



# ebXML Registry Profile for Web Ontology

## Language (OWL)

### Version 1.0 Draft 4

#### Draft OASIS Profile, March 08,2006

**Document identifier:**

regrep-owl-profile-1.0-draft 4

**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

**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](mailto:regrep-semantic@lists.oasis-open.org) list. Others should subscribe to and send comments to the  
26 [regrep-comment@lists.oasis-open.org](mailto:regrep-comment@lists.oasis-open.org) list. To subscribe, send an email message to [regrep-](mailto:regrep-comment-request@lists.oasis-open.org)  
27 [comment-request@lists.oasis-open.org](mailto:comment-request@lists.oasis-open.org) with the word "subscribe" as the body of the message.  
28

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  
32 (<http://www.oasis-open.org/committees/regrep/>).

---

# 1 Table of Contents

33

34	1 Table of Contents.....	3
35	1 Introduction.....	10
36	1.1 Terminology.....	10
37	1.2 Conventions.....	11
38	1.3 Recommendations.....	11
39	2 OWL Overview.....	12
40	2.1 OWL Lite Constructs.....	12
41	2.1.1 RDF Schema Features.....	12
42	2.1.2 (In)Equality.....	12
43	2.1.3 Property Characteristics .....	12
44	2.1.4 Property Restrictions.....	13
45	2.1.5 Restricted Cardinality.....	13
46	2.1.6 Class Intersection.....	13
47	2.1.7 Versioning.....	13
48	2.1.8 Annotation Properties .....	13
49	2.1.9 Datatypes .....	13
50	2.2 OWL DL Constructs.....	13
51	2.2.1 Class Axioms.....	13
52	2.2.2 Boolean Combinations of Class Expressions .....	14
53	2.2.3 Arbitrary Cardinality .....	14
54	2.2.4 Filler Information.....	14
55	3 ebXML Registry Overview.....	15
56	3.1 Overview of [ebRIM].....	15
57	3.1.1 RegistryObject.....	16
58	3.1.2 Object Identification.....	16
59	3.1.3 Object Naming and Description.....	17
60	3.1.4 Object Attributes.....	17
61	3.1.4.1 Slot Attributes.....	17
62	3.1.5 Object Classification.....	18
63	3.1.6 Object Association.....	18
64	3.1.7 Object References To Web Content.....	19
65	3.1.8 Object Packaging.....	19
66	3.1.9 ExtrinsicObject .....	20
67	3.1.10 Service Description.....	20
68	3.2 Overview of [ebRS].....	20
69	4 Representing OWL Lite Constructs in ebRIM .....	21
70	4.1 Representing RDF Schema Features in ebRIM.....	21
71	4.1.1 owl:Class → rim:ClassificationNode.....	21
72	4.1.2 rdf:Property → rim:Association Type Property.....	21
73	4.1.3 rdfs:subPropertyOf → rim:Association Type subPropertyOf.....	22
74	4.1.4 rdfs:subClassOf → rim:Association Type subClassOf.....	22
75	4.1.5 owl:Individual → rim:ExtrinsicObject.....	23
76	4.2 Representing OWL (In)Equality Constructs in ebXML RIM.....	23

77	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.....	25
82	4.3.1 owl:ObjectProperty → rim:Association Type objectProperty.....	25
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.....	27
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.....	32
96	4.7.1 owl:versionInfo, owl:priorVersion.....	32
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.....	33
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 SubClasses 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.....	41
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.....	44
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.....	48
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.....	49
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.....	50
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.....	51
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.....	52
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.....	53
192	6.20 SymmetricProperties Discovery Query.....	54
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  
245 the 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 the 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  
263 ebXML 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  
265 and precisely defined terms which can be used in formalizing knowledge and relationship among objects  
266 in a domain of interest. The other is to reason about the ontologies. When an ontology language like  
267 OWL is mapped to a class hierarchy like the one in ebXML, the first role can directly be achieved.  
268 Furthermore some implicit information can be obtained by predefined parameterized queries. However,  
269 when we want full reasoning power, we need reasoners. Yet, OWL reasoners can not directly run on the  
270 ebXML registry 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  
283 RFC 2119 [RFC211].

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.  
289 The 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  
296 readability, these key values are replaced by meaningful textual variables to represent such id values.  
297

298 For example, the following placeholder refers to the unique id defined for the canonical  
299 ClassificationNode that defines the Organization ObjectType defined in [ebRIM]:

300

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

## 302 1.3 Recommendations

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

305 AdhocQuery (id, lid, objectType, status, versionName, comment\_, queryLanguage, query);

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

---

## 2 OWL Overview

312

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

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

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

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

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

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

### 331 2.1 OWL Lite Constructs

#### 332 2.1.1 RDF Schema Features

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

#### 340 2.1.2 (In)Equality

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

#### 347 2.1.3 Property Characteristics

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

- 352 • SymmetricProperty
- 353 • FunctionalProperty
- 354 • InverseFunctionalProperty

## 355 2.1.4 Property Restrictions

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

## 360 2.1.5 Restricted Cardinality

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

## 364 2.1.6 Class Intersection

- 365 • intersectionOf

## 366 2.1.7 Versioning

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

## 373 2.1.8 Annotation Properties

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

## 380 2.1.9 Datatypes

- 381 • xsd datatypes

## 382 2.2 OWL DL Constructs

### 383 2.2.1 Class Axioms

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

387       •    rdfs:subClassOf (applied to class expressions)

## 388   2.2.2 Boolean Combinations of Class Expressions

- 389       •    unionOf
- 390       •    complementOf
- 391       •    intersectionOf

## 392   2.2.3 Arbitrary Cardinality

- 393       •    minCardinality
- 394       •    maxCardinality
- 395       •    cardinality

## 396   2.2.4 Filler Information

- 397       •    hasValue
- 398



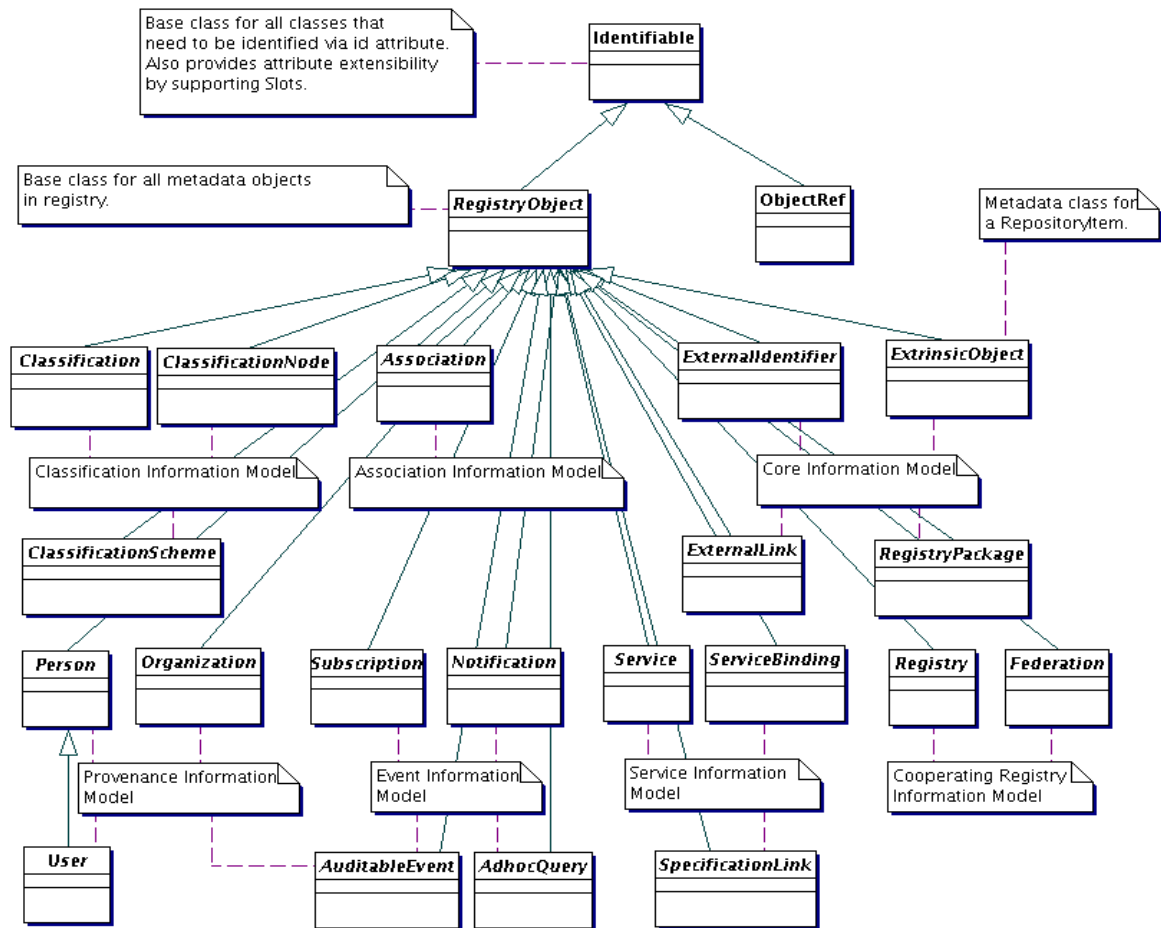


Figure 2: ebXML Registry Information Model, Inheritance View

421

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

### 423 3.1.1 RegistryObject

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

### 427 3.1.2 Object Identification

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

429

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

431

Listing 1: Example of id attribute

432 The id attribute value MAY potentially be human friendly.

433

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

435

Listing 2: Example of human friendly id attribute

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



439 human 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

```

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

```

Listing 6: Example of a Dynamic Attribute Using Slot

### 3.1.5 Object Classification

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

```

518
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.xml](http://www.oasis-open.org/committees/regrep/documents/3.0/canonical/SubmitObjectsRequest_AssociationTypeScheme.xml)

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  
585 as 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.].

## 4.1 Representing RDF Schema Features in ebRIM

647

### 4.1.1 owl:Class → rim:ClassificationNode

648

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

650

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

653

#### Example owl:Class

654

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

657

#### Example Corresponding ebRIM construct ClassificationNode

### 4.1.2 rdf:Property → rim:Association Type Property

658

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

663

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

667 </rdf:Property>

### Example rdf:Property

669

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

### 675 Example: ebRIM construct Association corresponding to rdf:Property

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

## 678 4.1.3 rdfs:subPropertyOf → rim:Association Type subPropertyOf

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

682

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

686

### Example rdfs:subPropertyOf

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

## 690 4.1.4 rdfs:subClassOf → rim:Association Type subclassOf

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

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

701

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

707

### Example rdfs:subClassOf

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

712

```
713 <rim:Association id='subClassOf1' associationType='urn:oasis:names:tc:ebxml-  
714   regrep:AssociationType:SubClassOf'  
715   sourceObject= 'AirReservationServices' targetObject='OWL-S Profile' >  
716 </rim:Association>
```

```

717 <rim:Association id='subClassOf2' associationType='urn:oasis:names:tc:ebxml-
718   regrep:AssociationType:SubClassOf'
719   sourceObject='AirReservationServices' targetObject='AirServices' >
720 </rim:Association>

```

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

#### 725 4.1.5 owl:Individual → rim:ExtrinsicObject

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

```

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

```

#### 737 Example owl:Class example

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

```

742
743 <TravelWebService rdf:ID="MyTravelWebService">
744   <paymentMethod> Cash </paymentMethod>
745 </TravelWebService>

```

#### 746 Example owl:Individual example

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

## 751 4.2 Representing OWL (In)Equality Constructs in ebXML RIM

### 752 4.2.1 owl:equivalentClass, owl:equivalentProperty → rim:Association Type 753 EquivalentTo

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

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

### 761 4.2.2 owl:sameAs → rim:Association Type sameAs

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

764 owl:sameAs construct is used to indicate that two instances in a knowledge base are the same. This

765 construct may be used to create a number of different names that refer to the same individual.

766

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

#### 770 **Example owl:sameAs**

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

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

### 775 4.2.3 owl:differentFrom → rim:Association Type differentFrom

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

780

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

#### 784 **Example owl:differentFrom**

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

### 789 4.2.4 owl:AllDifferent

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

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

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

#### 803 **Example owl:AllDifferentFrom**

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

808 **IMPORTANT NOTE:** When trying to submit the following "SubmitObjectsRequest", we get the following  
809 unexpected error from the freebXML which implies that in the new Registry implementation it is not  
810 possible to associate "slots" with RegistryPackages which seems there is a bug in the software.

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

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



## 814 4.3 Representing OWL Property Characteristics in ebRIM

### 815 4.3.1 owl:ObjectProperty → rim:Association Type objectProperty

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

819

```
820 <owl:ObjectProperty rdf:ID="hasAirport">  
821   <rdfs:domain rdf:resource="#City"/>  
822   <rdfs:range rdf:resource="#AirPort"/>  
823 </owl:ObjectProperty>
```

824

#### Example owl:ObjectProperty

825

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

830

#### Example Corresponding ebRIM construct Association

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

### 834 4.3.2 owl:DatatypeProperty → rim:Association Type DatatypeProperty

835 Similarly, to represent OWL DatatypeProperty in ebXML, a new Association Type called  
836 "DatatypeProperty" MUST be defined. Consider the following example which defines a datatype  
837 property "hasPrice" whose domain is the "AirReservationServices" and whose range is "XMLSchema  
838 nonNegativeInteger". How OWL XML Schema types are handled in ebXML RIM is described in Section  
839 4.9.

```
840 <owl:DatatypeProperty rdf:ID="hasPrice">  
841   <rdfs:domain rdf:resource="#AirReservationServices"/>  
842   <rdfs:range  
843   rdf:resource="http://www.w3.org/2000/10/XMLSchema/nonNegativeInteger"/>  
844 </owl:DatatypeProperty>
```

845

#### Example owl:DatatypeProperty

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

### 848 4.3.3 owl:TransitiveProperty → rim:Association Type transitiveProperty

849 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)  
850 [McGuinness, Harmelen]. Transitive property is a subproperty of ObjectProperty and MUST be defined  
851 as a new Association Type called "transitiveProperty" in ebRIM.

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

854

```
855 <owl:ObjectProperty rdf:ID="succeeds">  
856   <rdf:type rdf:resource="#owl:TransitiveProperty" />  
857   <rdfs:domain rdf:resource="#TravelWebService" />  
858   <rdfs:range rdf:resource="#TravelWebService" />  
859 </owl:ObjectProperty>
```

860

#### Example owl:TransitiveProperty

861 Assume the following two definitions which declare three Web service instances from TravelWebService

862 class where "MyHotelAvailabilityService" service succeeds "MyAirReservationService" and  
863 "MyInsuranceService" succeeds "MyHotelAvailabilityService". Since "succeeds" is a transitive property, it  
864 follows that "MyInsuranceService" succeeds "MyAirReservationService" although this fact is not explicitly  
865 stated.

866

```
867 <TravelWebService rdf:ID="MyHotelAvailabilityService">  
868   <succeeds rdf:resource="#MyAirReservationService" />  
869 </TravelWebService>  
870  
871 <TravelWebService rdf:ID="MyInsuranceService">  
872   <succeeds rdf:resource="#MyHotelAvailabilityService" />  
873 </TravelWebService>
```

#### 874 **Example owl:TransitiveProperty instances**

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

### 878 **4.3.4 owl:inverseOf → rim:Association Type inverseOf**

879 In OWL, one property may be stated to be the inverse of another property. If the property P1 is stated to  
880 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  
881 P1 property [McGuinness, Harmelen].

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

885

```
886 <owl:ObjectProperty rdf:ID="precedes">  
887   <owl:inverseOf rdf:resource="#succeeds" />  
888 </owl:ObjectProperty>
```

#### 889 **Example owl:inverseOf Property**

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

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

### 900 **4.3.5 owl:SymmetricProperty → rim:Association Type SymmetricProperty**

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

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

#### 913 **Example owl:SymmetricProperty**

914 Given that `HotelReservationWebService` complements `AirReservationWebService`, it is possible to  
915 deduce that `AirReservationWebService` complements `HotelReservationWebService`.  
916 `owl:SymmetricProperty` MUST be defined as a new type of `Association` in ebRIM called  
917 "SymmetricProperty". Furthermore the adhoc query presented in Section 6.20 MUST be available in the  
918 Registry to retrieve symmetric Associations of a `ClassificationNode`.

#### 919 4.3.6 `owl:FunctionalProperty` → `rim:Association Type FunctionalProperty`

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

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

#### 930 **Example owl:FunctionalProperty**

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

#### 934 4.3.7 `owl:InverseFunctionalProperty` → `rim:Association Type` 935 `InverseFunctionalProperty`

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

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

```
944 <owl:ObjectProperty rdf:ID="finalDestination">  
945   <rdf:type rdf:resource="&owl;InverseFunctionalProperty" />  
946   <rdfs:domain rdf:resource="#Airport"/>  
947   <rdfs:range rdf:resource="#Flight"/>  
948 </owl:ObjectProperty>
```

#### 949 **Example owl:InverseFunctionalProperty**

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

### 953 4.4 OWL Property Restrictions in ebXML RIM

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

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

963

964  
965  
966  
967

```

<owl:ObjectProperty rdf:ID="paymentMethod">
  <rdfs:domain rdf:resource="#TravelWebService"/>
  <rdfs:range rdf:resource="#PossiblePaymentMethods"/>
</owl:ObjectProperty >

```

968

### Example owl:ObjectProperty “paymentMethod”

969

970  
971  
972  
973  
974  
975  
976  
977

```

<owl:Class rdf:ID="AirReservationServices">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#paymentMethod"/>
      <owl:allValuesFrom rdf:resource= "#CreditCard"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

```

978

### Example owl:Restriction on ObjectProperty “paymentMethod”

979

980  
981  
982  
983  
984  
985

A new Association Type of “restriction” SHOULD be defined to represent OWL restriction. A slot of this Association Type SHOULD indicate the whether the restriction is “allValuesFrom” or “someValuesFrom”. When such restriction is submitted to the system, the registry MUST create a new Association instance, say, “paymentMethod\_1” of AssociationType “ObjectProperty” is created whose sourceObject is “AirReservationServices” and the targetObject is “CreditCard”. “paymentMethod\_1” Association instance is related with the “paymentMethod” Association instance by using an instance of the Association Type “restriction” as shown in the following example:

986

987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003

```

<rim:Association id = "paymentMethod_1"
  associationType =
"urn:oasis:names:tc:ebxml-regrep:AssociationType:ObjectProperty"
  sourceObject = "AirReservationServices"
  targetObject = "CreditCard" />

  <rim:Association id = "paymentMethodRestriction"
    associationType =
"urn:oasis:names:tc:ebxml-regrep:AssociationType:restriction"
    sourceObject = "paymentMethod"
    targetObject = "paymentMethod_1">
    <rim:Slot name="restrictionType">
      <rim:ValueList>
        <rim:Value>allValuesFrom</rim:Value>
      </rim:ValueList>
    </rim:Slot>
  </rim:Association>

```

1004

### Example Handling owl:Restriction in ebXML Registry

1005

1006  
1007

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

1008

1009  
1010  
1011  
1012

```

<owl:ObjectProperty rdf:ID="paymentMethodCC">
  <rdfs:domain rdf:resource="#AirReservationServices"/>
  <rdfs:range rdf:resource="#CreditCard"/>
</owl:ObjectProperty >

```

1013

### Example owl:ObjectProperty “paymentMethodCC”

1014

## 4.5 Representing OWL Restricted Cardinality in ebXML RIM

1015

### 4.5.1 owl:minCardinality (only 0 or 1)

1016

1017

In OWL, cardinality is stated on a property with respect to a particular class. If a minCardinality of 1 is stated on a property with respect to a class, then then any instance of the class will have at least one

1018 value for the restricted property. This restriction is another way of saying that the property is required to  
1019 have a value for all instances of the class. In OWL Lite, the only minimum cardinalities allowed are 0 or 1.  
1020 A minimum cardinality of zero on a property just states (in the absence of any more specific information)  
1021 that the property is optional with respect to a class [McGuinness, Harmelen].

1022 Consider for example the following OWL code which states that each instance of a “WebService” class  
1023 must have at least one price:

```
1024 <owl:Class rdf:ID="WebService">  
1025   <rdfs:subClassOf>  
1026     <owl:Restriction>  
1027       <owl:onProperty rdf:resource="#hasPrice"/>  
1028       <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">  
1029 1 </owl:minCardinality>  
1030     </owl:Restriction>  
1031   </rdfs:subClassOf>  
1032 </owl:Class>
```

### 1033 Example owl:minCardinality

1034 In ebXML RIM, cardinalities of Association Types MUST be defined by associating a minCardinality slot  
1035 with the Association Types as shown in the following example:

1036

```
1037 <rim:Association id = "hasPriceMinCardinalityRestriction"  
1038 associationType = "urn:oasis:names:tc:ebxml-  
1039 regrep:AssociationType:ObjectProperty" sourceObject = "WebService"  
1040 targetObject = "Price">  
1041   <rim:Name>  
1042     <rim:LocalizedString value = 'hasPrice' />  
1043   </rim:Name>  
1044   <rim:Slot name="minCardinality">  
1045     <rim:ValueList>  
1046       <rim:Value>1</rim:Value>  
1047     </rim:ValueList>  
1048   </rim:Slot>  
1049 </rim:Association>
```

### 1050 Example Representing owl:minCardinality in ebRIM

## 1051 4.5.2 owl:maxCardinality (only 0 or 1)

1052 In OWL, cardinality is stated on a property with respect to a particular class. If a maxCardinality of 1 is  
1053 stated on a property with respect to a class, then any instance of that class will be related to at most one  
1054 individual by that property. A maxCardinality 1 restriction is sometimes called a functional or unique  
1055 property. It may be useful to state that certain classes have no values for a particular property. This  
1056 situation is represented by a maximum cardinality of zero on the property [McGuinness, Harmelen].

1057 Consider for example the following OWL code which states that each instance of a “WebService” class  
1058 can have at most one price:

```
1059 <owl:Class rdf:ID="WebService">  
1060   <rdfs:subClassOf>  
1061     <owl:Restriction>  
1062       <owl:onProperty rdf:resource="#hasPrice"/>  
1063       <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">  
1064 1 </owl:maxCardinality>  
1065     </owl:Restriction>  
1066   </rdfs:subClassOf>  
1067 </owl:Class>
```

### 1068 Example owl:maxCardinality

1069 In ebXML RIM, cardinalities of Association Types MUST be defined by associating a maxCardinality slot  
1070 with the Association Types as shown in the following example:

1071

```
1072 <rim:Association id = "hasPriceMaxCardinalityRestriction"
```

```

1073 associationType = "urn:oasis:names:tc:ebxml-
1074 regrep:AssociationType:ObjectProperty" sourceObject = "WebService"
1075 targetObject = "Price">
1076   <rim:Name>
1077     <rim:LocalizedString value = 'hasPrice' />
1078   </rim:Name>
1079   <rim:Slot name="maxCardinality">
1080     <rim:ValueList>
1081       <rim:Value>1</rim:Value>
1082     </rim:ValueList>
1083   </rim:Slot>
1084 </rim:Association>

```

1085 **Example Representing owl:maxCardinality in ebRIM**

### 1086 4.5.3 owl:cardinality (only 0 or 1)

1087 In OWL Lite, cardinality is provided as a convenience when it is useful to state that a property on a class  
1088 has both minCardinality 0 and maxCardinality 0 or both minCardinality 1 and maxCardinality 1  
1089 [McGuinness, Harmelen].

1090 Consider for example the following OWL code which states that each instance of a "WebService" class  
1091 must have exactly one price:

```

1092 <owl:Class rdf:ID="WebService">
1093   <rdfs:subClassOf>
1094     <owl:Restriction>
1095       <owl:onProperty rdf:resource="#hasPrice"/>
1096       <owl:Cardinality rdf:datatype="&xsd;nonNegativeInteger"> 1
1097     </owl:Cardinality>
1098   </owl:Restriction>
1099   </rdfs:subClassOf>
1100 </owl:Class>

```

1101 **Example owl:Cardinality**

1102 In ebXML RIM, cardinalities of Association Types MUST be defined by associating a Cardinality slot with  
1103 the Association Types as shown in the following example:

```

1104
1105 <rim:Association id = "hasPriceCardinalityRestriction"
1106 associationType = "urn:oasis:names:tc:ebxml-
1107 regrep:AssociationType:ObjectProperty" sourceObject = "WebService"
1108 targetObject = "Price">
1109   <rim:Name>
1110     <rim:LocalizedString value = 'hasPrice' />
1111   </rim:Name>
1112   <rim:Slot name="cardinality">
1113     <rim:ValueList>
1114       <rim:Value>1</rim:Value>
1115     </rim:ValueList>
1116   </rim:Slot>
1117 </rim:Association>

```

1118 **Example Representing owl:Cardinality in ebRIM**

### 1119 4.6 Representing OWL Class Intersection in ebXML RIM

1120 OWL provides the means to manipulate class extensions using basic set operators. In OWL lite, only  
1121 "owl:intersectionOf" is available which defines a class that consists of exactly all objects that belong to  
1122 both of the classes. In the following example, "AirReservationServices" is defined as the intersection of  
1123 "AirServices" and "ReservationServices":

```

1124
1125 <owl:Class rdf:ID="AirReservationServices">
1126   <owl:intersectionOf rdf:parseType="Collection">
1127     <owl:Class rdf:about="#AirServices" />
1128     <owl:Class rdf:about="#ReservationServices" />

```

```
1129     </owl:intersectionOf>
1130 </owl:Class>
```

### 1131 **Example owl:intersectionOf**

1132 In ebXML RIM "owl:intersectionOf" set operator MUST be represented as follows:

- 1133 • A new Association Type called "intersectionOf" MUST be created.
- 1134 • A new ClassificationNode to denote the intersection of the classes MUST be created. For the  
1135 example, this could be "AirReservationServices" ClassificationNode.
- 1136 • Each of the intersected classes MUST be represented as members of a new RegistryPackage.  
1137 For the example, the RegistryPackage should contain "AirServices" and the  
1138 "RegistrationServices".
- 1139 • The new ClassificationNode denoting the intersection MUST be assigned as the sourceObject of  
1140 the "intersectionOf" association. For the example, "AirReservationServices" must be the the  
1141 sourceObject of the "intersectionOf" association.
- 1142 • The target class of the "intersectionOf" association MUST be set to the newly created  
1143 RegistryPackage. For the example given above, the RegistryPackage containing "AirServices"  
1144 and the "RegistrationServices" should be the target class of the "intersectionOf" association.

```
1145
1146 <rim:ClassificationNode id = "AirReservationServices" code =
1147 "AirReservationServices">
1148   <rim:Name>
1149     <rim:LocalizedString value = "AirReservationServices" />
1150   </rim:Name>
1151 </rim:ClassificationNode>
1152
1153
1154 <rim:RegistryPackage id = "IntersectionOfRegistryPackage" >
1155   <rim:Name>
1156     <rim:LocalizedString value =
1157 "IntersectionOfRegistryPackage"/>
1158   </rim:Name>
1159 </rim:RegistryPackage>
1160
1161 <rim:Association id = "HasMemberRegistryPackageAssoc1"
1162 associationType = "urn:oasis:names:tc:ebxml-
1163 regrep:AssociationType:HasMember" sourceObject =
1164 "IntersectionOfRegistryPackage"
1165 targetObject = "AirServices" />
1166
1167 <rim:Association id = "HasMemberRegistryPackageAssoc2"
1168 associationType = "urn:oasis:names:tc:ebxml-
1169 regrep:AssociationType:HasMember" sourceObject =
1170 "IntersectionOfRegistryPackage"
1171 targetObject = "ReservationServices" />
1172
1173 <rim:Association id = "IntersectionOfRegistryPackageAssoc"
1174 associationType = "urn:oasis:names:tc:ebxml-
1175 regrep:AssociationType:IntersectionOf" sourceObject =
1176 "AirReservationServices"
1177 targetObject = " IntersectionOfRegistryPackage " />
1178
```

### 1179 **Example Defining Intersection of ClassificationNodes in ebRIM**

1180 When such a representation is used to create a complex class (a new ClassificationNode) in RIM, it  
1181 becomes possible to infer that the objects (instances) classified by both of the classes  
1182 (ClassificationNodes) constituting the intersection are also the instances of this complex class. The adhoc  
1183 query presented in Section 6.23 MUST be available in the ebXML Registry to retrieve the direct instances  
1184 of the complex class and also the instances of the intersection of the classes.

1185

## 1186 4.7 Representing OWL Versioning in ebXML RIM

### 1187 4.7.1 owl:versionInfo, owl:priorVersion

1188 An owl:versionInfo statement generally has as its object a string giving information about this version, for  
1189 example RCS/ CVS keywords. This statement does not contribute to the logical meaning of the ontology  
1190 other than that given by the RDF(S) model theory [McGuinness, Harmelen].

1191 An owl:priorVersion statement contains a reference to another ontology. This identifies the specified  
1192 ontology as a prior version of the containing ontology [McGuinness, Harmelen].

1193 In ebXML, since a RegistryObject MAY have several versions, a logical id (called lid) is also defined  
1194 which is unique for different logical objects. However the lid attribute value MUST be the same for all  
1195 versions of the same logical object. Therefore, almost all the underlying ebXML relational tables keep  
1196 version information through "versionName" and "comment\_" attributes.

1197 "owl:version" information MUST be stored in the "versionName" and "comment\_" attributes of the table  
1198 ClassScheme in the Registry.

## 1199 4.8 Representing OWL Annotation Properties in ebXML RIM

### 1200 4.8.1 rdfs:label

1201 rdfs:label is an instance of rdf:Property that may be used to provide a human-readable version of a  
1202 resource's name [Brickley, Guha].

1203 In ebXML RIM, human readable names of resources are provided through rim:Name. rdfs:label MUST be  
1204 expressed through rim:Name.

1205

```
1206 <owl:Class rdf:ID="AirReservationServices">  
1207   <rdfs:label>Air Reservation Services</rdfs:label>  
1208 </owl:Class>
```

#### 1209 Example rdfs:label

1210

```
1211 <rim:ClassificationNode id = 'AirReservationServices' code =  
1212 'AirReservationServices'>  
1213   <rim:Name>  
1214     <rim:LocalizedString value = 'Air Reservation Services' />  
1215   </rim:Name>  
1216 </rim:ClassificationNode>
```

#### 1217 Example rim:Name

### 1218 4.8.2 rdfs:comment

1219 rdfs:comment is an instance of rdf:Property that may be used to provide a human-readable description of  
1220 a resource [Brickley, Guha].

1221 In ebXML RIM, this construct MUST be expressed through rim:Description.

1222

```
1223 <owl:Class rdf:ID="AirReservationServices">  
1224   <rdfs:comment>Open Travel Alliance Air Reservation Services  
1225   </rdfs:comment>  
1226 </owl:Class>
```

#### 1227 Example rdfs:comment

1228

```
1229 <rim:ClassificationNode id = 'AirReservationServices' code =  
1230 'AirReservationServices'>  
1231   <rim:Description>
```



```
1232         <rim:LocalizedString value = 'Open Travel Alliance Air
1233 Reservation Services' />
1234     </rim:Description>
1235 </rim:ClassificationNode>
```

1236 **Example: rim:Description**

### 1237 4.8.3 rdfs:seeAlso

1238 rdfs:seeAlso is an instance of rdf:Property that is used to indicate a resource that might provide additional  
1239 information about the subject resource [Brickley, Guha].

1240 This construct MUST be expressed in ebXML RIM by defining an ExternalLink, called,  
1241 "seeAlsoExternalLink".

1242

```
1243 <owl:Class rdf:ID="AirReservationServices">
1244     <rdfs:seeAlso rdf:resource="http://www.opentravel.org" />
1245 </owl:Class>
```

1246 **Example rdfs:seeAlso**

```
1247 <rim:ClassificationNode id = 'AirReservationServices' code =
1248 'AirReservationServices'>
1249 </rim:ClassificationNode>
1250
1251 <rim:ExternalLink id = "seeAlsoExternalLink"
1252     externalURI= "http://www.opentravel.org" >
1253 </rim:ExternalLink>
1254
1255 <rim:Association id = 'seeAlsoAssociation'
1256     associationType = 'urn:oasis:names:tc:ebxml-
1257 regrep:AssociationType:ExternallyLinks'
1258     sourceObject = 'AirReservationServices'
1259     targetObject = 'seeAlsoExternalLink' />
```

1260 **Example rim:seeAlsoExternalLink**

### 1261 4.9 OWL Datatypes in ebXML RIM

1262 OWL allows the use of XML Schema datatypes to describe part of the datatype domain by simply  
1263 including their URIs within an OWL ontology [McGuinness, Harmelen]. In ebXML, XML Schema  
1264 datatypes SHOULD be used by providing an external link from the registry.

1265 The following example demonstrates how XML Schema datatype "integer" can be referenced through an  
1266 ExternalLink called 'integer' and how to define a DatatypeProperty, namely, "hasPrice", whose target  
1267 object is the defined to be ExternalLink 'integer':

1268

```
1269 <rim:ExternalLink id = "integer"
1270     externalURI="http://www.w3.org/2001/XMLSchema#integer" >
1271     <rim:Name> <rim:LocalizedString value = "XML Schema integer" />
1272     </rim:Name>
1273 </rim:ExternalLink>
1274 <rim:Association id = 'hasPrice' associationType = 'urn:oasis:names:tc:ebxml-
1275 regrep:AssociationType:DatatypeProperty'
1276     sourceObject = 'AirReservationServices'
1277     targetObject = 'integer' >
1278     <rim:Name> <rim:LocalizedString value ="hasPrice" /></rim:Name>
1279 </rim:Association>
```

1281 **Example Corresponding ebRIM construct Association**

1282

---

## 5 Cataloging Service Profile

1283

1284 The ebXML Registry provides the ability for a content cataloging service to be configured for any type of  
1285 content. The cataloging service serves the following purposes:

- 1286 • Automates the mapping from the source information model (in this case OWL) to ebRIM. This  
1287 hides the complexity of the mapping from the OWL publisher and eliminates the need for any  
1288 special UI tools to be provided by the registry implementor for publishing OWL documents.
- 1289 • Selectively converts content into ebRIM compatible metadata when the content is cataloged after  
1290 being published. The generated metadata enables the selected content to be used as  
1291 parameter(s) in content specific parameterized queries.

1292 This section describes the cataloging service for cataloging OWL content.

1293 An OWL document, when published to an ebXML Registry implementing the OWL Profile, **MUST** be  
1294 cataloged as specified in this section using a OWL Content Cataloging Service as defined by [ebRS].

### 5.1 Invocation Control File

1295

1296 The OWL cataloging service **MAY** optionally support an invocation control file that declaratively specifies  
1297 the transforms necessary to catalog published OWL documents.

### 5.2 Input Metadata

1298

1299 The OWL cataloging service **MUST** be pre-configured to be automatically invoked when the following  
1300 types of metadata are published, as defined by the [ebRS] specifications.

1301 These are the only types of metadata that **MAY** describe a OWL document being published:

- 1302 • An ExtrinsicObject whose ObjectType references the canonical OWL ClassificationNode  
1303 specified in Section 7. The ExtrinsicObject **MUST** have an OWL document as its RepositoryItem.
- 1304 • An ExternalLink whose ObjectType references the canonical OWL ClassificationNode specified  
1305 in Section 7. In case of ExternalLink the OWL document **MUST** be resolvable via a URL  
1306 described by the value of the externalURI attribute of the ExternalLink. Recall that, in the  
1307 ExternalLink case the OWL document is not be stored in the repository.

1308

```
1309 <rim:ExtrinsicObject id="urn:acmeinc:ebxml:registry:3.0:owl">  
1310 ...  
1311 </rim:ExtrinsicObject>
```

#### Example of ExtrinsicObject Input Metadata

1312

1313

```
1314 <rim:ExternalLink  
1315   id="urn:acmeinc:ebxml:registry:3.0:owl"  
1316   externalURI="http://www.acme.com/owl/ebXMLRegistryService.owl"  
1317   >  
1318 ...  
1319 </rim:ExternalLink>
```

#### Example of ExternalLink Input Metadata

1320

### 5.3 Input Content

1321

1322 The OWL cataloging service expects an OWL document as its input content. The input content **MUST** be  
1323 processed by the OWL cataloging service regardless of whether it is a RepositoryItem for an  
1324 ExtrinsicObject or whether it is content external to repository that is referenced by an ExternalLink.

1325

## 1326 5.4 Output Metadata

1327 This section describes the metadata produced by the OWL cataloging service produces as output.

### 1328 5.4.1 owl:Class → rim:ClassificationNode

1329 The OWL Cataloging service MUST automatically produce a rim:ClassificationNode instance for each  
1330 owl:class element within the input OWL or its imports, as specified in the owl:Class →  
1331 rim:ClassificationNode mapping earlier in this document.

### 1332 5.4.2 rdf:Property → rim:Association Type Property

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

### 1336 5.4.3 rdfs:subPropertyOf → rim:Association Type subPropertyOf

1337 The OWL Cataloging service MUST automatically produce an rim:Association instance with  
1338 associationType subPropertyOf for each rdfs:subPropertyOf element within the input OWL or its imports,  
1339 as specified in the rdfs:subPropertyOf → rim:Association Type subPropertyOf mapping earlier in this  
1340 document.

### 1341 5.4.4 rdfs:subClassOf → rim:Association Type subClassOf

1342 The OWL Cataloging service MUST automatically produce an rim:Association instance with  
1343 associationType subClassOf for each rdfs:subClassOf element within the input OWL or its imports, as  
1344 specified in the rdfs:subClassOf → rim:Association Type subClassOf mapping earlier in this document.

### 1345 5.4.5 owl:Individual → rim:ExtrinsicObject

1346 The OWL Cataloging service MUST automatically produce rim:ExtrinsicObject instances for each  
1347 owl:Individual element within the input OWL or its imports, as specified in the owl:Individual →  
1348 rim:ExtrinsicObject mapping earlier in this document.

### 1349 5.4.6 owl:equivalentClass, owl:equivalentProperty → rim:Association Type 1350 EquivalentTo

1351 The OWL Cataloging service MUST automatically produce rim:Association instances with  
1352 associationType EquivalentTo for each owl:equivalentClass or owl:equivalentProperty element within the  
1353 input OWL or its imports, as specified in the owl:equivalentClass, owl:equivalentProperty →  
1354 rim:Association Type EquivalentTo mapping earlier in this document.

### 1355 5.4.7 owl:sameAs → rim:Association Type sameAs

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

### 1359 5.4.8 owl:differentFrom → rim:Association Type differentFrom

1360 The OWL Cataloging service MUST automatically produce rim:Association instances with  
1361 associationType differentFrom for each owl:differentFrom element within the input OWL or its imports, as  
1362 specified in the owl:differentFrom → rim:Association Type differentFrom mapping earlier in this  
1363 document.

#### 1364 [5.4.9 owl:AllDifferent → rim:RegistryPackage](#)

1365 The OWL Cataloging service MUST automatically produce rim:RegistryPackage instances for each  
1366 owl:AllDifferent element within the input OWL or its imports, as specified in the owl:AllDifferent →  
1367 rim:RegistryPackage mapping earlier in this document.

#### 1368 [5.4.10 owl:ObjectProperty → rim:Association Type objectProperty](#)

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

#### 1373 [5.4.11 owl:DatatypeProperty → rim:Association Type DatatypeProperty](#)

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

#### 1378 [5.4.12 owl:TransitiveProperty → rim:Association Type transitiveProperty](#)

1379 The OWL Cataloging service MUST automatically produce rim:Association instances with  
1380 associationType transitiveProperty for each owl:TransitiveProperty element within the input OWL or its  
1381 imports, as specified in the owl:TransitiveProperty → rim:Association Type transitiveProperty mapping  
1382 earlier in this document.

#### 1383 [5.4.13 owl:inverseOf → rim:Association Type inverseOf](#)

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

#### 1387 [5.4.14 owl:SymmetricProperty → rim:Association Type SymetricProperty](#)

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

#### 1392 [5.4.15 owl:FunctionalProperty → rim:Association Type FunctionalProperty](#)

1393 The OWL Cataloging service MUST automatically produce rim:Association instances with  
1394 associationType FunctionalProperty for each owl:FunctionalProperty element within the input OWL or its  
1395 imports, as specified in the owl:FunctionalProperty → rim:Association Type FunctionalProperty mapping  
1396 earlier in this document.

#### 1397 [5.4.16 owl:InverseFunctionalProperty → rim:Association Type 1398 InverseFunctionalProperty](#)

1399 The OWL Cataloging service MUST automatically produce rim:Association instances with  
1400 associationType InverseFunctionalProperty for each owl:InverseFunctionalProperty element within the  
1401 input OWL or its imports, as specified in the owl:InverseFunctionalProperty → rim:Association Type  
1402 InverseFunctionalProperty mapping earlier in this document.

#### 1403 [5.4.17 owl:minCardinality \(only 0 or 1\)](#)

1404 The OWL Cataloging service MUST automatically add a slot with name minCardinality to the relevant

1405 rim:Association instances for each owl:minCardinality element within the input OWL or its imports, as  
1406 specified in section 4.5.1 where how to represent owl:minCardinality is described.

#### 1407 **5.4.18 owl:maxCardinality (only 0 or 1)**

1408 The OWL Cataloging service MUST automatically add a slot with name maxCardinality to the relevant  
1409 rim:Association instances for each owl:maxCardinality element within the input OWL or its imports, as  
1410 specified in section 4.5.2 where how to represent owl:maxCardinality is described.

#### 1411 **5.4.19 owl:cardinality**

1412 The OWL Cataloging service MUST automatically add a slot with name cardinality to the relevant  
1413 rim:Association instances for each owl:cardinality element within the input OWL or its imports, as  
1414 specified in section 4.5.3 where how to represent owl:cardinality is described.

#### 1415 **5.4.20 owl:intersectionOf**

1416 The OWL Cataloging service MUST automatically produce a rim:RegistryPackage and a rim:Association  
1417 instances with type IntersectionOf for each owl:intersectionOf element within the input OWL or its  
1418 imports, as specified in section 4.6 where how to represent owl:intersectionOf is described.

#### 1419 **5.4.21 rdfs:label**

1420 The OWL Cataloging service MUST automatically produce a rim:Name instance for each rdfs:label  
1421 element within the input OWL or its imports, as specified in section 4.8.1 where how to represent  
1422 rdfs:label is described.

#### 1423 **5.4.22 rdfs:comment**

1424 The OWL Cataloging service MUST automatically produce a rim:Description instance for each  
1425 rdfs:comment element within the input OWL or its imports, as specified in section 4.8.2 where how to  
1426 represent rdfs:comment is described.

#### 1427 **5.4.23 rdfs:seeAlso**

1428 The OWL Cataloging service MUST automatically produce a rim:ExternalLink and a rim:Association with  
1429 type ExternallyLinks instances for each rdfs:seeAlso element within the input OWL or its imports, as  
1430 specified in section 4.8.3 where how to represent rdfs:seeAlso is described.

---

## 6 Discovery Profile

1431

1432 The ebXML Registry provides the ability for a user defined parameterized queries to be configured for  
1433 each type of content. The queries may be as complex or simple as the discovery use case requires. The  
1434 complexity of the parameterized queries may hidden from the registry client by storing them within the  
1435 ebXML Registry as instances of the AdhocQuery class, and being invoked by simply providing their  
1436 parameters. Query parameters are often pattern strings that may contain wildcard characters '%'  
1437 (matches any number of characters) and '\_' (matches exactly one character) as described by [ebRS].

1438 An ebXML Registry SHOULD provide a graphical user interface that displays any configured  
1439 parameterized query as a form which contains an appropriate field for entering each query parameter.

1440 This chapter defines the queries that MUST be support by an ebXML Registry implementing the OWL  
1441 Profile for processing the semantics provided in the OWL content. An implementation MAY also support  
1442 additional discovery queries for OWL content, some of which have already identified in this section.

1443 The queries defined in this chapter are parameterized queries stored in the Registry as instances of the  
1444 AdhocQuery type, in the same manner as any other RegistryObject.

1445 In the subsequent section each query is described simply in terms of its supported parameters that serve  
1446 as its search criteria. The actual AdhocQuery instances are much more complex in comparison but they  
1447 are not exposed to the client making the query. Details on these queries are specified canonically in  
1448 section 7.3 .

1449 Some of the queries that are necessary to process the semantics involved in OWL documents requires  
1450 SQL recursion mechanism. Since SQL 92, does not support recursion mechanism, those queries are  
1451 stated to be implemented optionally. Additionally for these types of discovery queries, the "stored  
1452 procedures" are presented in Section 7.3.

### 6.1 All SuperProperties Discovery Query

1453  
1454 As presented in Section 4.1.3, a new ebXML RIM Association Type called "SubPropertyOf" MUST be  
1455 defined to represent rdfs:subPropertyOf in ebRIM. Such a semantic enhancement brings the following  
1456 processing need: given a property, it should be possible to retrieve all of its super properties. This  
1457 requires a recursion mechanism in SQL queries.

1458 The freebXML implementation allows various relational database products such as Oracle, PostgreSQL  
1459 and MS SQL Server 2005 to be used as the database. These products have different support for  
1460 recursion mechanism in SQL Queries.

1461 The AllSuperProperties discovery query MAY be implemented by an ebXML Registry implementing this  
1462 profile. It allows the discovery of all super properties of a given property instance (Association instance  
1463 in ebXML terminology) recursively in a property hierarchy (hierarchy of Association Types) in freebXML  
1464 Registry implementations using MS SQL Server 2005 as the database.

#### 6.1.1 Parameter \$propertyName

1465  
1466 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
1467 value of Associations that have associationType of Property.

#### 6.1.2 Example of All SuperProperties Discovery Query

1468  
1469 The following example illustrates how to find all the super properties of a given property having a name  
1470 containing "creditCardPayment" if the query is implemented as an AdHoc Query.

1471

```
1472 <<rs:RequestSlotList>  
1473   <rim:Slot  
1474     name="urn:oasis:names:tc:ebxml-  
1475     regrep:3.0:rs:AdhocQueryRequest:queryId">  
1476     <rim:ValueList>
```

```

1477         <rim:Value>urn:oasis:names:tc:ebxml-
1478 regrep:query:FindAllSuperProperties</rim:Value>
1479         </rim:ValueList>
1480     </rim:Slot>
1481     <rim:Slot name="urn:oasis:names:tc:ebxml-
1482 regrep:rs:AdhocQueryRequest:queryId">
1483         <rim:ValueList>
1484             <rim:Value>urn:oasis:names:tc:ebxml-
1485 regrep:query:FindAllSuperProperties</rim:Value>
1486         </rim:ValueList>
1487     </rim:Slot>
1488     <rim:Slot name="$propertyName">
1489         <rim:ValueList>
1490             <rim:Value>%creditCardPayment%</rim:Value>
1491         </rim:ValueList>
1492     </rim:Slot>
1493 </rs:RequestSlotList>
1494
1495 <query:ResponseOption returnComposedObjects="true"
1496     returnType="LeafClassWithRepositoryItem"/>
1497
1498 <rim:AdhocQuery id="temporaryId">
1499     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1500 regrep:QueryLanguage:SQL-92">
1501     </rim:QueryExpression>
1502 </rim:AdhocQuery>

```

1503 Example of All SuperProperties Discovery Query

## 1504 6.2 Immediate SuperClass Discovery Query

1505 The Immediate SuperClass discovery query MUST be implemented by an ebXML Registry implementing  
1506 this profile. It allows the discovery of all of the immediate super classes of a given class.

### 1507 6.2.1 Parameter \$className

1508 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
1509 value of ClassificationNodes.

### 1510 6.2.2 Example of Immediate SuperClass Discovery Query

1511 The following example illustrates how to find all the immediate super classes of a given class that have a  
1512 name containing the string "AirReservationServices".

```

1513 <rs:RequestSlotList>
1514     <rim:Slot
1515         name="urn:oasis:names:tc:ebxml-
1516 regrep:3.0:rs:AdhocQueryRequest:queryId">
1517         <rim:ValueList>
1518             <rim:Value>urn:oasis:names:tc:ebxml-
1519 regrep:query:FindImmediateSuperClasses</rim:Value>
1520         </rim:ValueList>
1521     </rim:Slot>
1522     <rim:Slot name="urn:oasis:names:tc:ebxml-
1523 regrep:rs:AdhocQueryRequest:queryId">
1524         <rim:ValueList>
1525             <rim:Value>urn:oasis:names:tc:ebxml-
1526 regrep:query:FindImmediateSuperClasses</rim:Value>
1527         </rim:ValueList>
1528     </rim:Slot>
1529     <rim:Slot name="$className">
1530         <rim:ValueList>
1531             <rim:Value>%AirReservationServices%</rim:Value>
1532         </rim:ValueList>
1533     </rim:Slot>
1534 </rs:RequestSlotList>

```

```

1535
1536 <query:ResponseOption returnComposedObjects="true"
1537     returnType="LeafClassWithRepositoryItem"/>
1538
1539 <rim:AdhocQuery id="temporaryId">
1540     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1541     regrep:QueryLanguage:SQL-92">
1542         </rim:QueryExpression>
1543 </rim:AdhocQuery>

```

1544 Example of Immediate SuperClass Discovery Query

## 1545 6.3 Immediate SubClass Discovery Query

1546 The Immediate SubClass discovery query MUST be implemented by an ebXML Registry implementing  
1547 this profile. It allows the discovery of all of the immediate subclasses of a given class.

### 1548 6.3.1 Parameter \$className

1549 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
1550 value of ClassificationNode.

### 1551 6.3.2 Example of Immediate SubClasses Discovery Query

1552 The following example illustrates how to find all the immediate subclasses of a given class that have a  
1553 name containing the string "AirServices".

```

1554 <rs:RequestSlotList>
1555     <rim:Slot
1556         name="urn:oasis:names:tc:ebxml-
1557     regrep:3.0:rs:AdhocQueryRequest:queryId">
1558         <rim:ValueList>
1559             <rim:Value>urn:oasis:names:tc:ebxml-
1560     regrep:query:FindImmediateSubClasses</rim:Value>
1561         </rim:ValueList>
1562     </rim:Slot>
1563     <rim:Slot name="urn:oasis:names:tc:ebxml-
1564     regrep:rs:AdhocQueryRequest:queryId">
1565         <rim:ValueList>
1566             <rim:Value>urn:oasis:names:tc:ebxml-
1567     regrep:query:FindImmediateSubClasses</rim:Value>
1568         </rim:ValueList>
1569     </rim:Slot>
1570     <rim:Slot name="$className">
1571         <rim:ValueList>
1572             <rim:Value>%AirServices%</rim:Value>
1573         </rim:ValueList>
1574     </rim:Slot>
1575 </rs:RequestSlotList>
1576
1577 <query:ResponseOption returnComposedObjects="true"
1578     returnType="LeafClassWithRepositoryItem"/>
1579
1580 <rim:AdhocQuery id="temporaryId">
1581     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1582     regrep:QueryLanguage:SQL-92">
1583         </rim:QueryExpression>
1584 </rim:AdhocQuery>

```

1585 Example of Immediate SubClass Discovery Query

## 1586 6.4 All SuperClasses Discovery Query

1587 It should be noted that, given a class, finding its immediate subclasses, super classes is necessary but  
1588 not sufficient. Given a class, it should be possible to retrieve all of its subclasses, and all of its super



1589 classes. This requires a recursion mechanism in SQL queries. The freebXML implementation allows  
1590 various relational database products such as Oracle, PostgreSQL and MS SQL Server 2005 to be used  
1591 as the database. These products have different support for recursion mechanisms in SQL Queries.

1592 The All SuperClasses discovery query MAY be implemented by an ebXML Registry implementing this  
1593 profile. It allows the discovery of all super classes of a given ClassificationNode recursively in freebXML  
1594 Registry implementations using MS SQL Server 2005 as the database.

#### 1595 6.4.1 Parameter \$className

1596 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
1597 value of ClassificationNode.

#### 1598 6.4.2 Example of All SuperClasses Discovery Query

1599 The following example illustrates how to find all the super classes of a given class recursively that have  
1600 a name containing the string "AirReservationServices" if the query is implemented as an Adhoc Query .

```
1601 <rs:RequestSlotList>  
1602   <rim:Slot  
1603     name="urn:oasis:names:tc:ebxml-  
1604   regrep:3.0:rs:AdhocQueryRequest:queryId">  
1605     <rim:ValueList>  
1606       <rim:Value>urn:oasis:names:tc:ebxml-  
1607   regrep:query:FindAllSuperClasses</rim:Value>  
1608     </rim:ValueList>  
1609   </rim:Slot>  
1610   <rim:Slot name="urn:oasis:names:tc:ebxml-  
1611   regrep:rs:AdhocQueryRequest:queryId">  
1612     <rim:ValueList>  
1613       <rim:Value>urn:oasis:names:tc:ebxml-  
1614   regrep:query:FindAllSuperClasses</rim:Value>  
1615     </rim:ValueList>  
1616   </rim:Slot>  
1617   <rim:Slot name="$className">  
1618     <rim:ValueList>  
1619       <rim:Value>%AirReservationServices%</rim:Value>  
1620     </rim:ValueList>  
1621   </rim:Slot>  
1622 </rs:RequestSlotList>  
1623  
1624 <query:ResponseOption returnComposedObjects="true"  
1625   returnType="LeafClassWithRepositoryItem"/>  
1626  
1627 <rim:AdhocQuery id="temporaryId">  
1628   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
1629   regrep:QueryLanguage:SQL-92">  
1630     </rim:QueryExpression>  
1631 </rim:AdhocQuery>
```

1632 Example of All SuperClasses Discovery Query

#### 1633 6.5 All SubClasses Discovery Query

1634 The All SubClasses discovery query MAY be implemented by an ebXML Registry implementing this  
1635 profile. It allows the discovery of all subclasses of a given ClassificationNode recursively in a freebXML  
1636 Registry implementations supporting recursion.

#### 1637 6.5.1 Parameter \$className

1638 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
1639 value of ClassificationNode.

## 1640 6.5.2 Example of All SubClasses Discovery Query

1641 The following example illustrates how to find all the subclasses of a given class recursively that have a  
1642 name containing the string "AirServices", if the query is implemented as an Adhoc Query.

```
1643 <rs:RequestSlotList>
1644   <rim:Slot
1645     name="urn:oasis:names:tc:ebxml-
1646   regrep:3.0:rs:AdhocQueryRequest:queryId">
1647     <rim:ValueList>
1648       <rim:Value>urn:oasis:names:tc:ebxml-
1649   regrep:query:FindAllSubClasses</rim:Value>
1650     </rim:ValueList>
1651   </rim:Slot>
1652   <rim:Slot name="urn:oasis:names:tc:ebxml-
1653   regrep:rs:AdhocQueryRequest:queryId">
1654     <rim:ValueList>
1655       <rim:Value>urn:oasis:names:tc:ebxml-
1656   regrep:query:FindAllSubClasses</rim:Value>
1657     </rim:ValueList>
1658   </rim:Slot>
1659   <rim:Slot name="$className">
1660     <rim:ValueList>
1661       <rim:Value>%AirServices%</rim:Value>
1662     </rim:ValueList>
1663   </rim:Slot>
1664 </rs:RequestSlotList>
1665
1666 <query:ResponseOption returnComposedObjects="true"
1667   returnType="LeafClassWithRepositoryItem"/>
1668
1669 <rim:AdhocQuery id="temporaryId">
1670   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1671   regrep:QueryLanguage:SQL-92">
1672     </rim:QueryExpression>
1673 </rim:AdhocQuery>
```

1674 Example of All SubClasses Discovery Query

## 1675 6.6 EquivalentClasses Discovery Query

1676 The EquivalentClasses discovery query MUST be implemented by an ebXML Registry implementing this  
1677 profile. It allows the discovery of all the equivalent classes of a given ClassificationNode.

### 1678 6.6.1 Parameter \$className

1679 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
1680 value of ClassificationNodes.

### 1681 6.6.2 Example of EquivalentClasses Discovery Query

1682 The following example illustrates how to find all the equivalent classes of a given class that have a name  
1683 containing the string "AirServices" .

```
1684 <rs:RequestSlotList>
1685   <rim:Slot
1686     name="urn:oasis:names:tc:ebxml-
1687   regrep:3.0:rs:AdhocQueryRequest:queryId">
1688     <rim:ValueList>
1689       <rim:Value>urn:oasis:names:tc:ebxml-
1690   regrep:query:FindEquivalentClasses</rim:Value>
1691     </rim:ValueList>
1692   </rim:Slot>
1693   <rim:Slot name="urn:oasis:names:tc:ebxml-
1694   regrep:rs:AdhocQueryRequest:queryId">
1695     <rim:ValueList>
```

```

1696         <rim:Value>urn:oasis:names:tc:ebxml-
1697 regrep:query:FindEquivalentClasses</rim:Value>
1698         </rim:ValueList>
1699     </rim:Slot>
1700     <rim:Slot name="$className">
1701         <rim:ValueList>
1702             <rim:Value>%AirServices%</rim:Value>
1703         </rim:ValueList>
1704     </rim:Slot>
1705 </rs:RequestSlotList>
1706
1707 <query:ResponseOption returnComposedObjects="true"
1708     returnType="LeafClassWithRepositoryItem"/>
1709
1710 <rim:AdhocQuery id="temporaryId">
1711     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1712 regrep:QueryLanguage:SQL-92">
1713     </rim:QueryExpression>
1714 </rim:AdhocQuery>

```

1715 Example of Equivalent Classes Discovery Query

## 1716 6.7 EquivalentProperties Discovery Query

1717 The EquivalentProperties discovery query MUST be implemented by an ebXML Registry implementing  
1718 this profile. It allows the discovery of all the equivalent properties of a given Association that have  
1719 associationType of Property.

### 1720 6.7.1 Parameter \$propertyName

1721 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
1722 value of Associations that have associationType of Property

### 1723 6.7.2 Example of EquivalentProperties Discovery Query

1724 The following example illustrates how to find all the equivalent properties(Association Type) of a given  
1725 property (Association Type) that have a name containing the string "paymentMethods".

```

1726 <rs:RequestSlotList>
1727     <rim:Slot
1728         name="urn:oasis:names:tc:ebxml-
1729 regrep:3.0:rs:AdhocQueryRequest:queryId">
1730         <rim:ValueList>
1731             <rim:Value>urn:oasis:names:tc:ebxml-
1732 regrep:query:FindEquivalentProperties</rim:Value>
1733         </rim:ValueList>
1734     </rim:Slot>
1735     <rim:Slot name="urn:oasis:names:tc:ebxml-
1736 regrep:rs:AdhocQueryRequest:queryId">
1737         <rim:ValueList>
1738             <rim:Value>urn:oasis:names:tc:ebxml-
1739 regrep:query:FindEquivalentProperties</rim:Value>
1740         </rim:ValueList>
1741     </rim:Slot>
1742     <rim:Slot name="$propertyName">
1743         <rim:ValueList>
1744             <rim:Value>%paymentMethods%</rim:Value>
1745         </rim:ValueList>
1746     </rim:Slot>
1747 </rs:RequestSlotList>
1748
1749 <query:ResponseOption returnComposedObjects="true"
1750     returnType="LeafClassWithRepositoryItem"/>
1751
1752 <rim:AdhocQuery id="temporaryId">

```

```
1753     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1754     regrep:QueryLanguage:SQL-92">
1755         </rim:QueryExpression>
1756 </rim:AdhocQuery>
```

1757 Example of Equivalent Properties Discovery Query

## 1758 6.8 SameExtrinsicObjects Discovery Query

1759 The SameExtrinsicObjects discovery query MUST be implemented by an ebXML Registry implementing  
1760 this profile. It allows the discovery of all the "ExtrinsicObjects" defined to be the same with a given  
1761 ExtrinsicObject.

### 1762 6.8.1 Parameter \$extrinsicObjectName

1763 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
1764 value of ExtrinsicObjects.

### 1765 6.8.2 Example of SameExtrinsicObjects Discovery Query

1766 The following example illustrates how to find all the ExtrinsicObjects that are defined to be the same as  
1767 the ExtrinsicObject that have a name containing the string "MyDocument" .

```
1768 <rs:RequestSlotList>
1769     <rim:Slot
1770         name="urn:oasis:names:tc:ebxml-
1771         regrep:3.0:rs:AdhocQueryRequest:queryId">
1772         <rim:ValueList>
1773             <rim:Value>urn:oasis:names:tc:ebxml-
1774             regrep:query:FindTheSameExtrinsicObjects</rim:Value>
1775         </rim:ValueList>
1776     </rim:Slot>
1777     <rim:Slot name="urn:oasis:names:tc:ebxml-
1778     regrep:rs:AdhocQueryRequest:queryId">
1779         <rim:ValueList>
1780             <rim:Value>urn:oasis:names:tc:ebxml-
1781             regrep:query:FindTheSameExtrinsicObjects</rim:Value>
1782         </rim:ValueList>
1783     </rim:Slot>
1784     <rim:Slot name="$extrinsicObjectName">
1785         <rim:ValueList>
1786             <rim:Value>%MyDocument%</rim:Value>
1787         </rim:ValueList>
1788     </rim:Slot>
1789 </rs:RequestSlotList>
1790
1791 <query:ResponseOption returnComposedObjects="true"
1792     returnType="LeafClassWithRepositoryItem"/>
1793
1794 <rim:AdhocQuery id="temporaryId">
1795     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1796     regrep:QueryLanguage:SQL-92">
1797         </rim:QueryExpression>
1798 </rim:AdhocQuery>
```

1800 Example of SameExtrinsicObjects Discovery Query

## 1801 6.9 DifferentExtrinsicObjects Discovery Query

1802 The DifferentExtrinsicObjects discovery query MUST be implemented by an ebXML Registry  
1803 implementing this profile. It allows the discovery of all the "ExtrinsicObjects" defined to be the different  
1804 from a given ExtrinsicObject.

## 1805 6.9.1 Parameter \$extrinsicObjectName

1806 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
1807 value of ExtrinsicObjects.

## 1808 6.9.2 Example of DifferentExtrinsicObjects Discovery Query

1809 The following example illustrates how to find all the ExtrinsicObjects that are defined to be different from  
1810 the ExtrinsicObject that have a name containing the string "MyDocument" .

```
1811 <rs:RequestSlotList>
1812   <rim:Slot
1813     name="urn:oasis:names:tc:ebxml-
1814   regrep:3.0:rs:AdhocQueryRequest:queryId">
1815     <rim:ValueList>
1816       <rim:Value>urn:oasis:names:tc:ebxml-
1817   regrep:query:FindDifferentExtrinsicObjects</rim:Value>
1818     </rim:ValueList>
1819   </rim:Slot>
1820   <rim:Slot name="urn:oasis:names:tc:ebxml-
1821   regrep:rs:AdhocQueryRequest:queryId">
1822     <rim:ValueList>
1823       <rim:Value>urn:oasis:names:tc:ebxml-
1824   regrep:query:FindDifferentExtrinsicObjects</rim:Value>
1825     </rim:ValueList>
1826   </rim:Slot>
1827   <rim:Slot name="$extrinsicObjectName">
1828     <rim:ValueList>
1829       <rim:Value>%MyDocument%</rim:Value>
1830     </rim:ValueList>
1831   </rim:Slot>
1832 </rs:RequestSlotList>
1833
1834 <query:ResponseOption returnComposedObjects="true"
1835   returnType="LeafClassWithRepositoryItem"/>
1836
1837 <rim:AdhocQuery id="temporaryId">
1838   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1839   regrep:QueryLanguage:SQL-92">
1840     </rim:QueryExpression>
1841 </rim:AdhocQuery>
1842
```

1843 Example of DifferentExtrinsicObjects Discovery Query

## 1844 6.10 AllDifferentRegistryObject Discovery Query

1845 The AllDifferentRegistryObjects discovery query MUST be implemented by an ebXML Registry  
1846 implementing this profile. Given a RegistryObject, it allows the discovery of all the other member  
1847 "RegistryObjects" of a Registry package that are defined to be the different from each other through a  
1848 allDifferent slot.

### 1849 6.10.1 Parameter \$registryObjectName

1850 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
1851 value of RegistryObjects.

### 1852 6.10.2 Example of AllDifferentRegistryObjects Discovery Query

1853 The following example illustrates how to find all the RegistryObjects that are defined to be different from  
1854 the RegistryObject that have a name containing the string "MyDocument" .

```
1855 <rs:RequestSlotList>
1856   <rim:Slot
```

```

1858         name="urn:oasis:names:tc:ebxml-
1859 regrep:3.0:rs:AdhocQueryRequest:queryId">
1860         <rim:ValueList>
1861             <rim:Value>urn:oasis:names:tc:ebxml-
1862 regrep:query:FindAllDifferent</rim:Value>
1863         </rim:ValueList>
1864     </rim:Slot>
1865     <rim:Slot name="urn:oasis:names:tc:ebxml-
1866 regrep:rs:AdhocQueryRequest:queryId">
1867         <rim:ValueList>
1868             <rim:Value>urn:oasis:names:tc:ebxml-
1869 regrep:query:FindAllDifferent</rim:Value>
1870         </rim:ValueList>
1871     </rim:Slot>
1872     <rim:Slot name="$registryObjectName">
1873         <rim:ValueList>
1874             <rim:Value>%MyDocument%</rim:Value>
1875         </rim:ValueList>
1876     </rim:Slot>
1877 </rs:RequestSlotList>
1878
1879 <query:ResponseOption returnComposedObjects="true"
1880     returnType="LeafClassWithRepositoryItem"/>
1881
1882 <rim:AdhocQuery id="temporaryId">
1883     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1884 regrep:QueryLanguage:SQL-92">
1885     </rim:QueryExpression>
1886 </rim:AdhocQuery>

```

1887 Example of AllDifferentRegistryObjects Discovery Query

## 1888 6.11 ObjectProperties Discovery Query

1889 The ObjectProperties discovery query MUST be implemented by an ebXML Registry implementing this  
1890 profile. It allows the discovery of all of the objectProperties of a given classification node.

### 1891 6.11.1 Parameter \$className

1892 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
1893 value of ClassificationNodes.

### 1894 6.11.2 Example of ObjectProperties Discovery Query

1895 The following example illustrates how to find all the object properties of a given classification node having  
1896 a name containing "AirServices" .

```

1897
1898 <rs:RequestSlotList>
1899     <rim:Slot
1900         name="urn:oasis:names:tc:ebxml-
1901 regrep:3.0:rs:AdhocQueryRequest:queryId">
1902         <rim:ValueList>
1903             <rim:Value>urn:oasis:names:tc:ebxml-
1904 regrep:query:FindObjectProperties</rim:Value>
1905         </rim:ValueList>
1906     </rim:Slot>
1907     <rim:Slot name="urn:oasis:names:tc:ebxml-
1908 regrep:rs:AdhocQueryRequest:queryId">
1909         <rim:ValueList>
1910             <rim:Value>urn:oasis:names:tc:ebxml-
1911 regrep:query:FindObjectProperties</rim:Value>
1912         </rim:ValueList>
1913     </rim:Slot>
1914     <rim:Slot name="$className">

```

```

1915     <rim:ValueList>
1916         <rim:Value>%AirServices%</rim:Value>
1917     </rim:ValueList>
1918 </rim:Slot>
1919 </rs:RequestSlotList>
1920
1921 <query:ResponseOption returnComposedObjects="true"
1922     returnType="LeafClassWithRepositoryItem"/>
1923
1924 <rim:AdhocQuery id="temporaryId">
1925     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1926     regrep:QueryLanguage:SQL-92">
1927         </rim:QueryExpression>
1928 </rim:AdhocQuery>

```

Example of ObjectProperties Discovery Query

## 1930 6.12 ImmediateInheritedObjectProperties Discovery Query

1931 The ImmediateInheritedObjectProperties discovery query MUST be implemented by an ebXML Registry  
 1932 implementing this profile. It allows the discovery of all of the objectProperties of a given classification  
 1933 node including the ones inherited from its immediate super classes.

### 1934 6.12.1 Parameter \$className

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

## 1937 6.12.2 Example of ImmediateInheritedObjectProperties Discovery Query

1938 The following example illustrates how to find all the object properties of a given classification node having  
 1939 a name containing "AirServices" including the ones inherited from its immediate super classes.

```

1940
1941 <rs:RequestSlotList>
1942     <rim:Slot
1943         name="urn:oasis:names:tc:ebxml-
1944     regrep:3.0:rs:AdhocQueryRequest:queryId">
1945         <rim:ValueList>
1946             <rim:Value>urn:oasis:names:tc:ebxml-
1947     regrep:query:FindImmediateInheritedObjectProperties</rim:Value>
1948         </rim:ValueList>
1949     </rim:Slot>
1950     <rim:Slot name="urn:oasis:names:tc:ebxml-
1951     regrep:rs:AdhocQueryRequest:queryId">
1952         <rim:ValueList>
1953             <rim:Value>urn:oasis:names:tc:ebxml-
1954     regrep:query:FindImmediateInheritedObjectProperties</rim:Value>
1955         </rim:ValueList>
1956     </rim:Slot>
1957     <rim:Slot name="$className">
1958         <rim:ValueList>
1959             <rim:Value>%AirServices%</rim:Value>
1960         </rim:ValueList>
1961     </rim:Slot>
1962 </rs:RequestSlotList>
1963
1964 <query:ResponseOption returnComposedObjects="true"
1965     returnType="LeafClassWithRepositoryItem"/>
1966
1967 <rim:AdhocQuery id="temporaryId">
1968     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1969     regrep:QueryLanguage:SQL-92">
1970         </rim:QueryExpression>
1971 </rim:AdhocQuery>

```

1972

## Example of ImmediateInheritedObjectProperties Discovery Query

### 1973 6.13 AllInheritedObjectProperties Discovery Query

1974 It should be noted that, given a class, finding the object properties inherited from immediate super  
1975 classes is necessary but not sufficient. Given a class, it should be possible to retrieve all of the object  
1976 properties inherited from its super classes. This requires a recursion mechanism in SQL queries. The  
1977 freebXML implementation allows various relational database products such as Oracle, PostgreSQL and  
1978 MS SQL Server 2005 to be used as the database. These products have different support for recursion in  
1979 SQL Queries.

1980 The AllInheritedObjectProperties discovery query MAY be implemented by an ebXML Registry  
1981 implementing this profile. It allows the discovery of all inherited ObjectProperties recursively of a given  
1982 ClassificationNode in a ClassificationScheme in freebXML Registry implementations using MS SQL  
1983 Server 2005 as the database.

#### 1984 6.13.1 Parameter \$className

1985 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
1986 value of ClassificationNodes.

#### 1987 6.13.2 Example of AllInheritedObjectProperties Discovery Query

1988 The following example illustrates how to find all the object properties of a given classification node having  
1989 a name containing "AirReservationServices" including the ones inherited from all of its super classes  
1990 recursively, if the query is implemented as an Adhoc Query.

1991

```
1992 <rs:RequestSlotList>  
1993   <rim:Slot  
1994     name="urn:oasis:names:tc:ebxml-  
1995   regrep:3.0:rs:AdhocQueryRequest:queryId">  
1996     <rim:ValueList>  
1997       <rim:Value>urn:oasis:names:tc:ebxml-  
1998   regrep:query:FindAllInheritedObjectProperties</rim:Value>  
1999     </rim:ValueList>  
2000   </rim:Slot>  
2001   <rim:Slot name="urn:oasis:names:tc:ebxml-  
2002   regrep:rs:AdhocQueryRequest:queryId">  
2003     <rim:ValueList>  
2004       <rim:Value>urn:oasis:names:tc:ebxml-regrep:query:FindAll  
2005   InheritedObjectProperties</rim:Value>  
2006     </rim:ValueList>  
2007   </rim:Slot>  
2008   <rim:Slot name="$className">  
2009     <rim:ValueList>  
2010       <rim:Value>%AirReservationServices%</rim:Value>  
2011     </rim:ValueList>  
2012   </rim:Slot>  
2013 </rs:RequestSlotList>  
2014  
2015 <query:ResponseOption returnComposedObjects="true"  
2016   returnType="LeafClassWithRepositoryItem"/>  
2017  
2018 <rim:AdhocQuery id="temporaryId">  
2019   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
2020   regrep:QueryLanguage:SQL-92">  
2021     </rim:QueryExpression>  
2022 </rim:AdhocQuery>
```

2023

#### Example of AllInheritedObjectProperties Discovery Query



## 2024 6.14 DatatypeProperties Discovery Query

2025 The DatatypeProperties discovery query MUST be implemented by an ebXML Registry implementing this  
2026 profile. It allows the discovery of all of the datatypeProperties of a given classification node.

### 2027 6.14.1 Parameter \$className

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

### 2030 6.14.2 Example of DatatypeProperties Discovery Query

2031 The following example illustrates how to find all the datatype properties of a given classification node  
2032 having a name containing "AirReservationServices".

2033

```
2034 <rs:RequestSlotList>  
2035   <rim:Slot  
2036     name="urn:oasis:names:tc:ebxml-  
2037   regrep:3.0:rs:AdhocQueryRequest:queryId">  
2038     <rim:ValueList>  
2039       <rim:Value>urn:oasis:names:tc:ebxml-  
2040   regrep:query:FindDatatypeProperties</rim:Value>  
2041     </rim:ValueList>  
2042   </rim:Slot>  
2043   <rim:Slot name="urn:oasis:names:tc:ebxml-  
2044   regrep:rs:AdhocQueryRequest:queryId">  
2045     <rim:ValueList>  
2046       <rim:Value>urn:oasis:names:tc:ebxml-  
2047   regrep:query:FindDatatypeProperties</rim:Value>  
2048     </rim:ValueList>  
2049   </rim:Slot>  
2050   <rim:Slot name="$className">  
2051     <rim:ValueList>  
2052       <rim:Value>%AirReservationServices%</rim:Value>  
2053     </rim:ValueList>  
2054   </rim:Slot>  
2055 </rs:RequestSlotList>  
2056  
2057 <query:ResponseOption returnComposedObjects="true"  
2058   returnType="LeafClassWithRepositoryItem"/>  
2059  
2060 <rim:AdhocQuery id="temporaryId">  
2061   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
2062   regrep:QueryLanguage:SQL-92">  
2063     </rim:QueryExpression>  
2064 </rim:AdhocQuery>
```

2065 Example of DatatypeProperties Discovery Query

## 2066 6.15 AllInheritedDatatypeProperties Discovery Query

2067 It should be noted that, given a class, finding the datatype properties inherited from immediate super  
2068 classes is necessary but not sufficient. Given a class, it should be possible to retrieve all of the datatype  
2069 properties inherited from its super classes. This requires a recursion mechanism in SQL queries. The  
2070 freebXML implementation allows various relational database products such as Oracle, PostgreSQL and  
2071 MS SQL Server 2005 to be used as the database. These products have different support for recursion in  
2072 SQL Queries.

2073 The AllInheritedDatatypeProperties discovery query MAY be implemented by an ebXML Registry  
2074 implementing this profile. It allows the discovery of all inherited DatatypeProperties recursively of a given  
2075 ClassificationNode in a ClassificationScheme in freebXML Registry implementations using MS SQL  
2076 Server 2005 as the database.

### 2077 6.15.1 Parameter \$className

2078 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
2079 value of ClassificationNodes.

### 2080 6.15.2 Example of AllInheritedDatatypeProperties Discovery Query

2081 The following example illustrates how to find all the datatype properties of a given classification node  
2082 having a name containing "AirReservationServices" including the ones inherited from all of its super  
2083 classes recursively, if the query is implemented as an Adhoc Query.

2084

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

2116 Example of AllInheritedDatatypeProperties Discovery Query

### 2117 6.16 TransitiveRelationships Discovery Query

2118 To make any use of the transitive property in ebXML registries, coding is necessary to find out the  
2119 implied information. The TransitiveRelationships discovery query MUST be implemented by an ebXML  
2120 Registry implementing this profile to handle this semantics.

2121 Given a class which is a source of a transitive property, this discovery query retrieves not only the target  
2122 objects of a given transitive property, but if the target objects have the same property, it retrieves their  
2123 target objects too.

### 2124 6.16.1 Parameter \$className

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

### 2127 6.16.2 Parameter \$propertyName

2128 This parameter's value SHALL specify a string containing a pattern match against the name attribute

2129 value of Associations that have associationType of Property

### 2130 6.16.3 Example of TransitiveRelationships Discovery Query

2131 The following example illustrates how to retrieve all the target objects of the “succeeds” property of the  
2132 “AirReservationServices” including the target objects implied by a transitive property relationship.

2133

```
2134 <rs:RequestSlotList>
2135   <rim:Slot
2136     name="urn:oasis:names:tc:ebxml-
2137   regrep:3.0:rs:AdhocQueryRequest:queryId">
2138     <rim:ValueList>
2139       <rim:Value>urn:oasis:names:tc:ebxml-
2140   regrep:query:FindTransitiveRelationships</rim:Value>
2141     </rim:ValueList>
2142   </rim:Slot>
2143   <rim:Slot name="urn:oasis:names:tc:ebxml-
2144   regrep:rs:AdhocQueryRequest:queryId">
2145     <rim:ValueList>
2146       <rim:Value>urn:oasis:names:tc:ebxml-
2147   regrep:query:FindTransitiveRelationships</rim:Value>
2148     </rim:ValueList>
2149   </rim:Slot>
2150   <rim:Slot name="$className">
2151     <rim:ValueList>
2152       <rim:Value>%AirReservationServices%</rim:Value>
2153     </rim:ValueList>
2154   </rim:Slot>
2155   <rim:Slot name="$propertyName">
2156     <rim:ValueList>
2157       <rim:Value>%succeeds%</rim:Value>
2158     </rim:ValueList>
2159   </rim:Slot>
2160 </rs:RequestSlotList>
2161
2162 <query:ResponseOption returnComposedObjects="true"
2163   returnType="LeafClassWithRepositoryItem"/>
2164
2165 <rim:AdhocQuery id="temporaryId">
2166   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2167   regrep:QueryLanguage:SQL-92">
2168     </rim:QueryExpression>
2169 </rim:AdhocQuery>
```

2170 Example of TransitiveRelationships Discovery Query

### 2171 6.17 TargetObjects Discovery Query

2172 The TargetObjects discovery query MUST be implemented by an ebXML Registry implementing this  
2173 profile. It allows the discovery of the targetObjects from the Registry, given a Classification Node  
2174 (sourceObject) and a property name (Association Type).

#### 2175 6.17.1 Parameter \$className

2176 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
2177 value of ClassificationNodes.

#### 2178 6.17.2 Parameter \$propertyName

2179 This parameter's value SHALL specify a string containing a pattern match against the name attribute  
2180 value of Associations that have associationType of Property.

### 2181 6.17.3 Example of TargetObjects Discovery Query

2182 The following example illustrates how to retrieve all the target objects of the "paymentMethod" property of  
2183 the "AirReservationServices".

2184

```
2185 <rs:RequestSlotList>
2186   <rim:Slot
2187     name="urn:oasis:names:tc:ebxml-
2188   regrep:3.0:rs:AdhocQueryRequest:queryId">
2189     <rim:ValueList>
2190       <rim:Value>urn:oasis:names:tc:ebxml-
2191   regrep:query:FindTargetObjects</rim:Value>
2192     </rim:ValueList>
2193   </rim:Slot>
2194   <rim:Slot name="urn:oasis:names:tc:ebxml-
2195   regrep:rs:AdhocQueryRequest:queryId">
2196     <rim:ValueList>
2197       <rim:Value>urn:oasis:names:tc:ebxml-
2198   regrep:query:FindTargetObjects</rim:Value>
2199     </rim:ValueList>
2200   </rim:Slot>
2201   <rim:Slot name="$className">
2202     <rim:ValueList>
2203       <rim:Value>%AirReservationServices%</rim:Value>
2204     </rim:ValueList>
2205   </rim:Slot>
2206   <rim:Slot name="$propertyName">
2207     <rim:ValueList>
2208       <rim:Value>%paymentMethod%</rim:Value>
2209     </rim:ValueList>
2210   </rim:Slot>
2211 </rs:RequestSlotList>
2212
2213 <query:ResponseOption returnComposedObjects="true"
2214   returnType="LeafClassWithRepositoryItem"/>
2215
2216 <rim:AdhocQuery id="temporaryId">
2217   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2218   regrep:QueryLanguage:SQL-92">
2219     </rim:QueryExpression>
2220 </rim:AdhocQuery>
```

2221 Example of TargetObjects Discovery Query

2222

### 2223 6.18 TargetObjectsInverseOf Discovery Query

2224 The TargetObjectsInverseOf discovery query MUST be implemented by an ebXML Registry  
2225 implementing this profile. Given a Classification Node (sourceObject) and a property name (Association  
2226 Type), this query retrieves the source objects of the properties which are stated to be inverseOf the  
2227 property name given as a parameter, and considering the Classification Node name as the targetObject  
2228 of these properties.

#### 2229 6.18.1 Parameter \$className

2230 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
2231 value of ClassificationNodes.

#### 2232 6.18.2 Parameter \$propertyName

2233 This parameter's value SHALL specify a string containing a pattern match against the name attribute  
2234 value of Associations that have associationType of Property.

### 2235 6.18.3 Example of TargetObjectsInverseOf Discovery Query

2236 The following example illustrates how to retrieve all the source objects of the properties which are stated  
2237 to the the inverseOf the property "succeeds", considering the "AirReservationServices" as the target  
2238 object of these properties.

2239

```
2240 <rs:RequestSlotList>
2241   <rim:Slot
2242     name="urn:oasis:names:tc:ebxml-
2243     regrep:3.0:rs:AdhocQueryRequest:queryId">
2244     <rim:ValueList>
2245       <rim:Value>urn:oasis:names:tc:ebxml-
2246       regrep:query:FindTOinverseOf</rim:Value>
2247     </rim:ValueList>
2248   </rim:Slot>
2249   <rim:Slot name="urn:oasis:names:tc:ebxml-
2250   regrep:rs:AdhocQueryRequest:queryId">
2251     <rim:ValueList>
2252       <rim:Value>urn:oasis:names:tc:ebxml-
2253       regrep:query:FindTOinverseOf</rim:Value>
2254     </rim:ValueList>
2255   </rim:Slot>
2256   <rim:Slot name="$className">
2257     <rim:ValueList>
2258       <rim:Value>%AirReservationServices%</rim:Value>
2259     </rim:ValueList>
2260   </rim:Slot>
2261   <rim:Slot name="$propertyName">
2262     <rim:ValueList>
2263       <rim:Value>%succeeds%</rim:Value>
2264     </rim:ValueList>
2265   </rim:Slot>
2266 </rs:RequestSlotList>
2267
2268 <query:ResponseOption returnComposedObjects="true"
2269   returnType="LeafClassWithRepositoryItem"/>
2270
2271 <rim:AdhocQuery id="temporaryId">
2272   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2273   regrep:QueryLanguage:SQL-92">
2274     </rim:QueryExpression>
2275 </rim:AdhocQuery>
```

2276 Example of TargetObjectsInverseOf Discovery Query

2277

## 2278 6.19 InverseRanges Discovery Query

2279 The InverseRanges discovery query MUST be implemented by an ebXML Registry implementing this  
2280 profile to handle this semantics. Given a Classification Node (sourceObject) and a property name  
2281 (Association Type), this query retrieves not only the target objects of this property, but also the source  
2282 objects of the properties which are stated to be inverseOf the property name given as a parameter, and  
2283 considering the Classification Node name as the targetObject of these properties. This query can be  
2284 thought as the union of the queries presented in Sections 6.17 and 6.18.

### 2285 6.19.1 Parameter \$className

2286 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
2287 value of ClassificationNodes.

### 2288 6.19.2 Parameter \$propertyName

2289 This parameter's value SHALL specify a string containing a pattern match against the name attribute

2290 value of Associations that have associationType of Property

### 2291 6.19.3 Example of InverseRanges Discovery Query

2292 Consider, for example, the "succeeds" property defined in Section 4.3.3. To denote that a certain Web  
2293 service instance precedes another during execution, we may define the "precedes" property as an  
2294 inverse of the "succeeds" property as follows:

2295

```
2296 <owl:ObjectProperty rdf:ID="precedes">  
2297   <owl:inverseOf rdf:resource="#succeeds" />  
2298 </owl:ObjectProperty>
```

#### 2299 Example owl:inverseOf Property

2300 Assume that we want to find all the Web services which can succeed a given Web service. In such a  
2301 case, we need not only find all the Web services which succeeds this given Web service, that is the  
2302 target objects of "succeeds" Association instance, but we also need to find all the sourceObjects of the  
2303 "precedes" Association instance since "precedes" is declared to be the "inverseOf" succeeds Association  
2304 instance.

2305 The following example illustrates how to retrieve all the services that "succeeds"  
2306 "AirReservationServices" by also making use of its "precedes" property.

2307

```
2308 <rs:RequestSlotList>  
2309   <rim:Slot  
2310     name="urn:oasis:names:tc:ebxml-  
2311     regrep:3.0:rs:AdhocQueryRequest:queryId">  
2312     <rim:ValueList>  
2313       <rim:Value>urn:oasis:names:tc:ebxml-  
2314       regrep:query:FindInverseRanges</rim:Value>  
2315     </rim:ValueList>  
2316   </rim:Slot>  
2317   <rim:Slot name="urn:oasis:names:tc:ebxml-  
2318   regrep:rs:AdhocQueryRequest:queryId">  
2319     <rim:ValueList>  
2320       <rim:Value>urn:oasis:names:tc:ebxml-  
2321       regrep:query:FindInverseRanges</rim:Value>  
2322     </rim:ValueList>  
2323   </rim:Slot>  
2324   <rim:Slot name="$className">  
2325     <rim:ValueList>  
2326       <rim:Value>%AirReservationServices%</rim:Value>  
2327     </rim:ValueList>  
2328   </rim:Slot>  
2329   <rim:Slot name="$propertyName">  
2330     <rim:ValueList>  
2331       <rim:Value>%succeeds%</rim:Value>  
2332     </rim:ValueList>  
2333   </rim:Slot>  
2334 </rs:RequestSlotList>  
2335  
2336 <query:ResponseOption returnComposedObjects="true"  
2337   returnType="LeafClassWithRepositoryItem"/>  
2338  
2339 <rim:AdhocQuery id="temporaryId">  
2340   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
2341   regrep:QueryLanguage:SQL-92">  
2342     </rim:QueryExpression>  
2343 </rim:AdhocQuery>
```

2344

#### Example of InverseRanges Discovery Query

## 2345 6.20 SymmetricProperties Discovery Query

2346 The SymmetricProperties discovery query MUST be implemented by an ebXML Registry implementing  
2347 this profile. It allows the discovery of all of the Symmetric Properties of a given classification node.

### 2348 6.20.1 Parameter \$className

2349 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
2350 value of ClassificationNodes.

### 2351 6.20.2 Example of SymmetricProperties Discovery Query

2352 The following example illustrates how to find all the symmetric properties of a given classification node  
2353 having a name containing "AirReservationServices".

2354

```
2355 <rs:RequestSlotList>  
2356   <rim:Slot  
2357     name="urn:oasis:names:tc:ebxml-  
2358   regrep:3.0:rs:AdhocQueryRequest:queryId">  
2359     <rim:ValueList>  
2360       <rim:Value>urn:oasis:names:tc:ebxml-  
2361   regrep:query:FindSymmetricProperties</rim:Value>  
2362     </rim:ValueList>  
2363   </rim:Slot>  
2364   <rim:Slot name="urn:oasis:names:tc:ebxml-  
2365   regrep:rs:AdhocQueryRequest:queryId">  
2366     <rim:ValueList>  
2367       <rim:Value>urn:oasis:names:tc:ebxml-  
2368   regrep:query:FindSymmetricProperties</rim:Value>  
2369     </rim:ValueList>  
2370   </rim:Slot>  
2371   <rim:Slot name="$className">  
2372     <rim:ValueList>  
2373       <rim:Value>%AirReservationServices%</rim:Value>  
2374     </rim:ValueList>  
2375   </rim:Slot>  
2376 </rs:RequestSlotList>  
  
2377  
2378 <query:ResponseOption returnComposedObjects="true"  
2379   returnType="LeafClassWithRepositoryItem"/>  
2380  
2381 <rim:AdhocQuery id="temporaryId">  
2382   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
2383   regrep:QueryLanguage:SQL-92">  
2384     </rim:QueryExpression>  
2385 </rim:AdhocQuery>
```

2386

Example of SymmetricProperties Discovery Query

## 2387 6.21 FunctionalProperties Discovery Query

2388 The FunctionalProperties discovery query MUST be implemented by an ebXML Registry implementing  
2389 this profile. It allows the discovery of all of the Functional Properties of a given classification node.

### 2390 6.21.1 Parameter \$className

2391 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
2392 value of ClassificationNodes.

### 2393 6.21.2 Example of FunctionalProperties Discovery Query

2394 The following example illustrates how to find all the functional properties of a given classification node  
2395 having a name containing "AirReservationServices".

2396

```
2397 <rs:RequestSlotList>
2398   <rim:Slot
2399     name="urn:oasis:names:tc:ebxml-
2400   regrep:3.0:rs:AdhocQueryRequest:queryId">
2401     <rim:ValueList>
2402       <rim:Value>urn:oasis:names:tc:ebxml-
2403   regrep:query:FindFunctionalProperties</rim:Value>
2404     </rim:ValueList>
2405   </rim:Slot>
2406   <rim:Slot name="urn:oasis:names:tc:ebxml-
2407   regrep:rs:AdhocQueryRequest:queryId">
2408     <rim:ValueList>
2409       <rim:Value>urn:oasis:names:tc:ebxml-
2410   regrep:query:FindFunctionalProperties</rim:Value>
2411     </rim:ValueList>
2412   </rim:Slot>
2413   <rim:Slot name="$className">
2414     <rim:ValueList>
2415       <rim:Value>%AirReservationServices%</rim:Value>
2416     </rim:ValueList>
2417   </rim:Slot>
2418 </rs:RequestSlotList>
2419
2420 <query:ResponseOption returnComposedObjects="true"
2421   returnType="LeafClassWithRepositoryItem"/>
2422
2423 <rim:AdhocQuery id="temporaryId">
2424   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2425   regrep:QueryLanguage:SQL-92">
2426     </rim:QueryExpression>
2427 </rim:AdhocQuery>
```

2428 Example of Functional Properties Discovery Query

## 2429 6.22 InverseFunctionalProperties Discovery Query

2430 The InverseFunctionalProperties discovery query MUST be implemented by an ebXML Registry  
2431 implementing this profile. It allows the discovery of all of the Inverse Functional Properties of a given  
2432 classification node.

### 2433 6.22.1 Parameter \$className

2434 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
2435 value of ClassificationNodes.

### 2436 6.22.2 Example of InverseFunctionalProperties Discovery Query

2437 The following example illustrates how to find all the inverse functional properties of a given classification  
2438 node having a name containing "AirReservationServices".

2439

```
2440 <rs:RequestSlotList>
2441   <rim:Slot
2442     name="urn:oasis:names:tc:ebxml-
2443   regrep:3.0:rs:AdhocQueryRequest:queryId">
2444     <rim:ValueList>
2445       <rim:Value>urn:oasis:names:tc:ebxml-
2446   regrep:query:FindInverseFunctionalProperties</rim:Value>
2447     </rim:ValueList>
2448   </rim:Slot>
2449   <rim:Slot name="urn:oasis:names:tc:ebxml-
2450   regrep:rs:AdhocQueryRequest:queryId">
2451     <rim:ValueList>
```



```

2452         <rim:Value>urn:oasis:names:tc:ebxml-
2453 regrep:query:FindInverseFunctionalProperties</rim:Value>
2454     </rim:ValueList>
2455 </rim:Slot>
2456 <rim:Slot name="$className">
2457     <rim:ValueList>
2458         <rim:Value>%AirReservationServices%</rim:Value>
2459     </rim:ValueList>
2460 </rim:Slot>
2461 </rs:RequestSlotList>
2462
2463 <query:ResponseOption returnComposedObjects="true"
2464     returnType="LeafClassWithRepositoryItem"/>
2465
2466 <rim:AdhocQuery id="temporaryId">
2467     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2468 regrep:QueryLanguage:SQL-92">
2469         </rim:QueryExpression>
2470 </rim:AdhocQuery>

```

2471 Example of InverseFunctional Properties Discovery Query

## 2472 6.23 Instances Discovery Query

2473 When an intersection definition is used to create a complex class (a new ClassificationNode) in RIM as  
2474 described in Section 4.6, it becomes possible to infer that the objects (instances) classified by both of the  
2475 classes (ClassificationNodes) constituting the intersection are also the instances of this complex class.

2476 The Instances discovery query MUST be implemented by an ebXML Registry implementing this profile. It  
2477 allows the discovery of all of the direct instances of a given classification node and if it is a complex class  
2478 which is an intersection two classes, it also allows to retrieve the intersection of the instances of both of  
2479 the classes involved in the intersection definition.

### 2480 6.23.1 Parameter \$className

2481 This parameter's value SHALL specify a string containing a pattern to match against the name attribute  
2482 value of ClassificationNodes.

### 2483 6.23.2 Example of Instances Discovery Query

2484 Consider the "AirReservationServices" definition presented in Section 4.6. The following example  
2485 illustrates how to find all the direct instances of the "AirReservationServices" and also the instances  
2486 classified by both "AirServices" and also the "ReservationServices".

```

2487
2488 <rs:RequestSlotList>
2489     <rim:Slot
2490         name="urn:oasis:names:tc:ebxml-
2491 regrep:3.0:rs:AdhocQueryRequest:queryId">
2492         <rim:ValueList>
2493             <rim:Value>urn:oasis:names:tc:ebxml-
2494 regrep:query:FindInstances</rim:Value>
2495         </rim:ValueList>
2496     </rim:Slot>
2497     <rim:Slot name="urn:oasis:names:tc:ebxml-
2498 regrep:rs:AdhocQueryRequest:queryId">
2499         <rim:ValueList>
2500             <rim:Value>urn:oasis:names:tc:ebxml-
2501 regrep:query:FindInstances</rim:Value>
2502         </rim:ValueList>
2503     </rim:Slot>
2504     <rim:Slot name="$className">
2505         <rim:ValueList>
2506             <rim:Value>%AirReservationServices%</rim:Value>
2507         </rim:ValueList>

```

```
2508     </rim:Slot>
2509 </rs:RequestSlotList>
2510
2511 <query:ResponseOption returnComposedObjects="true"
2512     returnType="LeafClassWithRepositoryItem"/>
2513
2514 <rim:AdhocQuery id="temporaryId">
2515     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2516     regrep:QueryLanguage:SQL-92">
2517         </rim:QueryExpression>
2518     </rim:AdhocQuery>
```

2519

Example of Instances Discovery Query

---

## 2520 7 Canonical Metadata Definitions

2521 This chapter specifies the canonical metadata defined by this profile.

### 2522 7.1 ObjectType Extensions

2523 The following new extensions to the canonical ObjectType ClassificationScheme are described by this  
2524 profile:

2525

```
2526 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2527 regrep:ObjectType:RegistryObject:ExtrinsicObject"  
2528 lid="urn:oasis:names:tc:ebxml-  
2529 regrep:ObjectType:RegistryObject:ExtrinsicObject:OWL" code="OWL"  
2530 id="urn:oasis:names:tc:ebxml-  
2531 regrep:ObjectType:RegistryObject:ExtrinsicObject:OWL">  
2532 <rim:Name>  
2533 <rim:LocalizedString charset="UTF-8" value="label.OWL"/>  
2534 </rim:Name>  
2535 </rim:ClassificationNode>
```

### 2536 7.2 AssociationType Extensions

2537 The following new extensions to the AssociationType ClassificationScheme are described by this profile:

2538

```
2539 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2540 regrep:classificationScheme:AssociationType"  
2541 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:ObjectProperty"  
2542 code="ObjectProperty" id="urn:oasis:names:tc:ebxml-  
2543 regrep:AssociationType:ObjectProperty">  
2544 <rim:Name>  
2545 <rim:LocalizedString charset="UTF-8"  
2546 value="ObjectProperty"/>  
2547 </rim:Name>  
2548 </rim:ClassificationNode>  
2549 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2550 regrep:classificationScheme:AssociationType"  
2551 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:Property"  
2552 code="Property" id="urn:oasis:names:tc:ebxml-  
2553 regrep:AssociationType:Property">  
2554 <rim:Name>  
2555 <rim:LocalizedString charset="UTF-8" value="Property"/>  
2556 </rim:Name>  
2557 </rim:ClassificationNode>  
2558 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2559 regrep:classificationScheme:AssociationType"  
2560 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:SubPropertyOf"  
2561 code="SubPropertyOf" id="urn:oasis:names:tc:ebxml-  
2562 regrep:AssociationType:SubPropertyOf">  
2563 <rim:Name>  
2564 <rim:LocalizedString charset="UTF-8" value="SubPropertyOf"/>  
2565 </rim:Name>  
2566 </rim:ClassificationNode>  
2567 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
2568 regrep:classificationScheme:AssociationType"  
2569 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:SubClassOf"  
2570 code="SubClassOf" id="urn:oasis:names:tc:ebxml-  
2571 regrep:AssociationType:SubClassOf">  
2572 <rim:Name>  
2573 <rim:LocalizedString charset="UTF-8" value="SubClassOf"/>  
2574 </rim:Name>  
2575 </rim:ClassificationNode>  
2576
```

```

2577 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2578 regrep:classificationScheme:AssociationType"
2579 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:IntersectionOf"
2580 code="IntersectionOf" id="urn:oasis:names:tc:ebxml-
2581 regrep:AssociationType:IntersectionOf">
2582   <rim:Name>
2583     <rim:LocalizedString charset="UTF-8"
2584 value="IntersectionOf"/>
2585   </rim:Name>
2586 </rim:ClassificationNode>
2587
2588 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2589 regrep:classificationScheme:AssociationType"
2590 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:SameAs"
2591 code="SameAs" id="urn:oasis:names:tc:ebxml-
2592 regrep:AssociationType:SameAs">
2593   <rim:Name>
2594     <rim:LocalizedString charset="UTF-8" value="SameAs"/>
2595   </rim:Name>
2596 </rim:ClassificationNode>
2597
2598 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2599 regrep:classificationScheme:AssociationType"
2600 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:restriction"
2601 code="restriction" id="urn:oasis:names:tc:ebxml-
2602 regrep:AssociationType:restriction">
2603   <rim:Name>
2604     <rim:LocalizedString charset="UTF-8" value="restriction"/>
2605   </rim:Name>
2606 </rim:ClassificationNode>
2607
2608 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2609 regrep:classificationScheme:AssociationType"
2610 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:DifferentFrom"
2611 code="DifferentFrom" id="urn:oasis:names:tc:ebxml-
2612 regrep:AssociationType:DifferentFrom">
2613   <rim:Name>
2614     <rim:LocalizedString charset="UTF-8" value="DifferentFrom"/>
2615   </rim:Name>
2616 </rim:ClassificationNode>
2617
2618 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2619 regrep:classificationScheme:AssociationType"
2620 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:DatatypeProperty"
2621 code="DatatypeProperty" id="urn:oasis:names:tc:ebxml-
2622 regrep:AssociationType:DatatypeProperty">
2623   <rim:Name>
2624     <rim:LocalizedString charset="UTF-8"
2625 value="DatatypeProperty"/>
2626   </rim:Name>
2627 </rim:ClassificationNode>
2628
2629 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2630 regrep:classificationScheme:AssociationType"
2631 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:TransitiveProperty"
2632 code="TransitiveProperty" id="urn:oasis:names:tc:ebxml-
2633 regrep:AssociationType:TransitiveProperty">
2634   <rim:Name>
2635     <rim:LocalizedString charset="UTF-8"
2636 value="TransitiveProperty"/>
2637   </rim:Name>
2638 </rim:ClassificationNode>
2639

```

```

2640 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2641 regrep:classificationScheme:AssociationType"
2642 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:InverseOf"
2643 code="InverseOf" id="urn:oasis:names:tc:ebxml-
2644 regrep:AssociationType:InverseOf">
2645   <rim:Name>
2646     <rim:LocalizedString charset="UTF-8" value="InverseOf"/>
2647   </rim:Name>
2648 </rim:ClassificationNode>
2649
2650 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2651 regrep:classificationScheme:AssociationType"
2652 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:SymmetricProperty"
2653 code="SymmetricProperty" id="urn:oasis:names:tc:ebxml-
2654 regrep:AssociationType:SymmetricProperty">
2655   <rim:Name>
2656     <rim:LocalizedString charset="UTF-8"
2657 value="SymmetricProperty"/>
2658   </rim:Name>
2659 </rim:ClassificationNode>
2660
2661 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2662 regrep:classificationScheme:AssociationType"
2663 lid="urn:oasis:names:tc:ebxml-regrep:AssociationType:FunctionalProperty"
2664 code="FunctionalProperty" id="urn:oasis:names:tc:ebxml-
2665 regrep:AssociationType:FunctionalProperty">
2666   <rim:Name>
2667     <rim:LocalizedString charset="UTF-8"
2668 value="FunctionalProperty"/>
2669   </rim:Name>
2670 </rim:ClassificationNode>
2671
2672 <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
2673 regrep:classificationScheme:AssociationType"
2674 lid="urn:oasis:names:tc:ebxml-
2675 regrep:AssociationType:InverseFunctionalProperty"
2676 code="InverseFunctionalProperty" id="urn:oasis:names:tc:ebxml-
2677 regrep:AssociationType:InverseFunctionalProperty">
2678   <rim:Name>
2679     <rim:LocalizedString charset="UTF-8"
2680 value="InverseFunctionalProperty"/>
2681   </rim:Name>
2682 </rim:ClassificationNode>

```

### Extensions to the AssociationType ClassificationScheme

## 2684 7.3 Canonical Queries

2685 The following new canonical queries are described by this profile. Note that while these queries are  
2686 complex, the complexity is hidden from clients by exposing only the query parameters to them.

### 2687 7.3.1 All SuperProperties Discovery Query

2688 Since recursion is not supported by SQL-92, the stored procedure for this query is presented in this  
2689 section.

```

2690 CREATE PROCEDURE findAllSuperProperties
2691   @propertyName varchar(50)
2692 AS
2693 WITH
2694   Parents(superPropertyID) AS
2695   (
2696     SELECT A3.id
2697     FROM Association A1, Association A2, Association A3, Name_ N
2698     WHERE A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
2699 regrep:AssociationType:SubPropertyOf' AND
2700     A1.id = N.parent AND N.value LIKE @propertyName AND

```

```

2701         A2.sourceObject = A1.id AND A2.targetObject = A3.id
2702     UNION ALL
2703         SELECT A.targetObject
2704         FROM Association A JOIN Parents P
2705         ON P.superPropertyID = A.sourceObject
2706         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2707 regrep:AssociationType:SubPropertyOf'
2708     )
2709     SELECT * FROM Parents
2710 GO

```

2711 **Recursive stored procedure for MS SQL Server 2005 retrieving all super properties of a given**  
2712 **property (Association)**

2713 **7.3.2 Immediate SuperClass Discovery Query**

```

2714     <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2715 regrep:query:FindImmediateSuperClasses" id="urn:oasis:names:tc:ebxml-
2716 regrep:query:FindImmediateSuperClasses">
2717     <rim:Name>
2718         <rim:LocalizedString
2719 value="label.FindImmediateSuperClasses"/>
2720     </rim:Name>
2721     <rim:Description>
2722         <rim:LocalizedString
2723 value="label.FindImmediateSuperClasses.desc"/>
2724     </rim:Description>
2725     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2726 regrep:QueryLanguage:SQL-92">
2727         SELECT C2.*
2728         FROM ClassificationNode C2, Association A, Name_ N,
2729 ClassificationNode C1
2730         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2731 regrep:AssociationType:SubClassOf' AND
2732         C1.id = N.parent AND
2733         N.value LIKE '$className' AND
2734         A.sourceObject = C1.id AND
2735         A.targetObject = C2.id
2736     </rim:QueryExpression>
2737 </rim:AdhocQuery>
2738

```

2739 **The Adhoc Query retrieving immediate super classes of a given classification node**

2740 **7.3.3 Immediate SubClass Discovery Query**

```

2741 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2742 regrep:query:FindImmediateSubClasses" id="urn:oasis:names:tc:ebxml-
2743 regrep:query:FindImmediateSubClasses">
2744     <rim:Name>
2745         <rim:LocalizedString value="label.FindImmediateSubClasses"/>
2746     </rim:Name>
2747     <rim:Description>
2748         <rim:LocalizedString
2749 value="label.FindImmediateSubClasses.desc"/>
2750     </rim:Description>
2751     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2752 regrep:QueryLanguage:SQL-92">
2753         SELECT C2.*
2754         FROM ClassificationNode C2, Association A, Name_ N,
2755 ClassificationNode C1
2756         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2757 regrep:AssociationType:SubClassOf' AND
2758         C1.id = N.parent AND
2759         N.value LIKE '$className' AND
2760         A.sourceObject = C2.id AND
2761         A.targetObject = C1.id

```

2762 </rim:QueryExpression>  
2763 </rim:AdhocQuery>

2764 **The Adhoc Query retrieving immediate subclasses of a given classification node**

### 2765 7.3.4 All SuperClasses Discovery Query

2766 Since recursion is not supported by SQL-92, the stored procedure for this query is presented in this  
2767 section.

```
2768 CREATE PROCEDURE findAllSuperClasses  
2769 @className varchar(50)  
2770 AS  
2771 WITH  
2772 Parents(superClassID) AS  
2773 (  
2774 SELECT C2.id  
2775 FROM Association A, Name_ N, ClassificationNode C1,  
2776 ClassificationNode C2  
2777 WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-  
2778 regrep:AssociationType:SubClassOf' AND  
2779 C1.id = N.parent AND N.value LIKE @className AND  
2780 A.sourceObject = C1.id AND A.targetObject = C2.id  
2781 UNION ALL  
2782 SELECT A.targetObject  
2783 FROM Association A JOIN Parents P  
2784 ON P.superClassID = A.sourceObject  
2785 WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-  
2786 regrep:AssociationType:SubClassOf'  
2787 )  
2788 SELECT * FROM Parents  
2789 GO
```

2790 **Recursive stored procedure for MS SQL Server 2005 database retrieving all super classes of a**  
2791 **given classification node**

### 2792 7.3.5 All SubClasses Discovery Query

2793 Since recursion is not supported by SQL-92, the stored procedure for this query is presented in this  
2794 section.

```
2795 CREATE PROCEDURE findAllSubClasses  
2796 @className varchar(50)  
2797 AS  
2798 WITH  
2799 Children(subClassID) AS  
2800 (  
2801 SELECT C1.id  
2802 FROM Association A, Name_ N, ClassificationNode C1,  
2803 ClassificationNode C2  
2804 WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-  
2805 regrep:AssociationType:SubClassOf' AND  
2806 C2.id = N.parent AND N.value LIKE @className AND  
2807 A.sourceObject = C1.id AND A.targetObject = C2.id  
2808 UNION ALL  
2809 SELECT A.sourceObject  
2810 FROM Association A JOIN Children C  
2811 ON C.subClassID = A.targetObject  
2812 WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-  
2813 regrep:AssociationType:SubClassOf'  
2814 )  
2815 SELECT * FROM Children  
2816 GO
```

2817 **Recursive stored procedure for MS SQL Server 2005 database retrieving all subclasses of a**  
2818 **given classification node**

### 2819 7.3.6 EquivalentClasses Discovery Query

```
2820 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2821 regrep:query:FindEquivalentClasses" id="urn:oasis:names:tc:ebxml-
2822 regrep:query:FindEquivalentClasses">
2823   <rim:Name>
2824     <rim:LocalizedString value="label.FindEquivalentClasses"/>
2825   </rim:Name>
2826   <rim:Description>
2827     <rim:LocalizedString
2828 value="label.FindEquivalentClasses.desc"/>
2829   </rim:Description>
2830   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2831 regrep:QueryLanguage:SQL-92">
2832     SELECT C2.*
2833     FROM ClassificationNode C2, Association A, Name_ N,
2834 ClassificationNode C
2835     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2836 regrep:AssociationType:EquivalentTo' AND
2837     C.id = N.parent AND
2838     N.value LIKE '$className' AND
2839     A.sourceObject = C.id AND
2840     A.targetObject = C2.id
2841   </rim:QueryExpression>
2842 </rim:AdhocQuery>
```

2843 **Adhoc Query retrieving all the equivalent classes of a given classification node**

### 2844 7.3.7 EquivalentProperties Discovery Query

```
2845 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2846 regrep:query:FindEquivalentProperties" id="urn:oasis:names:tc:ebxml-
2847 regrep:query:FindEquivalentProperties">
2848   <rim:Name>
2849     <rim:LocalizedString
2850 value="label.FindEquivalentProperties"/>
2851   </rim:Name>
2852   <rim:Description>
2853     <rim:LocalizedString
2854 value="label.FindEquivalentProperties.desc"/>
2855   </rim:Description>
2856   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2857 regrep:QueryLanguage:SQL-92">
2858     SELECT A3.*
2859     FROM Association A3, Association A1, Name_ N, Association
2860 A2
2861     WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
2862 regrep:AssociationType:EquivalentTo' AND
2863     A2.id = N.parent AND
2864     N.value LIKE '$propertyName' AND
2865     A1.sourceObject = A2.id AND
2866     A1.targetObject = A3.id
2867   </rim:QueryExpression>
2868 </rim:AdhocQuery>
```

2869 **Adhoc Query retrieving all the equivalent Association Type of a given Association Type**

### 2870 7.3.8 SameExtrinsicObjects Discovery Query

```
2871 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2872 regrep:query:FindTheSameExtrinsicObjects" id="urn:oasis:names:tc:ebxml-
2873 regrep:query:FindTheSameExtrinsicObjects">
2874   <rim:Name>
2875     <rim:LocalizedString
2876 value="label.FindTheSameExtrinsicObjects"/>
2877   </rim:Name>
2878   <rim:Description>
```



```

2879         <rim:LocalizedString
2880 value="label.FindTheSameExtrinsicObjects.desc"/>
2881     </rim:Description>
2882     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:exxml-
2883 regrep:QueryLanguage:SQL-92">
2884         SELECT E2.*
2885         FROM ExtrinsicObject E2, Association A, Name_ N,
2886 ExtrinsicObject E
2887         WHERE A.associationType LIKE ''urn:oasis:names:tc:exxml-
2888 regrep:AssociationType:SameAs'' AND
2889         E.id = N.parent AND
2890         N.value LIKE ''$extrinsicObjectName'' AND
2891         A.sourceObject = E.id AND
2892         A.targetObject = E2.id
2893     </rim:QueryExpression>
2894 </rim:AdhocQuery>

```

2895 **Adhoc Query retrieving all the "ExtrinsicObjects" defined to be the same with a given**  
2896 **ExtrinsicObject**

### 2897 7.3.9 DifferentExtrinsicObjects Discovery Query

```

2898 <rim:AdhocQuery lid="urn:oasis:names:tc:exxml-
2899 regrep:query:FindDifferentExtrinsicObjects" id="urn:oasis:names:tc:exxml-
2900 regrep:query:FindDifferentExtrinsicObjects">
2901     <rim:Name>
2902         <rim:LocalizedString
2903 value="label.FindDifferentExtrinsicObjects"/>
2904     </rim:Name>
2905     <rim:Description>
2906         <rim:LocalizedString
2907 value="label.FindDifferentExtrinsicObjects.desc"/>
2908     </rim:Description>
2909     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:exxml-
2910 regrep:QueryLanguage:SQL-92">
2911         SELECT E2.*
2912         FROM ExtrinsicObject E2, Association A, Name_ N,
2913 ExtrinsicObject E
2914         WHERE A.associationType LIKE ''urn:oasis:names:tc:exxml-
2915 regrep:AssociationType:DifferentFrom'' AND
2916         E.id = N.parent AND
2917         N.value LIKE ''$extrinsicObjectName'' AND
2918         A.sourceObject = E.id AND
2919         A.targetObject = E2.id
2920     </rim:QueryExpression>
2921 </rim:AdhocQuery>

```

2922 **Adhoc Query retrieving all the "ExtrinsicObjects" defined to be different from a given**  
2923 **ExtrinsicObject**

### 2924 7.3.10 AllDifferentRegistryObject Discovery Query

```

2925 <rim:AdhocQuery lid="urn:oasis:names:tc:exxml-
2926 regrep:query:FindAllDifferent" id="urn:oasis:names:tc:exxml-
2927 regrep:query:FindAllDifferent">
2928     <rim:Name>
2929         <rim:LocalizedString value="label.FindAllDifferent"/>
2930     </rim:Name>
2931     <rim:Description>
2932         <rim:LocalizedString value="label.FindAllDifferent.desc"/>
2933     </rim:Description>
2934     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:exxml-
2935 regrep:QueryLanguage:SQL-92">
2936         SELECT RO2.*
2937         FROM RegistryObject RO2, Association A1, Association A2,
2938 Name_ N, RegistryObject RO,
2939         RegistryPackage RP<!--, Slot S-->

```

```

2940         WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
2941 regrep:AssociationType:HasMember' AND
2942         RO.id = N.parent AND
2943         N.value LIKE '$registryObjectName' AND
2944         A1.sourceObject = RP.id AND
2945         <!-- S.parent = RP.id AND
2946         S.name_ LIKE 'allDifferent' AND S.value LIKE 'true' AND
2947 -->
2948         A1.targetObject = RO.id AND
2949         A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
2950 regrep:AssociationType:HasMember' AND
2951         A2.sourceObject = RP.id AND
2952         A2.targetObject != RO.id AND
2953         A2.targetObject = RO2.id
2954     </rim:QueryExpression>
2955 </rim:AdhocQuery>

```

2956 **Adhoc Query retrieving all the "RegistryObjects" defined to be different from a given**  
2957 **RegistryObject through a "allDifferentFrom" construct**

### 2958 7.3.11 ObjectProperties Discovery Query

```

2959 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2960 regrep:query:FindObjectProperties" id="urn:oasis:names:tc:ebxml-
2961 regrep:query:FindObjectProperties">
2962     <rim:Name>
2963         <rim:LocalizedString value="label.FindObjectProperties"/>
2964     </rim:Name>
2965     <rim:Description>
2966         <rim:LocalizedString
2967 value="label.FindObjectProperties.desc"/>
2968     </rim:Description>
2969     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2970 regrep:QueryLanguage:SQL-92">
2971         SELECT A.*
2972         FROM Association A, Name_ N, ClassificationNode C
2973         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
2974 regrep:AssociationType:ObjectProperty' AND
2975         C.id = N.parent AND
2976         N.value LIKE '$className' AND
2977         A.sourceObject = C.id
2978     </rim:QueryExpression>
2979 </rim:AdhocQuery>

```

2980 **Adhoc Query retrieving all the object properties of a given classification node**

### 2981 7.3.12 ImmediateInheritedObjectProperties Discovery Query

```

2982 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
2983 regrep:query:FindImmediateInheritedObjectProperties"
2984 id="urn:oasis:names:tc:ebxml-
2985 regrep:query:FindImmediateInheritedObjectProperties">
2986     <rim:Name>
2987         <rim:LocalizedString
2988 value="label.FindImmediateInheritedObjectProperties"/>
2989     </rim:Name>
2990     <rim:Description>
2991         <rim:LocalizedString
2992 value="label.FindImmediateInheritedObjectProperties.desc"/>
2993     </rim:Description>
2994     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
2995 regrep:QueryLanguage:SQL-92">
2996         SELECT A2.*
2997         FROM Association A, Name_ N, ClassificationNode C1,
2998         ClassificationNode C2, Association A2
2999         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3000 regrep:AssociationType:SubClassOf' AND

```

```

3001         C1.id = N.parent AND
3002         N.value LIKE '$className' AND
3003         A.sourceObject = C1.id AND
3004         A.targetObject = C2.id AND
3005         A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3006 regrep:AssociationType:ObjectProperty' AND
3007         A2.sourceObject=C2.id
3008     </rim:QueryExpression>
3009 </rim:AdhocQuery>

```

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

### 3012 7.3.13 AllInheritedObjectProperties Discovery Query

3013 Since recursion is not supported by SQL-92, the stored procedure for this query is presented in this  
3014 section.

```

3015 CREATE PROCEDURE findAllInheritedObjectProperties
3016     @className varchar(50)
3017 AS
3018 WITH Parents(superClassID) AS (
3019     SELECT C2.id
3020     FROM Association A, Name_ N, ClassificationNode C1,
3021     ClassificationNode C2
3022     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3023 regrep:AssociationType:SubClassOf' AND
3024         C1.id = N.parent AND N.value LIKE @className AND
3025         A.sourceObject = C1.id AND A.targetObject = C2.id
3026 UNION ALL
3027     SELECT A.targetObject
3028     FROM Association A JOIN Parents P
3029     ON P.superClassID = A.sourceObject
3030     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3031 regrep:AssociationType:SubClassOf'
3032 ) SELECT A.id
3033     FROM Association A, Parents P
3034     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3035 regrep:AssociationType:ObjectProperty' AND
3036         A.sourceObject=P.superClassID
3037 UNION
3038
3039 SELECT A.id
3040 FROM Name_ N, ClassificationNode C, Association A
3041 WHERE C.id = N.parent AND N.value LIKE @className AND
3042         A.sourceObject=C.id AND A.associationType LIKE
3043 'urn:oasis:names:tc:ebxml-regrep:AssociationType:ObjectProperty'
3044 GO

```

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

### 3047 7.3.14 DatatypeProperties Discovery Query

```

3048 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3049 regrep:query:FindDatatypeProperties" id="urn:oasis:names:tc:ebxml-
3050 regrep:query:FindDatatypeProperties">
3051     <rim:Name>
3052         <rim:LocalizedString value="label.FindDatatypeProperties"/>
3053     </rim:Name>
3054     <rim:Description>
3055         <rim:LocalizedString
3056 value="label.FindDatatypeProperties.desc"/>
3057     </rim:Description>
3058     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3059 regrep:QueryLanguage:SQL-92">
3060         SELECT A.*

```

```

3061         FROM Association A, Name_ N, ClassificationNode C
3062         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3063 regrep:AssociationType:DatatypeProperty' AND
3064         C.id = N.parent AND
3065         N.value LIKE '$className' AND
3066         A.sourceObject = C.id
3067     </rim:QueryExpression>
3068 </rim:AdhocQuery>

```

3069 **Adhoc Query retrieving all the datatype properties of a given classification node**

### 3070 7.3.15 AllInheritedDatatypeProperties Discovery Query

3071 Since recursion is not supported by SQL-92, the stored procedure for this query is presented in this  
3072 section.

```

3073 CREATE PROCEDURE findAllInheritedDatatypeProperties
3074     @className varchar(50)
3075 AS
3076 WITH Parents(superClassID) AS (
3077     SELECT C2.id
3078     FROM Association A, Name_ N, ClassificationNode C1,
3079 ClassificationNode C2
3080     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3081 regrep:AssociationType:SubClassOf' AND
3082     C1.id = N.parent AND N.value LIKE @className AND
3083     A.sourceObject = C1.id AND A.targetObject = C2.id
3084 UNION ALL
3085     SELECT A.targetObject
3086     FROM Association A JOIN Parents P
3087     ON P.superClassID = A.sourceObject
3088     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3089 regrep:AssociationType:SubClassOf'
3090 ) SELECT A.id
3091     FROM Association A, Parents P
3092     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3093 regrep:AssociationType:DatatypeProperty' AND
3094     A.sourceObject=P.superClassID
3095 UNION
3096
3097 SELECT A.id
3098 FROM Name_ N, ClassificationNode C, Association A
3099 WHERE C.id = N.parent AND N.value LIKE @className AND
3100     A.sourceObject=C.id AND A.associationType LIKE
3101 'urn:oasis:names:tc:ebxml-regrep:AssociationType:DatatypeProperty'
3102 GO

```

3103 **Recursive stored procedure for MS SQL Server 2005 database retrieving all inherited Datatype**  
3104 **Properties of a given classification node**

### 3105 7.3.16 TransitiveRelationships Discovery Query

```

3106 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3107 regrep:query:FindTransitiveRelationships" id="urn:oasis:names:tc:ebxml-
3108 regrep:query:FindTransitiveRelationships">
3109     <rim:Name>
3110         <rim:LocalizedString
3111 value="label.FindTransitiveRelationships"/>
3112     </rim:Name>
3113     <rim:Description>
3114         <rim:LocalizedString
3115 value="label.FindTransitiveRelationships.desc"/>
3116     </rim:Description>
3117     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3118 regrep:QueryLanguage:SQL-92">
3119         SELECT C.*

```

```

3120         FROM ClassificationNode C, Association A1, Association A2,
3121 Name_ N1, Name_ N2, Name_ N3
3122         WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
3123 regrep:AssociationType:TransitiveProperty' AND
3124         A1.id = N1.parent AND
3125         N1.value LIKE '$propertyName' AND
3126         A1.sourceObject = N3.parent AND
3127         N3.value LIKE '$className' AND
3128         A2.sourceObject = A1.targetObject AND
3129         A2.id = N2.parent AND
3130         N2.value LIKE '$propertyName' AND
3131         A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3132 regrep:AssociationType:TransitiveProperty' AND
3133         A2.targetObject = C.id
3134         <!-- UNION
3135         SELECT C.*
3136         FROM ClassificationNode C, Association A1, Name_ N1, Name_
3137 N3
3138         WHERE A1.associationType LIKE 'urn:oasis:names:tc:ebxml-
3139 regrep:AssociationType:TransitiveProperty' AND
3140         A1.id = N1.parent AND
3141         N1.value LIKE '$propertyName' AND
3142         A1.sourceObject = N3.parent AND
3143         N3.value LIKE '$className' AND
3144         A1.targetObject = C.id -->
3145     </rim:QueryExpression>
3146 </rim:AdhocQuery>

```

3147 **Adhoc Query retrieving the objects in transitive relationship with a given object**

### 3148 7.3.17 TargetObjects Discovery Query

```

3149 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3150 regrep:query:FindTargetObjects" id="urn:oasis:names:tc:ebxml-
3151 regrep:query:FindTargetObjects">
3152     <rim:Name>
3153         <rim:LocalizedString value="label.FindTargetObjects"/>
3154     </rim:Name>
3155     <rim:Description>
3156         <rim:LocalizedString value="label.FindTargetObjects.desc"/>
3157     </rim:Description>
3158     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3159 regrep:QueryLanguage:SQL-92">
3160         SELECT C2.*
3161         FROM ClassificationNode C2, Association A, Name_ N, Name_
3162 N2, ClassificationNode C1
3163         WHERE A.id=N2.parent AND
3164         N2.value LIKE '$propertyName' AND
3165         C1.id = N.parent AND
3166         N.value LIKE '$className' AND
3167         A.sourceObject = C1.id AND
3168         A.targetObject = C2.id
3169     </rim:QueryExpression>
3170 </rim:AdhocQuery>

```

3171 **Adhoc Query retrieving the Target Objects from the Registry, given a Source Object and**  
3172 **an Association**

### 3173 7.3.18 TargetObjectsInverseOf Discovery Query

```

3174 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3175 regrep:query:FindTOinverseOf" id="urn:oasis:names:tc:ebxml-
3176 regrep:query:FindTOinverseOf">
3177     <rim:Name>
3178         <rim:LocalizedString value="label.FindTOinverseOf"/>
3179     </rim:Name>
3180     <rim:Description>

```

```

3181         <rim:LocalizedString value="label.FindTOinverseOf.desc"/>
3182     </rim:Description>
3183     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3184 regrep:QueryLanguage:SQL-92">
3185         SELECT C2.*
3186         FROM ClassificationNode C2, Association A1, Association A2,
3187 Association A3, Name_ N, Name_ N2, ClassificationNode C1
3188         WHERE A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3189 regrep:AssociationType:InverseOf' AND
3190         A1.id = N.parent AND
3191         N.value LIKE '$propertyName' AND
3192         A2.sourceObject = A1.id AND
3193         A2.targetObject = A3.id AND
3194         C1.id = N2.parent AND
3195         N2.value LIKE '$className' AND
3196         A3.targetObject = C1.id AND
3197         A3.sourceObject = C2.id
3198     </rim:QueryExpression>
3199 </rim:AdhocQuery>

```

3200 **Adhoc query retrieving the Source Objects of an Association which is in "inverseOf"**  
3201 **relationship to this Association**

### 3202 7.3.19 InverseRanges Discovery Query

```

3203 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3204 regrep:query:FindInverseRanges" id="urn:oasis:names:tc:ebxml-
3205 regrep:query:FindInverseRanges">
3206     <rim:Name>
3207         <rim:LocalizedString value="label.FindInverseRanges"/>
3208     </rim:Name>
3209     <rim:Description>
3210         <rim:LocalizedString value="label.FindInverseRanges.desc"/>
3211     </rim:Description>
3212     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3213 regrep:QueryLanguage:SQL-92">
3214         <!-- SELECT C2.*
3215         FROM Association A, Name_ N, Name_ N2, ClassificationNode
3216 C1, ClassificationNode C2
3217         WHERE A.id=N2.parent AND
3218         N2.value LIKE '$propertyName' AND
3219         C1.id = N.parent AND
3220         N.value LIKE '$className' AND
3221         A.sourceObject = C1.id AND
3222         A.targetObject = C2.id
3223         UNION -->
3224         SELECT C2.*
3225         FROM ClassificationNode C2, Association A1, Association A2,
3226 Association A3, Name_ N, NAME_ N2, ClassificationNode C1
3227         WHERE A2.associationType LIKE 'urn:oasis:names:tc:ebxml-
3228 regrep:AssociationType:InverseOf' AND
3229         A1.id = N.parent AND
3230         N.value LIKE '$propertyName' AND
3231         A2.sourceObject = A1.id AND
3232         A2.targetObject = A3.id AND
3233         C1.id = N2.parent AND
3234         N2.value LIKE '$className' AND
3235         A1.sourceObject = C1.id AND
3236         A3.sourceObject = C2.id
3237     </rim:QueryExpression>
3238 </rim:AdhocQuery>

```

3239 **Adhoc Query Retrieving both the Target Objects of a given Association and the Source**  
3240 **Objects of an Association which is in "inverseOf" relationship to this Association**

### 3241 7.3.20 SymmetricProperties Discovery Query

```
3242 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3243 regrep:query:FindSymmetricProperties" id="urn:oasis:names:tc:ebxml-
3244 regrep:query:FindSymmetricProperties">
3245   <rim:Name>
3246     <rim:LocalizedString value="label.FindSymmetricProperties"/>
3247   </rim:Name>
3248   <rim:Description>
3249     <rim:LocalizedString
3250 value="label.FindSymmetricProperties.desc"/>
3251   </rim:Description>
3252   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3253 regrep:QueryLanguage:SQL-92">
3254     SELECT A.*
3255     FROM Association A, Name_N, ClassificationNode C
3256     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3257 regrep:AssociationType:SymmetricProperty' AND
3258     C.id = N.parent AND
3259     N.value LIKE '$className' AND
3260     A.sourceObject = C.id
3261   </rim:QueryExpression>
3262 </rim:AdhocQuery>
```

3263 **Adhoc Query retrieving all the Symmetric properties of a given classification node**

### 3264 7.3.21 FunctionalProperties Discovery Query

```
3265 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3266 regrep:query:FindFunctionalProperties" id="urn:oasis:names:tc:ebxml-
3267 regrep:query:FindFunctionalProperties">
3268   <rim:Name>
3269     <rim:LocalizedString
3270 value="label.FindFunctionalProperties"/>
3271   </rim:Name>
3272   <rim:Description>
3273     <rim:LocalizedString
3274 value="label.FindFunctionalProperties.desc"/>
3275   </rim:Description>
3276   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3277 regrep:QueryLanguage:SQL-92">
3278     SELECT A.*
3279     FROM Association A, Name_N, ClassificationNode C
3280     WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3281 regrep:AssociationType:FunctionalProperty' AND
3282     C.id = N.parent AND
3283     N.value LIKE '$className' AND
3284     A.sourceObject = C.id
3285   </rim:QueryExpression>
3286 </rim:AdhocQuery>
```

3287 **Adhoc Query retrieving all the Functional properties of a given classification node**

### 3288 7.3.22 InverseFunctionalProperties Discovery Query

```
3289 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
3290 regrep:query:FindInverseFunctionalProperties"
3291 id="urn:oasis:names:tc:ebxml-
3292 regrep:query:FindInverseFunctionalProperties">
3293   <rim:Name>
3294     <rim:LocalizedString
3295 value="label.FindInverseFunctionalProperties"/>
3296   </rim:Name>
3297   <rim:Description>
3298     <rim:LocalizedString
3299 value="label.FindInverseFunctionalProperties.desc"/>
3300   </rim:Description>
```

```

3301     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3302 regrep:QueryLanguage:SQL-92">
3303         SELECT A.*
3304         FROM Association A, Name_N, ClassificationNode C
3305         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3306 regrep:AssociationType:InverseFunctionalProperty' AND
3307         C.id = N.parent AND
3308         N.value LIKE '$className' AND
3309         A.sourceObject = C.id
3310     </rim:QueryExpression>
3311 </rim:AdhocQuery>

```

3312 **Adhoc Query retrieving all the Inverse Functional properties of a given classification node**

### 3313 7.3.23 Instances Discovery Query Discovery Query

```

3314 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-regrep:query:FindInstances"
3315 id="urn:oasis:names:tc:ebxml-regrep:query:FindInstances">
3316     <rim:Name>
3317         <rim:LocalizedString value="label.FindInstances"/>
3318     </rim:Name>
3319     <rim:Description>
3320         <rim:LocalizedString value="label.FindInstances.desc"/>
3321     </rim:Description>
3322     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
3323 regrep:QueryLanguage:SQL-92">
3324         <!-- SELECT S.* FROM Service S, (
3325         SELECT A.targetObject AS id
3326         FROM RegistryPackage R, Association A
3327         WHERE R.id=A.sourceObject AND
3328         A.associationType = 'urn:oasis:names:tc:ebxml-
3329 regrep:AssociationType:HasMember' AND
3330         R.id IN (
3331         SELECT A.targetObject
3332         FROM Association A, Name_N, ClassificationNode C
3333         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3334 regrep:AssociationType:IntersectionOf' AND
3335         C.id = N.parent AND
3336         N.value LIKE '$className' AND
3337         A.sourceObject = C.id
3338         )
3339         ) AS T1, (
3340         SELECT A.targetObject AS id
3341         FROM RegistryPackage R, Association A
3342         WHERE R.id=A.sourceObject AND
3343         A.associationType = 'urn:oasis:names:tc:ebxml-
3344 regrep:AssociationType:HasMember' AND
3345         R.id IN (
3346         SELECT A.targetObject
3347         FROM Association A, Name_N, ClassificationNode C
3348         WHERE A.associationType LIKE 'urn:oasis:names:tc:ebxml-
3349 regrep:AssociationType:IntersectionOf' AND
3350         C.id = N.parent AND
3351         N.value LIKE '$className' AND
3352         A.sourceObject = C.id
3353         )
3354         ) AS T2
3355         WHERE S.id IN (
3356         SELECT classifiedObject
3357         FROM Classification
3358         WHERE classificationNode=T1.id
3359         INTERSECT
3360         SELECT classifiedObject
3361         FROM Classification
3362         WHERE classificationNode=T2.id
3363         ) AND T1.id!=T2.id
3364         UNION -->

```



```
3365         SELECT S.*
3366         FROM Service S, Classification C, ClassificationNode CN,
3367 Name_ N
3368         WHERE S.id = C. classifiedObject AND
3369         C.classificationNode = CN.id AND
3370         N.value LIKE '$className' AND
3371         N.parent = CN.id
3372     </rim:QueryExpression>
3373 </rim:AdhocQuery>
```

3374 **Adhoc Query Retrieving the instances of intersected classes**

3375

## 8 OWL Profile References

3376

### 8.1 Normative References

3377

[Bechhofer, Harmelen, Hendler, Horrocks, McGuinness, Patel-Schneider, Stein]

3378

Bechhofer, S., Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., Stein, L. A., OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004

3380

<http://www.w3.org/TR/2004/REC-owl-ref-20040210/>

3381

3382

[Brickley, Guha] Brickley, D., Guha, R.V., RDF Vocabulary Description Language 1.0: RDF Schema

3383

W3C Recommendation 10 February 2004

3384

<http://www.w3.org/TR/rdf-schema/>

3385

3386

[DAML+OIL] <http://www.daml.org/>

3387

[ebRIM] ebXML Registry Information Model version 3.0

3388

<http://docs.oasis-open.org/regrep/regrep-rim/v3.0/regrep-rim-3.0-os.pdf>

3389

3390

[ebRS] ebXML Registry Services Specification version 3.0

3391

<http://docs.oasis-open.org/regrep/regrep-rs/v3.0/regrep-rs-3.0-os.pdf>

3392

[ebRR-DPT] Deployment Profile Template For ebXML Registry 3.0 OASIS Specifications V\_0.1.2

3393

[ebMS-DPT] Deployment Profile Template For OASIS Specification ebXML Message Service 2.0

3394

[McGuinness, Harmelen] McGuinness, D. L., Harmelen, F., OWL Web Ontology Language Overview,

3395

W3C Recommendation 10 February 2004, <http://www.w3.org/TR/owl-features/>

3396

[OWL] Web Ontology Language (OWL), <http://www.w3.org/2004/OWL/>

3397

[RDF] Resource Description Framework, <http://www.w3.org/RDF/>

3398

[Smith, Welty, McGuinness] Smith, M. K., Welty, C., McGuinness, D. L.,

3399

OWL Web Ontology Language Guide, W3C Recommendation 10 February 2004,

3400

<http://www.w3.org/TR/owl-guide/>

3401

[SQL 92] SQL ISO/IEC 9075:1992 Information technology - Database languages - SQL.

3402

[SQL 99] ISO/IEC 9075:1999(E) Information technology - Database languages – SQL.

3403

[UML] Unified Modeling Language version 1.5

3404

<http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.pdf>

3405

[WSDL] WSDL Specification

3406

<http://www.w3.org/TR/wsdl>

3407

### 8.2 Informative References

3408

[Dogac, et. al.] Dogac A., Kabak Y., Laleci G. C. Mattocks, F. Najmi, J. Pollock

- 3409 Enhancing ebXML Registries to Make them OWL Aware  
3410 Distributed and Parallel Databases Journal, Springer-Verlag, Vol. 18, No. 1, July 2005, pp. 9-36.  
3411  
3412 [IMPL] ebXML Registry 3.0 Implementations  
3413 freebXML Registry: A royalty free, open source ebXML Registry Implementation  
3414 <http://ebxmlrr.sourceforge.net>  
3415  
3416 [LeeHendler]  
3417 Berners-Lee, T., Hendler, J., Lassila, O., "The Semantic Web", Scientific American, May 2001.  
3418  
3419 [StaabStuder] Staab, S., Studer, R., Handbook on Ontologies, Springer, 2004.