedString
2898 value="label.FindDifferentExtrinsicObjects"/>
2899     </rim:Name>
2900     <rim:Description>
2901         <rim:LocalizedString
2902 value="label.FindDifferentExtrinsicObjects.desc"/>
2903     </rim:Description>
2904     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2905 regrep:QueryLanguage:SQL-92">
2906         SELECT E2.*
2907         FROM ExtrinsicObject E2, Association A, Name_ N,
2908 ExtrinsicObject E
2909         WHERE A.associationType LIKE ''urn:oasis:names:tc:ebxml-
2910 regrep:AssociationType:DifferentFrom'' AND
2911         E.id = N.parent AND
2912         N.value LIKE ''$extrinsicObjectName'' AND
2913         A.sourceObject = E.id AND
2914         A.targetObject = E2.id
2915     </rim:QueryExpression>
2916 </rim:AdhocQuery>

```

2917 **Adhoc Query retrieving all the "ExtrinsicObjects" defined to be different from a given**
2918 **ExtrinsicObject**

2919 7.3.10 AllDifferentRegistryObject Discovery Query

```

2920 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2921 regrep:query:FindAllDifferent" id="urn:oasis:names:tc:ebxml-
2922 regrep:query:FindAllDifferent">
2923     <rim:Name>
2924         <rim:LocalizedString value="label.FindAllDifferent"/>
2925     </rim:Name>
2926     <rim:Description>
2927         <rim:LocalizedString value="label.FindAllDifferent.desc"/>
2928     </rim:Description>
2929     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2930 regrep:QueryLanguage:SQL-92">
2931         SELECT RO2.*
2932         FROM RegistryObject RO2, Association A1, Association A2,
2933 Name_ N, RegistryObject RO,
2934 RegistryPackage RP<!--, Slot S-->

```

```

2935         WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
2936 regrep:AssociationType:HasMember' AND
2937         RO.id = N.parent AND
2938         N.value LIKE '$registryObjectName' AND
2939         A1.sourceObject = RP.id AND
2940         <!-- S.parent = RP.id AND
2941         S.name_ LIKE 'allDifferent' AND S.value LIKE 'true' AND
2942 -->
2943         A1.targetObject = RO.id AND
2944         A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
2945 regrep:AssociationType:HasMember' AND
2946         A2.sourceObject = RP.id AND
2947         A2.targetObject != RO.id AND
2948         A2.targetObject = R02.id
2949     </rim:QueryExpression>
2950 </rim:AdhocQuery>

```

2951 **Adhoc Query retrieving all the "RegistryObjects" defined to be different from a given**
2952 **RegistryObject through a "allDifferentFrom" construct**

2953 7.3.11 ObjectProperties Discovery Query

```

2954 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2955 regrep:query:FindObjectProperties" id="urn:oasis:names:tc:ebxml-
2956 regrep:query:FindObjectProperties">
2957     <rim:Name>
2958         <rim:LocalizedString value="label.FindObjectProperties"/>
2959     </rim:Name>
2960     <rim:Description>
2961         <rim:LocalizedString
2962 value="label.FindObjectProperties.desc"/>
2963     </rim:Description>
2964     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2965 regrep:QueryLanguage:SQL-92">
2966         SELECT A.*
2967         FROM Association A, Name_ N, ClassificationNode C
2968         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2969 regrep:AssociationType:ObjectProperty' AND
2970         C.id = N.parent AND
2971         N.value LIKE '$className' AND
2972         A.sourceObject = C.id
2973     </rim:QueryExpression>
2974 </rim:AdhocQuery>

```

2975 **Adhoc Query retrieving all the object properties of a given classification node**

2976 7.3.12 ImmediateInheritedObjectProperties Discovery Query

```

2977 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2978 regrep:query:FindImmediateInheritedObjectProperties"
2979 id="urn:oasis:names:tc:ebxml-
2980 regrep:query:FindImmediateInheritedObjectProperties">
2981     <rim:Name>
2982         <rim:LocalizedString
2983 value="label.FindImmediateInheritedObjectProperties"/>
2984     </rim:Name>
2985     <rim:Description>
2986         <rim:LocalizedString
2987 value="label.FindImmediateInheritedObjectProperties.desc"/>
2988     </rim:Description>
2989     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2990 regrep:QueryLanguage:SQL-92">
2991         SELECT A2.*
2992         FROM Association A, Name_ N, ClassificationNode C1,
2993 ClassificationNode C2, Association A2
2994         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2995 regrep:AssociationType:SubClassOf' AND

```

```

2996         C1.id = N.parent AND
2997         N.value LIKE '$className' AND
2998         A.sourceObject = C1.id AND
2999         A.targetObject = C2.id AND
3000         A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3001 regrep:AssociationType:ObjectProperty' AND
3002         A2.sourceObject=C2.id
3003     </rim:QueryExpression>
3004 </rim:AdhocQuery>

```

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

3007 7.3.13 AllInheritedObjectProperties Discovery Query

3008 Since recursion is not supported by SQL-92, the stored procedure for this query is presented in this
3009 section.

```

3010 CREATE PROCEDURE findAllInheritedObjectProperties
3011     @className varchar(50)
3012 AS
3013 WITH Parents(superClassID) AS (
3014     SELECT C2.id
3015     FROM Association A, Name_ N, ClassificationNode C1,
3016     ClassificationNode C2
3017     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3018 regrep:AssociationType:SubClassOf' AND
3019         C1.id = N.parent AND N.value LIKE @className AND
3020         A.sourceObject = C1.id AND A.targetObject = C2.id
3021     UNION ALL
3022     SELECT A.targetObject
3023     FROM Association A JOIN Parents P
3024     ON P.superClassID = A.sourceObject
3025     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3026 regrep:AssociationType:SubClassOf'
3027 ) SELECT A.id
3028     FROM Association A, Parents P
3029     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3030 regrep:AssociationType:ObjectProperty' AND
3031         A.sourceObject=P.superClassID
3032 UNION
3033
3034 SELECT A.id
3035 FROM Name_ N, ClassificationNode C, Association A
3036 WHERE C.id = N.parent AND N.value LIKE @className AND
3037         A.sourceObject=C.id AND A.associationType LIKE
3038 'urn:oasis:names:tc:ebxml-regrep:AssociationType:ObjectProperty'
3039 GO

```

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

3042 7.3.14 DatatypeProperties Discovery Query

```

3043 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3044 regrep:query:FindDatatypeProperties" id="urn:oasis:names:tc:ebxml-
3045 regrep:query:FindDatatypeProperties">
3046     <rim:Name>
3047         <rim:LocalizedString value="label.FindDatatypeProperties"/>
3048     </rim:Name>
3049     <rim:Description>
3050         <rim:LocalizedString
3051 value="label.FindDatatypeProperties.desc"/>
3052     </rim:Description>
3053     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3054 regrep:QueryLanguage:SQL-92">
3055         SELECT A.*

```

```

3056         FROM Association A, Name_ N, ClassificationNode C
3057         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3058 regrep:AssociationType:DatatypeProperty' AND
3059         C.id = N.parent AND
3060         N.value LIKE '$className' AND
3061         A.sourceObject = C.id
3062     </rim:QueryExpression>
3063 </rim:AdhocQuery>

```

3064 **Adhoc Query retrieving all the datatype properties of a given classification node**

3065 7.3.15 AllInheritedDatatypeProperties Discovery Query

3066 Since recursion is not supported by SQL-92, the stored procedure for this query is presented in this
3067 section.

```

3068 CREATE PROCEDURE findAllInheritedDatatypeProperties
3069     @className varchar(50)
3070 AS
3071 WITH Parents(superClassID) AS (
3072     SELECT C2.id
3073     FROM Association A, Name_ N, ClassificationNode C1,
3074     ClassificationNode C2
3075     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3076 regrep:AssociationType:SubClassOf' AND
3077     C1.id = N.parent AND N.value LIKE @className AND
3078     A.sourceObject = C1.id AND A.targetObject = C2.id
3079 UNION ALL
3080     SELECT A.targetObject
3081     FROM Association A JOIN Parents P
3082     ON P.superClassID = A.sourceObject
3083     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3084 regrep:AssociationType:SubClassOf'
3085 ) SELECT A.id
3086     FROM Association A, Parents P
3087     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3088 regrep:AssociationType:DatatypeProperty' AND
3089     A.sourceObject=P.superClassID
3090 UNION
3091     SELECT A.id
3092     FROM Name_ N, ClassificationNode C, Association A
3093     WHERE C.id = N.parent AND N.value LIKE @className AND
3094     A.sourceObject=C.id AND A.associationType LIKE
3095     'urn:oasis:names:tc:ebxml-regrep:AssociationType:DatatypeProperty'
3096 GO
3097

```

3098 **Recursive stored procedure for MS SQL Server 2005 database retrieving all inherited Datatype**
3099 **Properties of a given classification node**

3100 7.3.16 TransitiveRelationships Discovery Query

```

3101 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3102 regrep:query:FindTransitiveRelationships" id="urn:oasis:names:tc:ebxml-
3103 regrep:query:FindTransitiveRelationships">
3104     <rim:Name>
3105         <rim:LocalizedString
3106 value="label.FindTransitiveRelationships"/>
3107     </rim:Name>
3108     <rim:Description>
3109         <rim:LocalizedString
3110 value="label.FindTransitiveRelationships.desc"/>
3111     </rim:Description>
3112     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3113 regrep:QueryLanguage:SQL-92">
3114         SELECT C.*

```

```

3115         FROM ClassificationNode C, Association A1, Association A2,
3116 Name_ N1, Name_ N2, Name_ N3
3117         WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
3118 regrep:AssociationType:TransitiveProperty' AND
3119         A1.id = N1.parent AND
3120         N1.value LIKE '$propertyName' AND
3121         A1.sourceObject = N3.parent AND
3122         N3.value LIKE '$className' AND
3123         A2.sourceObject = A1.targetObject AND
3124         A2.id = N2.parent AND
3125         N2.value LIKE '$propertyName' AND
3126         A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3127 regrep:AssociationType:TransitiveProperty' AND
3128         A2.targetObject = C.id
3129         <!-- UNION
3130         SELECT C.*
3131         FROM ClassificationNode C, Association A1, Name_ N1, Name_
3132 N3
3133         WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
3134 regrep:AssociationType:TransitiveProperty' AND
3135         A1.id = N1.parent AND
3136         N1.value LIKE '$propertyName' AND
3137         A1.sourceObject = N3.parent AND
3138         N3.value LIKE '$className' AND
3139         A1.targetObject = C.id -->
3140     </rim:QueryExpression>
3141 </rim:AdhocQuery>

```

3142 **Adhoc Query retrieving the objects in transitive relationship with a given object**

3143 7.3.17 TargetObjects Discovery Query

```

3144 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3145 regrep:query:FindTargetObjects" id="urn:oasis:names:tc:ebxml-
3146 regrep:query:FindTargetObjects">
3147     <rim:Name>
3148         <rim:LocalizedString value="label.FindTargetObjects"/>
3149     </rim:Name>
3150     <rim:Description>
3151         <rim:LocalizedString value="label.FindTargetObjects.desc"/>
3152     </rim:Description>
3153     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3154 regrep:QueryLanguage:SQL-92">
3155         SELECT C2.*
3156         FROM ClassificationNode C2, Association A, Name_ N, Name_
3157 N2, ClassificationNode C1
3158         WHERE A.id=N2.parent AND
3159         N2.value LIKE '$propertyName' AND
3160         C1.id = N.parent AND
3161         N.value LIKE '$className' AND
3162         A.sourceObject = C1.id AND
3163         A.targetObject = C2.id
3164     </rim:QueryExpression>
3165 </rim:AdhocQuery>

```

3166 **Adhoc Query retrieving the Target Objects from the Registry, given a Source Object and an**
3167 **Association**

3168 7.3.18 TargetObjectsInverseOf Discovery Query

```

3169 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3170 regrep:query:FindTOinverseOf" id="urn:oasis:names:tc:ebxml-
3171 regrep:query:FindTOinverseOf">
3172     <rim:Name>
3173         <rim:LocalizedString value="label.FindTOinverseOf"/>
3174     </rim:Name>
3175     <rim:Description>

```

```

3176         <rim:LocalizedString value="label.FindTOinverseOf.desc"/>
3177     </rim:Description>
3178     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3179 regrep:QueryLanguage:SQL-92">
3180         SELECT C2.*
3181         FROM ClassificationNode C2, Association A1, Association A2,
3182 Association A3, Name_ N, Name_ N2, ClassificationNode C1
3183         WHERE A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3184 regrep:AssociationType:InverseOf' AND
3185         A1.id = N.parent AND
3186         N.value LIKE '$propertyName' AND
3187         A2.sourceObject = A1.id AND
3188         A2.targetObject = A3.id AND
3189         C1.id = N2.parent AND
3190         N2.value LIKE '$className' AND
3191         A3.targetObject = C1.id AND
3192         A3.sourceObject = C2.id
3193     </rim:QueryExpression>
3194 </rim:AdhocQuery>

```

3195 **Adhoc query retrieving the Source Objects of an Association which is in "inverseOf"**
3196 **relationship to this Association**

3197 7.3.19 InverseRanges Discovery Query

```

3198 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3199 regrep:query:FindInverseRanges" id="urn:oasis:names:tc:ebxml-
3200 regrep:query:FindInverseRanges">
3201     <rim:Name>
3202         <rim:LocalizedString value="label.FindInverseRanges"/>
3203     </rim:Name>
3204     <rim:Description>
3205         <rim:LocalizedString value="label.FindInverseRanges.desc"/>
3206     </rim:Description>
3207     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3208 regrep:QueryLanguage:SQL-92">
3209         <!-- SELECT C2.*
3210         FROM Association A, Name_ N, Name_ N2, ClassificationNode
3211 C1, ClassificationNode C2
3212         WHERE A.id=N2.parent AND
3213         N2.value LIKE '$propertyName' AND
3214         C1.id = N.parent AND
3215         N.value LIKE '$className' AND
3216         A.sourceObject = C1.id AND
3217         A.targetObject = C2.id
3218         UNION -->
3219         SELECT C2.*
3220         FROM ClassificationNode C2, Association A1, Association A2,
3221 Association A3, Name_ N, NAME_ N2, ClassificationNode C1
3222         WHERE A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3223 regrep:AssociationType:InverseOf' AND
3224         A1.id = N.parent AND
3225         N.value LIKE '$propertyName' AND
3226         A2.sourceObject = A1.id AND
3227         A2.targetObject = A3.id AND
3228         C1.id = N2.parent AND
3229         N2.value LIKE '$className' AND
3230         A1.sourceObject = C1.id AND
3231         A3.sourceObject = C2.id
3232     </rim:QueryExpression>
3233 </rim:AdhocQuery>

```

3234 **Adhoc Query Retrieving both the Target Objects of a given Association and the Source**
3235 **Objects of an Association which is in "inverseOf" relationship to this Association**

3236 7.3.20 SymmetricProperties Discovery Query

```
3237 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3238 regrep:query:FindSymmetricProperties" id="urn:oasis:names:tc:ebxml-
3239 regrep:query:FindSymmetricProperties">
3240   <rim:Name>
3241     <rim:LocalizedString value="label.FindSymmetricProperties"/>
3242   </rim:Name>
3243   <rim:Description>
3244     <rim:LocalizedString
3245 value="label.FindSymmetricProperties.desc"/>
3246   </rim:Description>
3247   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3248 regrep:QueryLanguage:SQL-92">
3249     SELECT A.*
3250     FROM Association A, Name_N, ClassificationNode C
3251     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3252 regrep:AssociationType:SymmetricProperty' AND
3253     C.id = N.parent AND
3254     N.value LIKE '$className' AND
3255     A.sourceObject = C.id
3256   </rim:QueryExpression>
3257 </rim:AdhocQuery>
```

3258 **Adhoc Query retrieving all the Symmetric properties of a given classification node**

3259 7.3.21 FunctionalProperties Discovery Query

```
3260 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3261 regrep:query:FindFunctionalProperties" id="urn:oasis:names:tc:ebxml-
3262 regrep:query:FindFunctionalProperties">
3263   <rim:Name>
3264     <rim:LocalizedString
3265 value="label.FindFunctionalProperties"/>
3266   </rim:Name>
3267   <rim:Description>
3268     <rim:LocalizedString
3269 value="label.FindFunctionalProperties.desc"/>
3270   </rim:Description>
3271   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3272 regrep:QueryLanguage:SQL-92">
3273     SELECT A.*
3274     FROM Association A, Name_N, ClassificationNode C
3275     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3276 regrep:AssociationType:FunctionalProperty' AND
3277     C.id = N.parent AND
3278     N.value LIKE '$className' AND
3279     A.sourceObject = C.id
3280   </rim:QueryExpression>
3281 </rim:AdhocQuery>
```

3282 **Adhoc Query retrieving all the Functional properties of a given classification node**

3283 7.3.22 InverseFunctionalProperties Discovery Query

```
3284 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3285 regrep:query:FindInverseFunctionalProperties"
3286 id="urn:oasis:names:tc:ebxml-
3287 regrep:query:FindInverseFunctionalProperties">
3288   <rim:Name>
3289     <rim:LocalizedString
3290 value="label.FindInverseFunctionalProperties"/>
3291   </rim:Name>
3292   <rim:Description>
3293     <rim:LocalizedString
3294 value="label.FindInverseFunctionalProperties.desc"/>
3295   </rim:Description>
```



```

3296     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3297 regrep:QueryLanguage:SQL-92">
3298         SELECT A.*
3299         FROM Association A, Name_N, ClassificationNode C
3300         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3301 regrep:AssociationType:InverseFunctionalProperty' AND
3302         C.id = N.parent AND
3303         N.value LIKE '$className' AND
3304         A.sourceObject = C.id
3305     </rim:QueryExpression>
3306 </rim:AdhocQuery>

```

3307 **Adhoc Query retrieving all the Inverse Functional properties of a given classification node**

3308 7.3.23 Instances Discovery Query Discovery Query

```

3309 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-regrep:query:FindInstances"
3310 id="urn:oasis:names:tc:ebxml-regrep:query:FindInstances">
3311     <rim:Name>
3312         <rim:LocalizedString value="label.FindInstances"/>
3313     </rim:Name>
3314     <rim:Description>
3315         <rim:LocalizedString value="label.FindInstances.desc"/>
3316     </rim:Description>
3317     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3318 regrep:QueryLanguage:SQL-92">
3319         <!-- SELECT S.* FROM Service S, (
3320         SELECT A.targetObject AS id
3321         FROM RegistryPackage R, Association A
3322         WHERE R.id=A.sourceObject AND
3323         A.associationType = 'urn:oasis:names:tc:ebxml-
3324 regrep:AssociationType:HasMember' AND
3325         R.id IN (
3326         SELECT A.targetObject
3327         FROM Association A, Name_N, ClassificationNode C
3328         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3329 regrep:AssociationType:IntersectionOf' AND
3330         C.id = N.parent AND
3331         N.value LIKE '$className' AND
3332         A.sourceObject = C.id
3333         )
3334         ) AS T1, (
3335         SELECT A.targetObject AS id
3336         FROM RegistryPackage R, Association A
3337         WHERE R.id=A.sourceObject AND
3338         A.associationType = 'urn:oasis:names:tc:ebxml-
3339 regrep:AssociationType:HasMember' AND
3340         R.id IN (
3341         SELECT A.targetObject
3342         FROM Association A, Name_N, ClassificationNode C
3343         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3344 regrep:AssociationType:IntersectionOf' AND
3345         C.id = N.parent AND
3346         N.value LIKE '$className' AND
3347         A.sourceObject = C.id
3348         )
3349         ) AS T2
3350         WHERE S.id IN (
3351         SELECT classifiedObject
3352         FROM Classification
3353         WHERE classificationNode=T1.id
3354         INTERSECT
3355         SELECT classifiedObject
3356         FROM Classification
3357         WHERE classificationNode=T2.id
3358         ) AND T1.id!=T2.id
3359         UNION -->

```



```
3360          SELECT S.*
3361          FROM Service S, Classification C, ClassificationNode CN,
3362 Name_ N
3363          WHERE S.id = C. classifiedObject AND
3364          C.classificationNode = CN.id AND
3365          N.value LIKE '$className' AND
3366          N.parent = CN.id
3367          </rim:QueryExpression>
3368 </rim:AdhocQuery>
```

3369 **Adhoc Query Retrieving the instances of intersected classes**

3370 8 OWL Profile References

3371 8.1 Normative References

- 3372 [Bechhofer, Harmelen, Hendler, Horrocks, McGuinness, Patel-Schneider, Stein]
3373 Bechhofer, S., Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., Stein, L.
3374 A., OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004
3375 <http://www.w3.org/TR/2004/REC-owl-ref-20040210/>
3376
- 3377 [Brickley, Guha] Brickley, D., Guha, R.V., RDF Vocabulary Description Language 1.0: RDF Schema
3378 W3C Recommendation 10 February 2004
3379 <http://www.w3.org/TR/rdf-schema/>
3380
- 3381 [DAML+OIL] <http://www.daml.org/>
3382 [ebRIM] ebXML Registry Information Model version 3.0
3383 <http://docs.oasis-open.org/regrep/regrep-rim/v3.0/regrep-rim-3.0-os.pdf>
3384
- 3385 [ebRS] ebXML Registry Services Specification version 3.0
3386 <http://docs.oasis-open.org/regrep/regrep-rs/v3.0/regrep-rs-3.0-os.pdf>
- 3387 [ebRR-DPT] Deployment Profile Template For ebXML Registry 3.0 OASIS Specifications V_0.1.2
- 3388 [ebMS-DPT] Deployment Profile Template For OASIS Specification ebXML Message Service 2.0
- 3389 [McGuinness, Harmelen] McGuinness, D. L., Harmelen, F., OWL Web Ontology Language Overview,
3390 W3C Recommendation 10 February 2004, <http://www.w3.org/TR/owl-features/>
- 3391 [OWL] Web Ontology Language (OWL), <http://www.w3.org/2004/OWL/>
- 3392 [RDF] Resource Description Framework, <http://www.w3.org/RDF/>
- 3393 [Smith, Welty, McGuinness] Smith, M. K., Welty, C., McGuinness, D. L.,
3394 OWL Web Ontology Language Guide, W3C Recommendation 10 February 2004,
3395 <http://www.w3.org/TR/owl-guide/>
- 3396 [SQL 92] SQL ISO/IEC 9075:1992 Information technology - Database languages - SQL.
- 3397 [SQL 99] ISO/IEC 9075:1999(E) Information technology - Database languages – SQL.
- 3398 [UML] Unified Modeling Language version 1.5
3399 <http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.pdf>
- 3400 [WSDL] WSDL Specification
3401 <http://www.w3.org/TR/wsdl>
- ### 3402 8.2 Informative References
- 3403 [Dogac, et. al. 2005] Dogac A., Kabak Y., Laleci G. C. Mattocks, F. Najmi, J. Pollock

3404 Enhancing ebXML Registries to Make them OWL Aware
3405 Distributed and Parallel Databases Journal, Springer-Verlag, Vol. 18, No. 1, July 2005, pp. 9-36.
3406
3407 [Dogac et. al. 2006] Dogac A., Laleci G., Kabak Y., Unal S., Beale T., Heard S., Elkin P., Najmi F.,
3408 Mattocks C., Webber D., Kernberg M.
3409 Exploiting ebXML Registry Semantic Constructs for Handling Archetype Metadata in Healthcare
3410 Informatics
3411 International Journal of Metadata, Semantics and Ontologies, Volume 1, No. 1, 2006.
3412
3413 [IMPL] ebXML Registry 3.0 Implementations
3414 freebXML Registry: A royalty free, open source ebXML Registry Implementation
3415 <http://ebxmlrr.sourceforge.net>
3416
3417 [LeeHendler]
3418 Berners-Lee, T., Hendler, J., Lassila, O., "The Semantic Web", Scientific American, May 2001.
3419
3420 [StaabStuder] Staab, S., Studer, R., Handbook on Ontologies, Springer, 2004.