



Creating A Single Global Electronic Market

1
2
3
4

5 **OASIS/ebXML Registry Information Model v1.10**
6 **DRAFT**

7 **OASIS/ebXML Registry ~~Project Team~~ Technical**
8 **Committee**

9 ~~Working Draft 3/19/2001~~ 30 August 2001 ~~27 June 2001~~
10 **18 May 2001**

11 ~~This version: Version 0.61~~

12
13 **1 Status of this Document**

14
15 ~~This document specifies an ebXML DRAFT DRAFT STANDARD for the~~
16 ~~eBusiness community.~~

17
18 Distribution of this document is unlimited.

19
20 ~~The document formatting is based on the Internet Society's Standard RFC~~
21 ~~format.~~

22
23 ***This version:***

24
25 http://www.ebxml.org/specdrafts/project_teams/registry/private/RegistryInfoModel
26 [v1.00-61.pdf](http://www.ebxml.org/specdrafts/project_teams/registry/private/RegistryInfoModel)

27 <http://www.oasis-open.org/committees/regrep/documents/rimV1->
28 [10.pdf](http://www.ebxml.org/specs/ebRIM.pdf) [doc//www.ebxml.org/specs/ebRIM.pdf](http://www.ebxml.org/specs/ebRIM.pdf)

29
30 ***Latest version:***

31 <http://www.oasis-open.org/committees/regrep/documents/rimV1->
32 [10.pdf](http://www.ebxml.org/specs/ebRIM.pdf) [doc//www.ebxml.org/specs/ebRIM.pdf](http://www.ebxml.org/specs/ebRIM.pdf)

34
35
36
37
38
39
40
41
42
43
44

http://www.ebxml.org.org//pspecdrafts/project_teams/registry/private/RegistryInfoModelv1.0.pdf

Previous version:

http://www.ebxml.org/project_teams/registry/private/RegistryInfoModelv0.9060.pdf

44 **2 OASIS/ebXML Registry Technical** 45 **Committeeparticipants**

46 ~~We would like to recognize the following for their significant participation to the~~
47 ~~development of this document. The authors wish to acknowledge the support of~~
48 ~~the members of the Registry Project Team who contributed ideas to this~~
49 ~~specification by the group's discussion e-mail list, on conference calls and during~~
50 ~~face-to-face meetings. This document, in its current form, is a draft working~~
51 ~~document of the OASIS ebXML Registry Technical Committee. It build upon~~
52 ~~version 1.0 which has beenwas approved by the OASIS/ebXML Registry~~
53 ~~Technical Committee as DRAFT Specification of the TC. At the time of thatis~~
54 ~~approval the following were members of the OASIS/ebXML Registry Technical~~
55 ~~Committee.~~

56
57 Nagwa Abdelghfour, Sun Microsystems
58 Nicholas Berry, Boeing
59 Kathryn Breining, Boeing
60 Lisa Carnahan, US NIST (TC Chair)
61 Dan Chang, IBM
62 Joseph M. Chiusano, LMI
63 Joe Dalman, Tie Commerce
64 Suresh Damodaran, Sterling Commerce
65 Vadim Draluk, BEA
66 John Evdemon, Vitria Technologies
67 Anne Fischer, Drummond Group
68 Sally Fuger, AIAG
69 Len Gallagher, NIST
70 Michael Joya, XMLGlobal
71 Una Kearns, Documentum
72 Kyu-Chul Lee, Chungnam National University
73 Megan MacMillan, Gartner Solista
74 Norbert Mikula, DataChannel
75 Joel Munter, Intel
76 Farrukh Najmi, Sun Microsystems
77 Joel Neu, Vitria Technologies
78 Sanjay Patil, IONA
79 Neal Smith, Chevron
80 Nikola Stojanovic, Encoda Systems Inc.
81 David Webber, XMLGlobal
82 Prasad Yendluri, webmethods
83 Yutaka Yoshida, Sun Microsystems
84 Lisa Carnahan, NIST
85 Joe Dalman, Tie

OASIS/ebXML Registry Information Model

Page 3

- 86 ~~Philippe DeSmedt₁ - Viquity~~
- 87 ~~Sally Fuger₁ - AIAG~~
- 88 ~~Len Gallagher₁ - NIST~~
- 89 ~~Steve Hanna₁ - Sun Microsystems~~
- 90 ~~Scott Hinkelman₁ - IBM~~
- 91 ~~Michael Kass, NIST~~
- 92 ~~Jong-L Kim₁ - Innodigital~~
- 93 ~~Kyu-Chul Lee, Chungnam National University~~
- 94 ~~Sangwon Lim, Korea Institute for Electronic Commerce~~
- 95 ~~Bob Miller₁ - GXS~~
- 96 ~~Kunio Mizoguchi₁ - Electronic Commerce Promotion Council of Japan~~
- 97 ~~Dale Moberg₁ - Sterling Commerce~~
- 98 ~~Ron Monzillo₁ - Sun Microsystems~~
- 99 ~~JP Morgenthal₁ - eThink Systems, Inc.~~
- 100 ~~Joel Munter₁ - Intel~~
- 101 ~~Farrukh Najmi₁ - Sun Microsystems~~
- 102 ~~Scott Nieman₁ - Norstan Consulting~~
- 103 ~~Frank Olken₁ - Lawrence Berkeley National Laboratory~~
- 104 ~~Michael Park₁ - eSum Technologies~~
- 105 ~~Bruce Peat₁ - eProcess Solutions~~
- 106 ~~Mike Rowley₁ - Excelon Corporation~~
- 107 ~~Waqar Sadiq₁ - Vitria~~
- 108 ~~Krishna Sankar₁ - Cisco Systems Inc.~~
- 109 ~~Kim Tae Soo₁ - Government of Korea~~
- 110 ~~Nikola Stojanovic₁ - Encoda Systems, Inc.~~
- 111 ~~David Webber₁ - XML Global~~
- 112 ~~Yutaka Yoshida₁ - Sun Microsystems~~
- 113 ~~Prasad Yendluri₁ - webmethods~~
- 114 ~~Peter Z. Zhoo₁ - Knowledge For the new Millennium~~
- 115
- 116

116	Table of Contents	
117		
118	<u>1 STATUS OF THIS DOCUMENT.....1</u>	
119	<u>2 OASIS/EBXML REGISTRY TECHNICAL COMMITTEE.....3</u>	
120	<u>INTRODUCTION14</u>	
121	2.1 SUMMARY OF CONTENTS OF DOCUMENT.....14	
122	2.2 GENERAL CONVENTIONS14	
123	2.2.1 Naming Conventions.....15	
124	2.3 AUDIENCE.....15	
125	2.4 RELATED DOCUMENTS15	
126	<u>3 DESIGN OBJECTIVES16</u>	
127	3.1 GOALS16	
128	<u>4 SYSTEM OVERVIEW16</u>	
129	4.1 ROLE OF EBXML REGISTRY16	
130	4.2 REGISTRY SERVICES17	
131	4.3 WHAT THE REGISTRY INFORMATION MODEL DOES17	
132	4.4 HOW THE REGISTRY INFORMATION MODEL WORKS17	
133	4.5 WHERE THE REGISTRY INFORMATION MODEL MAY BE IMPLEMENTED17	
134	4.6 CONFORMANCE TO AN EBXML REGISTRY.....17	
135	<u>5 REGISTRY INFORMATION MODEL: HIGH LEVEL PUBLIC VIEW.....18</u>	
136	5.1 REGISTRYOBJECT21	
137	5.2 SLOT.....22	
138	5.3 ASSOCIATION.....22	
139	5.4 EXTERNALIDENTIFIER22	
140	5.5 EXTERNALLINK22	
141	5.6 CLASSIFICATIONSCHEME.....22	
142	5.7 CLASSIFICATIONNODE.....23	
143	5.8 CLASSIFICATION23	
144	5.9 PACKAGE.....23	
145	5.10 AUDITABLEEVENT.....23	
146	5.11 USER.....23	
147	5.12 POSTALADDRESS23	
148	5.13 ORGANIZATION.....23	
149	<u>6 REGISTRY INFORMATION MODEL: DETAIL VIEW.....24</u>	
150	6.1 ATTRIBUTE AND METHODS OF INFORMATION MODEL CLASSES27	
151	6.2 DATA TYPES28	

152	6.3 CLASS REGISTRYOBJECT	28
153	6.3.1 Attribute Summary.....	29
154	6.3.2 Attribute accesControlPolicy.....	29
155	6.3.3 Attribute description.....	29
156	6.3.4 Attribute id.....	29
157	6.3.5 Attribute name.....	30
158	6.3.6 Attribute objectType.....	30
159	6.3.7 Method Summary.....	32
160	6.4 CLASS REGISTRYENTRY	34
161	6.4.1 Attribute Summary.....	35
162	6.4.2 Attribute expiration.....	35
163	6.4.3 Attribute majorVersion.....	35
164	6.4.4 Attribute minorVersion.....	35
165	6.4.5 Attribute stability.....	36
166	6.4.6 Attribute status.....	36
167	6.4.7 Attribute userVersion.....	37
168	6.5 CLASS SLOT.....	39
169	6.5.1 A.....	39
170	6.5.2 Attribute name.....	40
171	6.5.3 Attribute slotType.....	40
172	6.5.4 Attribute values.....	40
173	6.6 CLASS EXTRINSICOBJECT	40
174	6.6.1 Attribute Summary.....	40
175	6.6.2 Attribute contentURI.....	41
176	6.6.3 Attribute isOpaque.....	41
177	6.6.4 Attribute mimeType.....	41
178	6.7 CLASS PACKAGE.....	42
179	6.7.1 Attribute Summary.....	42
180	6.7.2 Method Summary.....	42
181	6.8 CLASS EXTERNALIDENTIFIER	42
182	6.8.1 Attribute Summary.....	42
183	6.8.2 Attribute registryObject.....	43
184	6.8.3 Attribute value.....	43
185	6.8.4 Inherited Attribute id.....	43
186	6.8.5 Inherited Attribute name.....	43
187	6.9 CLASS EXTERNALLINK	43
188	6.9.1 Attribute Summary.....	44
189	6.9.2 Attribute externalURI.....	44
190	6.9.3 Method Summary.....	44
191	7 REGISTRY AUDIT TRAIL.....	44
192	7.1 CLASS AUDITABLEEVENT.....	45
193	7.1.1 Attribute Summary.....	45
194	7.1.2 Attribute eventType.....	45

195	7.1.3 Attribute RegistryObject	46
196	7.1.4 Attribute timestamp.....	46
197	7.1.5 Attribute user.....	46
198	7.2 CLASS USER	46
199	7.2.1 Attribute Summary.....	46
200	7.2.2 Attribute address.....	47
201	7.2.3 Attribute email.....	47
202	7.2.4 Attribute organization.....	47
203	7.2.5 Attribute personName.....	47
204	7.2.6 Attribute telephoneNumbers.....	47
205	7.2.7 Attribute url	47
206	7.3 CLASS ORGANIZATION	47
207	7.3.1 Attribute Summary.....	48
208	7.3.2 Attribute address.....	48
209	7.3.3 Attribute parent.....	48
210	7.3.4 Attribute primaryContact.....	48
211	7.3.5 Attribute telephoneNumbers.....	48
212	7.4 CLASS POSTALADDRESS.....	49
213	7.4.1 Attribute Summary.....	49
214	7.4.2 Attribute city.....	49
215	7.4.3 Attribute country.....	49
216	7.4.4 Attribute postalCode.....	49
217	7.4.5 Attribute state.....	49
218	7.4.6 Attribute street	49
219	7.4.7 Method Summary.....	50
220	7.5 CLASS TELEPHONENUMBER	50
221	7.5.1 Attribute Summary.....	50
222	7.5.2 Attribute areaCode.....	50
223	7.5.3 Attribute countryCode	50
224	7.5.4 Attribute extension.....	51
225	7.5.5 Attribute number.....	51
226	7.5.6 Attribute phoneType.....	51
227	7.5.7 Attribute url	51
228	7.6 CLASS PERSONNAME.....	51
229	7.6.1 Attribute Summary.....	51
230	7.6.2 Attribute firstName.....	51
231	7.6.3 Attribute lastName.....	51
232	7.6.4 Attribute middleName.....	52
233	8 REGISTRYOBJECT NAMING	52
234	9 ASSOCIATION OF REGISTRYENTRY.....	52
235	9.1 CLASS ASSOCIATION.....	53
236	9.1.1 Attribute Summary.....	54

237 9.1.2 Attribute associationType54

238 9.1.3 Attribute sourceObject.....55

239 9.1.4 Attribute targetObject.....55

240 9.1.5 Inherited Attribute id55

241 **10 CLASSIFICATION OF REGISTRYOBJECT.....57**

242 10.1 CLASS CLASSIFICATIONSCHEME.....61

243 10.2 CLASS CLASSIFICATIONNODE.....61

244 10.2.1 Attribute Summary.....62

245 10.2.2 Attribute parent.....62

246 10.2.3 Attribute code.....62

247 10.2.4 Method Summary.....62

248 10.3 CLASS CLASSIFICATION.....63

249 10.3.1 Attribute Summary.....63

250 10.3.2 Attribute classificationNode63

251 10.3.3 Attribute classifiedObject64

252 10.3.4 Inherited Attribute id64

253 10.3.5 Context Sensitive Classification64

254 10.4 EXAMPLE OF CLASSIFICATION SCHEMES.....65

255 10.5 STANDARDIZED TAXONOMY SUPPORT66

256 10.5.1 Full-featured Taxonomy Based Classification66

257 10.5.2 Light Weight Taxonomy Based Classification67

258 **11 INFORMATION MODEL: SECURITY VIEW67**

259 11.1 CLASS ACCESSCONTROLPOLICY.....69

260 11.2 CLASS PERMISSION69

261 11.3 CLASS PRIVILEGE69

262 11.4 CLASS PRIVILEGEATTRIBUTE.....70

263 11.5 CLASS ROLE70

264 11.6 CLASS GROUP.....70

265 11.7 CLASS IDENTITY71

266 11.8 CLASS PRINCIPAL71

267 **12 REFERENCES72**

268 **13 DISCLAIMER73**

269 **14 CONTACT INFORMATION.....74**

270 **COPYRIGHT STATEMENT.....75**

271 **1—STATUS OF THIS DOCUMENT.....1**

272 **2—EBXML PARTICIPANTS.....2**

273 **3—INTRODUCTION6**

274	3.1	SUMMARY OF CONTENTS OF DOCUMENT.....	6
275	3.2	GENERAL CONVENTIONS	6
276	3.2.1	<i>Naming Conventions</i>	7
277	3.3	AUDIENCE.....	7
278	3.4	RELATED DOCUMENTS	7
279	4	DESIGN OBJECTIVES	8
280	4.1	GOALS	8
281	5	SYSTEM OVERVIEW	8
282	5.1	ROLE OF EBXML <i>REGISTRY</i>	8
283	5.2	<i>REGISTRY SERVICES</i>	8
284	5.3	WHAT THE REGISTRY INFORMATION MODEL DOES	8
285	5.4	HOW THE REGISTRY INFORMATION MODEL WORKS	9
286	5.5	WHERE THE REGISTRY INFORMATION MODEL MAY BE IMPLEMENTED	9
287	5.6	<i>CONFORMANCE AS AN EBXML REGISTRY</i>	9
288	6	REGISTRY INFORMATION MODEL: HIGH LEVEL PUBLIC VIEW.....	9
289	6.1	REGISTRYENTRY	10
290	6.2	SLOT.....	11
291	6.3	ASSOCIATION.....	11
292	6.4	EXTERNALIDENTIFIER	11
293	6.5	EXTERNALLINK	11
294	6.6	CLASSIFICATIONNODE.....	11
295	6.7	CLASSIFICATION	11
296	6.8	PACKAGE.....	11
297	6.9	AUDITABLEEVENT.....	12
298	6.10	USER.....	12
299	6.11	POSTALADDRESS	12
300	6.12	ORGANIZATION.....	12
301	7	REGISTRY INFORMATION MODEL: DETAIL VIEW.....	12
302	7.1	INTERFACE REGISTRYOBJECT	13
303	7.2	INTERFACE VERSIONABLE	15
304	7.3	INTERFACE REGISTRYENTRY.....	15
305	7.3.1	<i>Pre-defined RegistryEntry Status Types</i>	17
306	7.3.2	<i>Pre-defined Object Types</i>	18
307	7.3.3	<i>Pre-defined RegistryEntry Stability Enumerations</i>	19
308	7.4	INTERFACE SLOT.....	19
309	7.5	INTERFACE EXTRINSICOBJECT.....	20
310	7.6	INTERFACE INTRINSICOBJECT.....	21
311	7.7	INTERFACE PACKAGE	22
312	7.8	INTERFACE EXTERNALIDENTIFIER.....	22
313	7.9	INTERFACE EXTERNALLINK.....	22

314	<u>8</u>	<u>REGISTRY AUDIT TRAIL</u>	<u>23</u>
315	8.1	INTERFACE AUDITABLEEVENT	23
316	8.1.1	<i>Pre-defined Auditable Event Types</i>	24
317	8.2	INTERFACE USER	25
318	8.3	INTERFACE ORGANIZATION	26
319	8.4	CLASS POSTALADDRESS	26
320	8.5	CLASS TELEPHONENUMBER	27
321	8.6	CLASS PERSONNAME	27
322	<u>9</u>	<u>REGISTRYENTRY NAMING</u>	<u>28</u>
323	<u>10</u>	<u>ASSOCIATION OF REGISTRYENTRY</u>	<u>28</u>
324	10.1	INTERFACE ASSOCIATION	28
325	10.1.1	<i>Pre-defined Association Types</i>	29
326	<u>11</u>	<u>CLASSIFICATION OF REGISTRYENTRY</u>	<u>31</u>
327	11.1	INTERFACE CLASSIFICATIONNODE	33
328	11.2	INTERFACE CLASSIFICATION	34
329	11.2.1	<i>Context Sensitive Classification</i>	35
330	11.3	EXAMPLE OF CLASSIFICATION SCHEMES	36
331	11.4	STANDARDIZED TAXONOMY SUPPORT	36
332	11.4.1	<i>Full featured Taxonomy Based Classification</i>	36
333	11.4.2	<i>Light Weight Taxonomy Based Classification</i>	37
334	<u>12</u>	<u>INFORMATION MODEL: SECURITY VIEW</u>	<u>37</u>
335	12.1	INTERFACE ACCESSCONTROLPOLICY	38
336	12.2	INTERFACE PERMISSION	39
337	12.3	INTERFACE PRIVILEGE	39
338	12.4	INTERFACE PRIVILEGEATTRIBUTE	40
339	12.5	INTERFACE ROLE	40
340	12.6	INTERFACE GROUP	40
341	12.7	INTERFACE IDENTITY	41
342	12.8	INTERFACE PRINCIPAL	41
343	<u>13</u>	<u>REFERENCES</u>	<u>42</u>
344	<u>14</u>	<u>DISCLAIMER</u>	<u>42</u>
345	<u>15</u>	<u>CONTACT INFORMATION</u>	<u>43</u>
346		<u>COPYRIGHT STATEMENT</u>	<u>44</u>
347	<u>1</u>	<u>STATUS OF THIS DOCUMENT</u>	<u>1</u>
348	<u>2</u>	<u>EBXML PARTICIPANTS</u>	<u>2</u>

349	<u>3</u>	<u>INTRODUCTION</u>	<u>6</u>
350	3.1	SUMMARY OF CONTENTS OF DOCUMENT	6
351	3.2	GENERAL CONVENTIONS	6
352	3.3	AUDIENCE	6
353	3.4	RELATED DOCUMENTS	7
354	<u>4</u>	<u>DESIGN OBJECTIVES</u>	<u>7</u>
355	4.1	GOALS	7
356	4.2	CAVEATS AND ASSUMPTIONS	7
357	<u>5</u>	<u>SYSTEM OVERVIEW</u>	<u>7</u>
358	5.1	ROLE OF EBXML REGISTRY	7
359	5.2	REGISTRY SERVICES	8
360	5.3	WHAT THE REGISTRY INFORMATION MODEL DOES	8
361	5.4	HOW THE REGISTRY INFORMATION MODEL WORKS	8
362	5.5	WHERE THE REGISTRY INFORMATION MODEL MAY BE IMPLEMENTED	8
363	5.6	CONFORMANCE AS AN EBXML REGISTRY	8
364	<u>6</u>	<u>REGISTRY INFORMATION MODEL: PUBLIC VIEW</u>	<u>9</u>
365	6.1	REGISTRYENTRY	10
366	6.2	SLOT	10
367	6.3	ASSOCIATION	10
368	6.4	EXTERNALIDENTIFIER	10
369	6.5	EXTERNALLINK	10
370	6.6	CLASSIFICATIONNODE	10
371	6.7	CLASSIFICATION	11
372	6.8	PACKAGE	11
373	6.9	AUDITABLEEVENT	11
374	6.10	USER	11
375	6.11	POSTALADDRESS	11
376	6.12	ORGANIZATION	11
377	<u>7</u>	<u>REGISTRY INFORMATION MODEL: DETAIL VIEW</u>	<u>11</u>
378	7.1	INTERFACE OBJECT	12
379	7.2	INTERFACE VERSIONABLE	14
380	7.3	INTERFACE REGISTRYENTRY	14
381	7.3.1	Pre-defined RegistryEntry Status Types	16
382	7.3.2	Pre-Defined Object Types	17
383	7.3.3	Pre-defined RegistryEntry Stability Enumerations	18
384	7.4	INTERFACE SLOT	18
385	7.5	INTERFACE EXTRINSIC OBJECT	19
386	7.6	INTERFACE INTRINSIC OBJECT	20
387	7.7	INTERFACE PACKAGE	20
388	7.8	INTERFACE EXTERNALIDENTIFIER	21

389	7.9	INTERFACE <i>EXTERNALLINK</i>	21
390	8	REGISTRY AUDIT TRAIL	22
391	8.1	INTERFACE <i>AUDITABLEEVENT</i>	22
392	8.1.1	Pred defined Auditable Event Types	23
393	8.2	INTERFACE <i>USER</i>	23
394	8.3	INTERFACE <i>ORGANIZATION</i>	24
395	8.4	CLASS <i>POSTALADDRESS</i>	25
396	8.5	CLASS <i>TELEPHONENUMBER</i>	26
397	8.6	CLASS <i>PERSONNAME</i>	26
398	9	REGISTRY ENTRY NAMING	26
399	10	ASSOCIATION OF REGISTRY ENTRIES	27
400	10.1	INTERFACE <i>ASSOCIATION</i>	27
401	10.1.1	Pre-defined Association Types	28
402	11	CLASSIFICATION OF REGISTRY ENTRIES	29
403	11.1	INTERFACE <i>CLASSIFICATIONNODE</i>	31
404	11.2	INTERFACE <i>CLASSIFICATION</i>	32
405	11.2.1	Context Sensitive Classification	33
406	11.3	EXAMPLE OF CLASSIFICATION SCHEMES	34
407	11.4	STANDARDIZED TAXONOMY SUPPORT	35
408	11.4.1	Full featured Taxonomy Based Classification	35
409	11.4.2	Light Weight Taxonomy Based Classification	35
410	12	INFORMATION MODEL: SECURITY VIEW	36
411	12.1	INTERFACE <i>ACCESSCONTROLPOLICY</i>	37
412	12.2	INTERFACE <i>PERMISSION</i>	38
413	12.3	INTERFACE <i>PRIVILEGE</i>	38
414	12.4	INTERFACE <i>PRIVILEGEATTRIBUTE</i>	39
415	12.5	INTERFACE <i>ROLE</i>	39
416	12.6	INTERFACE <i>GROUP</i>	39
417	12.7	INTERFACE <i>IDENTITY</i>	39
418	12.8	INTERFACE <i>PRINCIPAL</i>	40
419	13	REFERENCES	41
420	14	DISCLAIMER	41
421	15	CONTACT INFORMATION	42
422		COPYRIGHT STATEMENT	43

423 **Table of Figures**

424 [Figure 2: Information Model High Level Public View.....](#) 21

425 [Figure 3: Information Model *Inheritance* View.....](#) 27

426 [Figure 4: Example of RegistryEntry Association.....](#) 53

427 [Figure 5: Example showing a *Classification* Tree.....](#) 59

428 [Figure 6: Information Model *Classification* View.....](#) 60

429 [Figure 7: Classification *Instance* Diagram](#) 61

430 [Figure 8: Context Sensitive *Classification*.....](#) 65

431 [Figure 9: Information Model: Security View.....](#) 68

432 [Figure 1: Information Model Public View.....](#) 9

433 [Figure 2: Information Model Inheritance View.....](#) 12

434 [Figure 3: Example of Registry Entry Association.....](#) 27

435 [Figure 4: Example showing a Classification Tree.....](#) 30

436 [Figure 5: Information Model Classification View.....](#) 31

437 [Figure 6: Classification Instance Diagram](#) 31

438 [Figure 7: Context Sensitive Classification.....](#) 34

439 [Figure 8: Information Model: Security View.....](#) 37

440 **Table of Tables**

441 [Table 1: Sample *Classification* Schemes.....](#) 66

442 [Table 1: Sample Classification Schemes.....](#) 35

443

444

444 Introduction

445 3.12.1 Summary of Contents of Document

446 This document specifies the information model for the ebXML *Registry*.

447

448 A separate document, ebXML Registry Services Specification [[ebRS](#)], describes
449 how to build *Registry Services* that provide access to the information content in
450 the ebXML *Registry*.

451 3.22.2 General Conventions

452 ~~o UML diagrams are used as a way to concisely describe concepts. They are~~
453 ~~not intended to convey any specific *implementation* or methodology~~
454 ~~requirements.~~

455 ~~o Interfaces are often used in UML diagrams. They are used instead of *Classes*~~
456 ~~with attributes to provide an abstract definition without implying any specific~~
457 ~~*implementation*. Specifically, they do not imply that objects in the *Registry* will be~~
458 ~~accessed directly via these interfaces. Objects in the *Registry* are accessed via~~
459 ~~interfaces described in the ebXML Registry Services Specification. Each *get*~~
460 ~~method in every interface has an explicit indication of the attribute name that the~~
461 ~~*get* method maps to. For example *getName* method maps to an attribute named~~
462 ~~*name*.~~

463 ~~o *UML* diagrams are used as a way to concisely describe concepts. They~~
464 ~~are not intended to convey any specific *Implementation* or methodology~~
465 ~~requirements.~~

466 ~~The term “*repository item*” is used to refer to an object that has been~~
467 ~~submitted to a Registry for storage and safekeeping (e.g. an XML~~
468 ~~document or a DTD). Every repository item is described by a~~
469 ~~*RegistryEntryRegistryObject* instance. The term “*repository item*” is used to~~
470 ~~refer to actual *Registry* content (e.g. a *DTD*, as opposed to metadata~~
471 ~~about the *DTD*). It is important to note that the information model is not~~
472 ~~modeling actual repository items.~~

473 ~~o~~
474 ~~The term “*RegistryEntryRegistryObject*” is used to refer to an object that~~
475 ~~provides metadata about a *content instance* (repository item).~~

476 ~~o The term “*RegistryObject*” is also used to refer to the name of the most~~
477 ~~base class in the information model.~~

478 ~~interface in the information model to avoid the confusion with the common term~~
479 ~~“*object*”.~~

480 ~~However, when the term “*object*” is used to refer to a *class* or an interface in the~~
481 ~~information model, it may also mean *RegistryObject* because almost all~~
482 ~~classes are descendants of *RegistryObject*.~~

483
 484 ○ The information model does not contain deal with any elements that are
 485 the actual content of the Registry (repository item) repository. All
 486 eElements of the information model represent metadata about the content
 487 and not the content itself.

488
 489 Software practitioners MAY use this document in combination with other ebXML
 490 specification documents when creating ebXML compliant software.

491
 492 The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD,
 493 SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in
 494 this document, are to be interpreted as described in RFC 2119 [Bra97].
 495

496 2.2.1 Naming Conventions

497
 498 In order to enforce a consistent capitalization and naming convention in this
 499 document, "Upper Camel Case" (UCC) and "Lower Camel Case" (LCC)
 500 Capitalization styles are used in the following conventions

- 501
 502 ○ Element name is in UCC convention
 503 ○ (example: <UpperCamelCaseElement/>).
 504 ○ Attribute name is in LCC convention
 505 ○ (example: <UpperCamelCaseElement
 506 lowerCamelCaseAttribute="Whatever"/>).
 507 ○ Class, Interface names use UCC convention
 508 ○ (examples: ClassificationNode, Versionable).
 509 ○ Method name uses LCC convention
 510 ○ (example: getName(), setName()).

511
 512 Also, Capitalized Italics words are defined in the ebXML Glossary [ebGLOSS].

513 3.32.3 Audience

514 The target audience for this specification is the community of software
 515 developers who are:

- 516 ○ Implementers of ebXML *Registry Services*
 517 ○ Implementers of ebXML *Registry Clients*

518 3.42.4 Related Documents

519 The following specifications provide some background and related information to
 520 the reader:

- 521 a) ebXML Registry Business Domain Model [BDM] – defines requirements for
 522 ebXML Registry Services

- 523 ~~b)a)~~ a) ebXML Registry Services Specification [ebRS] - defines the actual
 524 Registry Services based on this information model
 525 ~~e)b)~~ b) ebXML Collaboration-Protocol Profile and Agreement Specification
 526 [ebCPPA] (~~under development~~) - defines how profiles can be defined for a
 527 pParty and how two pParties' profiles may be used to define a pParty
 528 agreement
 529 ~~d)c)~~ c) ebXML Business Process Specification Schema [BPebBPSS]
 530 d) ebXML Technical Architecture Specification [ebTA]
 531

532 4.3 Design Objectives

533 4.13.1 Goals

534 The goals of this version of the specification are to:

- 535 ○ Communicate what information is in the *Registry* and how that information
 536 is organized
- 537 ○ Leverage as much as possible the work done in the OASIS [OAS] and the
 538 ISO 11179 [ISO] Registry models
- 539 ○ Align with relevant works in progress within other ebXML working groups
- 540 ○ Be able to evolve to support future ebXML *Registry* requirements
- 541 ○ Be compatible with other ebXML specifications

542 4.2 ~~Caveats and Assumptions~~

543 ~~The Registry Information Model specification is first in a series of phased~~
 544 ~~deliverables. Later versions of the document will include additional functionality~~
 545 ~~planned for current and future development.~~

546 5.4 System Overview

547 5.14.1 Role of ebXML Registry

548
 549 The *Registry* provides a stable store where content information submitted by a
 550 *Submitting Organization* is made persistent. Such content information is used to
 551 facilitate ebXML-based Business to Business (B2B) partnerships and
 552 transactions. Submitted content may be XML schema and documents, process
 553 descriptions, Core Components, context descriptions, UML models,
 554 information about parties and even software components.

555 **5.24.2 Registry Services**

556 A set of *Registry Services* that provide access to *Registry* content to clients of the
 557 *Registry* is defined in the ebXML Registry Services Specification [ebRS]. This
 558 document does not provide details on these services but may occasionally refer
 559 to them.

560 **5.34.3 What the Registry Information Model Does**

561 The Registry Information Model provides a blueprint or high-level schema for the
 562 ebXML *Registry*. Its primary value is for implementers of ebXML *Registries*. It
 563 provides these implementers with information on the type of metadata that is
 564 stored in the *Registry* as well as the relationships among metadata *Classes*.

565 The Registry information model:

- 566 ○ Defines what types of objects are stored in the *Registry*
- 567 ○ Defines how stored objects are organized in the *Registry*
- 568 ○ Is based on ebXML metamodels from various working groups

570 **5.44.4 How the Registry Information Model Works**

571 Implementers of the ebXML *Registry* ~~may~~MAY use the information model to
 572 determine which *Classes* to include in their *Registry* ~~i~~Implementation and what
 573 attributes and methods these *Classes* ~~may~~may have. They ~~may~~MAY also use it
 574 to determine what sort of database schema their *Registry* ~~i~~Implementation
 575 ~~may~~may need.

576 [Note]The information model is meant to be
 577 illustrative and does not prescribe any
 578 specific ~~i~~Implementation choices.
 579

580 **5.54.5 Where the Registry Information Model May Be**

581 ~~Implemented~~
 582 The Registry Information Model ~~may~~MAY be implemented within an ebXML
 583 *Registry* in ~~the~~ form of a relational database schema, object database schema or
 584 some other physical schema. It ~~may~~MAY also be implemented as interfaces and
 584 *Classes* within a *Registry* ~~i~~Implementation.

585 **5.64.6 Conformance to an ebXML Registry**

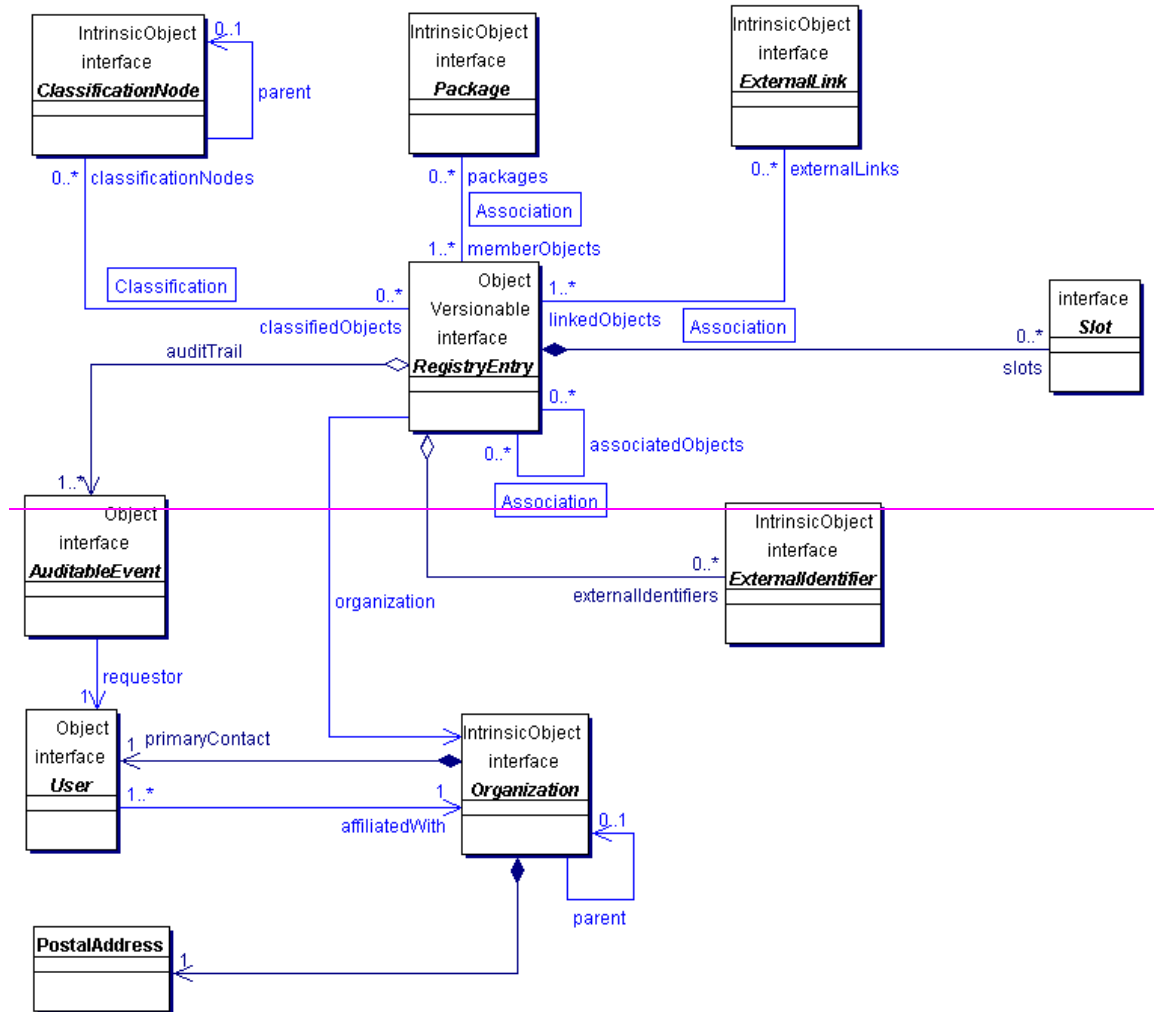
586
 587 If an ~~i~~Implementation claims ~~e~~Conformance to this specification then it supports
 588 all required information model *Classes* and interfaces, their attributes and their
 589 semantic definitions that are visible through the ebXML *Registry Services*.

590 **65 Registry Information Model: High LevelPublic Public**
591 **View**

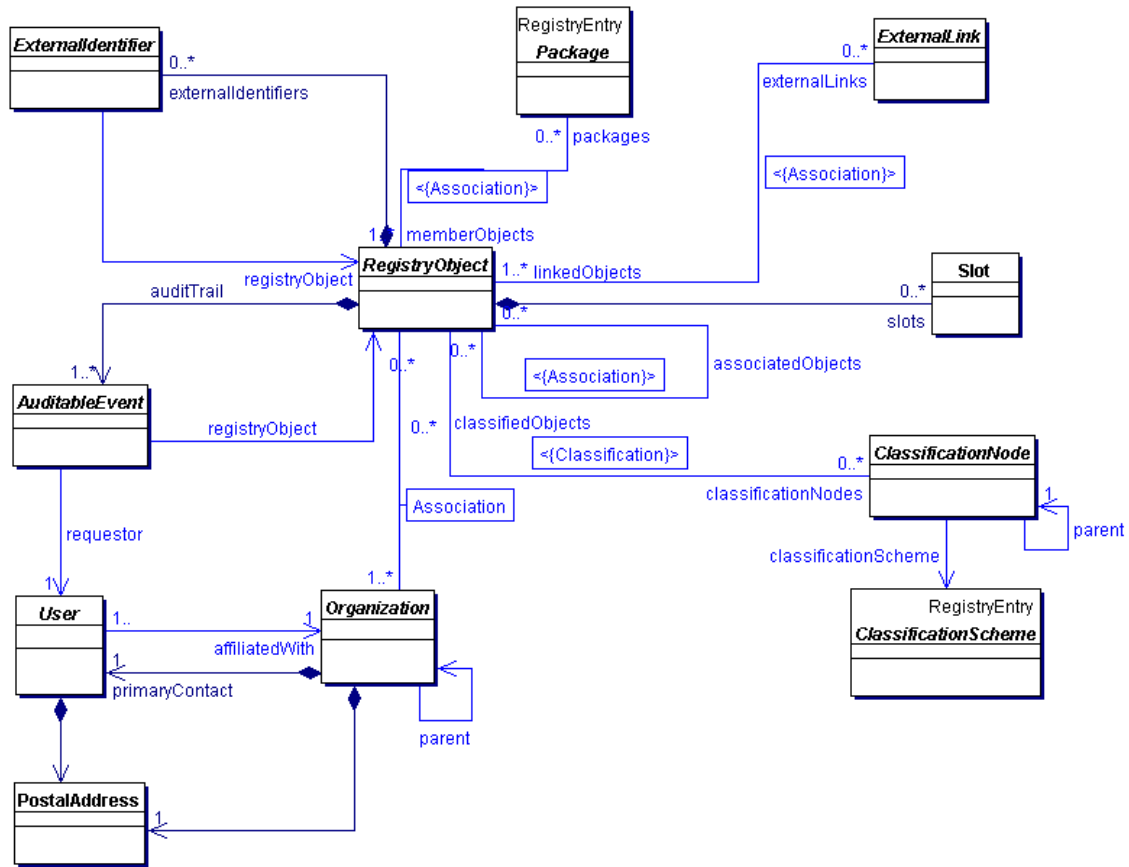
592 This ~~chapter~~section provides a high level public view of the most visible objects
593 in the *Registry*.

594
595 ~~Figure 1~~~~Figure 1~~~~Figure 1~~~~Figure 1~~ shows the high level public view of the
596 objects in the *Registry* and their relationships as a UML Class Digram. It does
597 not show inheritance, Class attributes or Class methods.

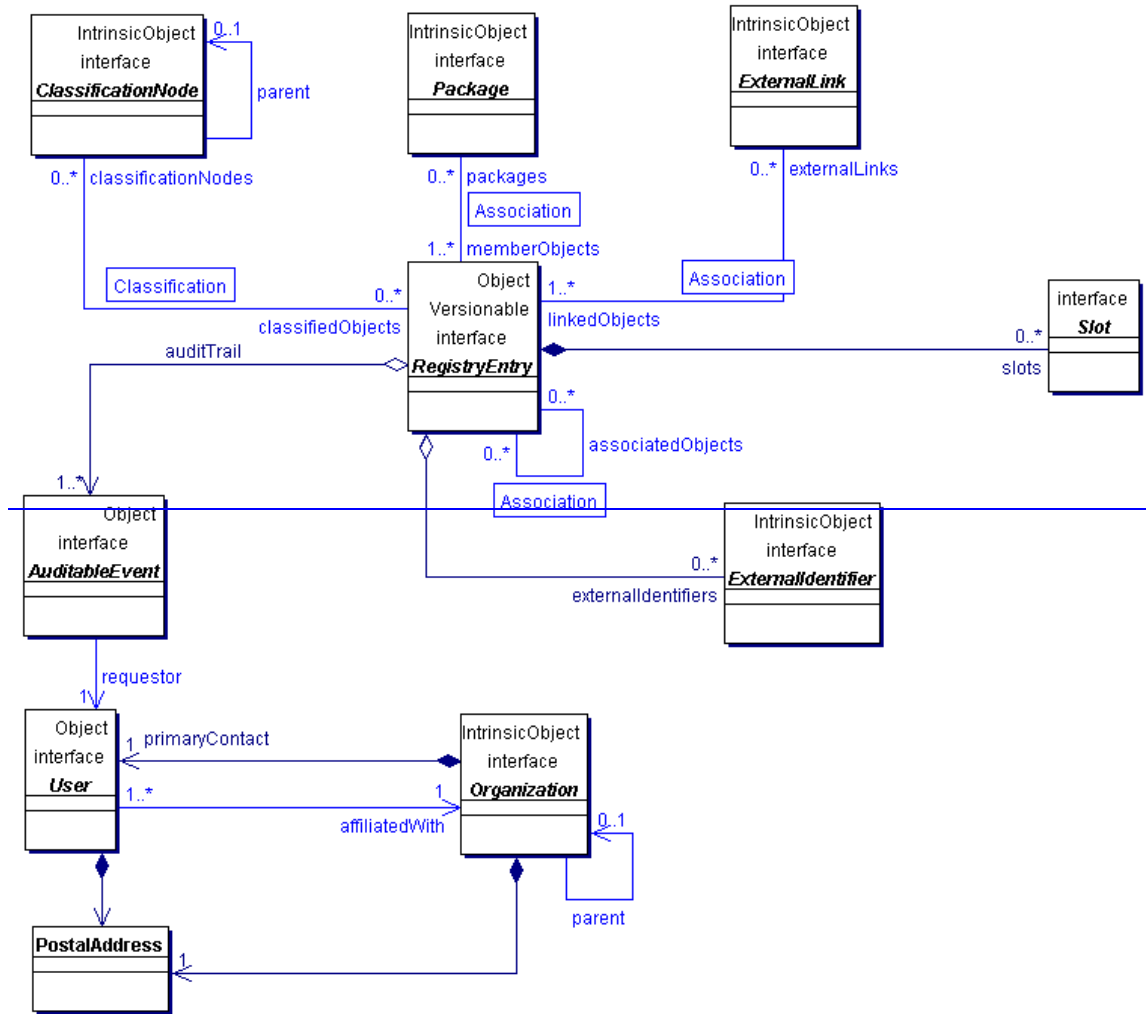
598 The reader is again reminded that the information model is not modeling actual
599 repository items.
600



601



602



603

604

Figure 14: Information Model High Level Public View

605 **6.15.1 RegistryObjectEntry**

606 The RegistryObject class is an abstract base class used by most classes in the
 607 model. It provides minimal metadata for registry objects. It also provides methods
 608 for accessing related objects that provide additional dynamic metadata for the
 609 registry object. The central object in the information model is a RegistryEntry. An
 610 i/instance of RegistryEntry exists for each content i/instance submitted to the
 611 Registry. Instances of the RegistryEntry C/class provide metadata about a
 612 repository item in the Registry. The actual repository item (e.g. a DTD) is not
 613 contained in an i/instance of the RegistryEntry C/class. Note that most C/classes in
 614 the information model are specialized sub-classes of RegistryEntry. Each
 615 RegistryEntry is related to exactly one repository item, however, in the future
 616 revision of this document, it may be related to multiple repository items.

617 **6.25.2 Slot**

618 Slot [instances](#) provide a dynamic way to add arbitrary attributes to
619 [RegistryEntryRegistryObject instances](#). This ability to add attributes
620 dynamically to [RegistryEntryRegistryObject instances](#) enables
621 extensibility within the Registry Information Model. For example, if a company
622 wants to add a “copyright” attribute to each RegistryObject instance that it
623 submits, it can do so by adding a slot with name “copyright” and value containing
624 the copyrights statement.

625 **6.35.3 Association**

626 Association [instances](#) are [RegistryEntriesRegistryObject instances](#) that
627 are used to define many-to-many associations between objects in the information
628 model. Associations are described in detail in [chaptersection 940](#).

629 **6.45.4 ExternalIdentifier**

630 ExternalIdentifier [instances](#) provide additional identifier information to [a
631 RegistryEntryRegistryObject instance](#), such as DUNS number, Social Security
632 Number, or an alias name of the organization.

633 **6.55.5 ExternalLink**

634 ExternalLink [instances](#) are [RegistryEntriesRegistryObject instances](#)
635 that model a named URI to content that is not managed by the *Registry*. Unlike
636 managed content, such external content may change or be deleted at any time
637 without the knowledge of the *Registry*. A [RegistryEntryRegistryObject instance](#)
638 may be associated with any number of ExternalLinks.
639 Consider the case where a *Submitting Organization* submits a repository item
640 (e.g. a *DTD*) and wants to associate some external content to that object (e.g.
641 the *Submitting Organization's* home page). The ExternalLink enables this
642 capability. A potential use of the ExternalLink capability may be in a GUI tool that
643 displays the ExternalLinks to a [RegistryEntryRegistryObject](#). The user may click
644 on such links and navigate to an external web page referenced by the link.

645 **5.6 ClassificationScheme**

646 A ClassificationScheme instance is a RegistryObject instance that represents a
647 structured way to classify or categorize RegistryObject instances. A very
648 common example of a classification scheme in science is the *Classification of
649 living things* where living things are categorized in under a tree like structure.
650 Another example is the Dewey Decimal system used in libraries to categorize
651 books and other publications. ClassificationScheme is described in detail in
652 [section 10](#).

653 **6.65.7 ClassificationNode**

654 ClassificationNode [instances](#) are [RegistryEntriesRegistryObject](#)
655 [instances](#) that are used to define tree structures [under a ClassificationScheme](#),
656 where each node in the tree is a ClassificationNode [and the root is the](#)
657 [ClassificationScheme](#). *Classification* trees constructed with ClassificationNodes
658 are used to define [eClassification](#) schemes or ontologies. ClassificationNode is
659 described in detail in [chaptersection 1044](#).

660 **6.75.8 Classification**

661 Classification [instances](#) are [RegistryObjectEntries](#) that are used to classify
662 [repository items by associating their RegistryEntryother RegistryObject](#)
663 [instances](#) with a ClassificationNode within a [eClassificationS](#)-scheme.
664 Classification is described in detail in [chaptersection 1044](#).

665 **6.85.9 Package**

666 Package [instances](#) are [RegistryEntry instancesies](#) that group logically related
667 [RegistryEntriesRegistryObject instances](#) together. One use of a Package is to
668 allow operations to be performed on an entire [pPackage](#) of objects. For example
669 all objects belonging to a Package may be deleted in a single request.

670 **6.95.10 AuditableEvent**

671 AuditableEvent [instances](#) are [ObjectsRegistryObject instances](#) that
672 are used to provide an audit trail for [RegistryEntriesRegistryObject instances](#).
673 AuditableEvent is described in detail in [chaptersection 78](#).

674 **6.105.11 User**

675 User [instances](#) are [ObjectsRegistryObject instances](#) that are used to
676 provide information about registered users within the [rRegistry](#). User objects are
677 used in audit trail for [RegistryEntriesRegistryObject instances](#). User is described
678 in detail in [chaptersection 78](#).
679

680 **6.115.12 PostalAddress**

681 PostalAddress is a simple reusable [eEntity](#)-[Class](#) that defines attributes of a
682 postal address.
683

684 **6.125.13 Organization**

685 Organization [instances](#) are [RegistryEntriesRegistryObject instances](#)
686 that provide information on organizations such as a *Submitting Organization*.
687 Each Organization [instance](#) may have a reference to a parent Organization.

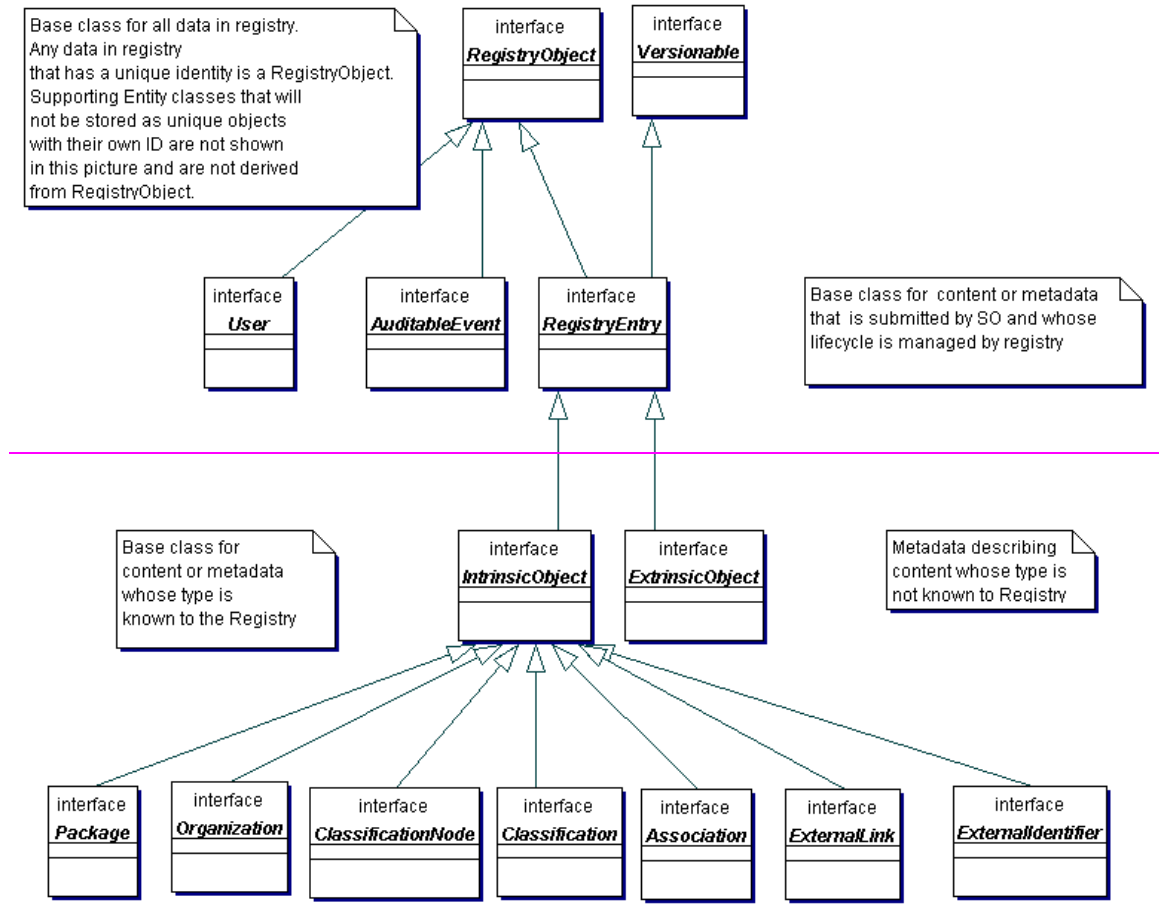
688 **76 Registry Information Model: Detail View**

689 This [chaptersection](#) covers the information model [Classes](#) in more detail than
690 the Public View. The detail view introduces some additional [Classes](#) within the
691 model that were not described in the public view of the information model.

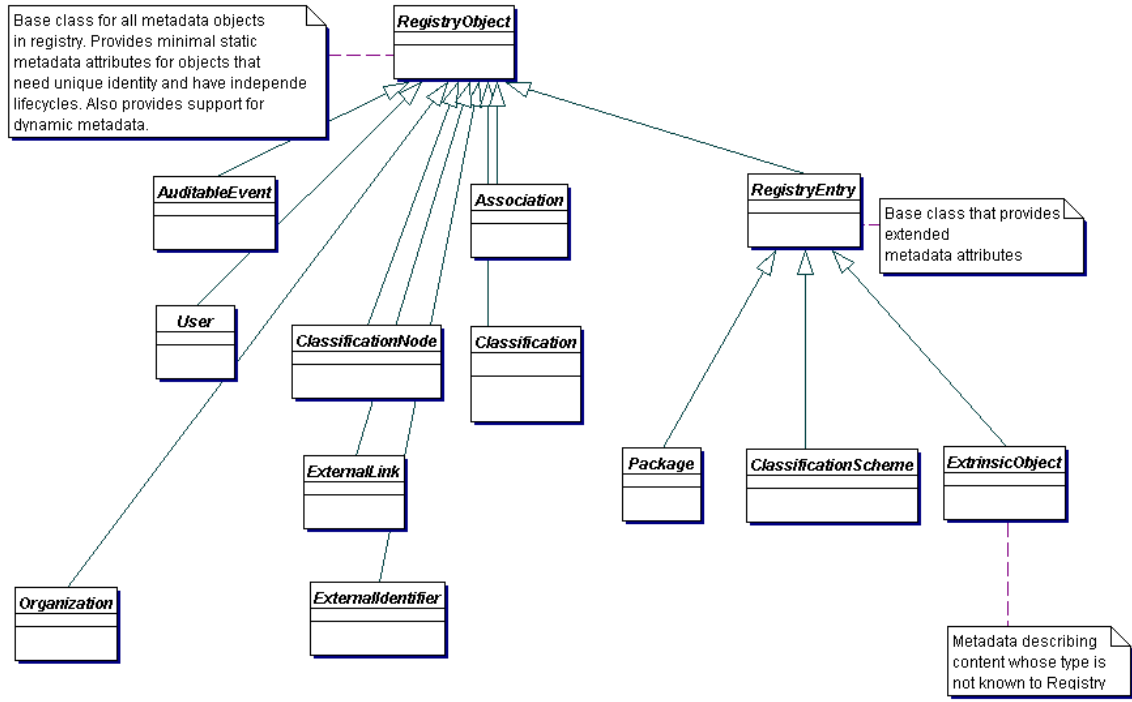
692
693 [Figure 2](#)~~[Figure 2](#)~~~~[Figure 2](#)~~~~[Figure 2](#)~~ shows the [inheritance](#) or “is a”
694 relationships between the [Classes](#) in the information model. Note that it does
695 not show the other types of relationships, such as “has a” relationships, since
696 they have already been shown in a previous figure. *Class* attributes and *class*
697 methods are also not shown. Detailed description of methods and attributes of
698 most interfaces and [Classes](#) will be displayed in tabular form following the
699 description of each [Class](#) in the model.

700
701 The [interfaceclass](#) Association will be covered in detail separately in
702 [chaptersection 910](#). The [interfacesclasses](#) Classification and ClassificationNode
703 will be covered in detail separately in [chaptersection 1011](#).

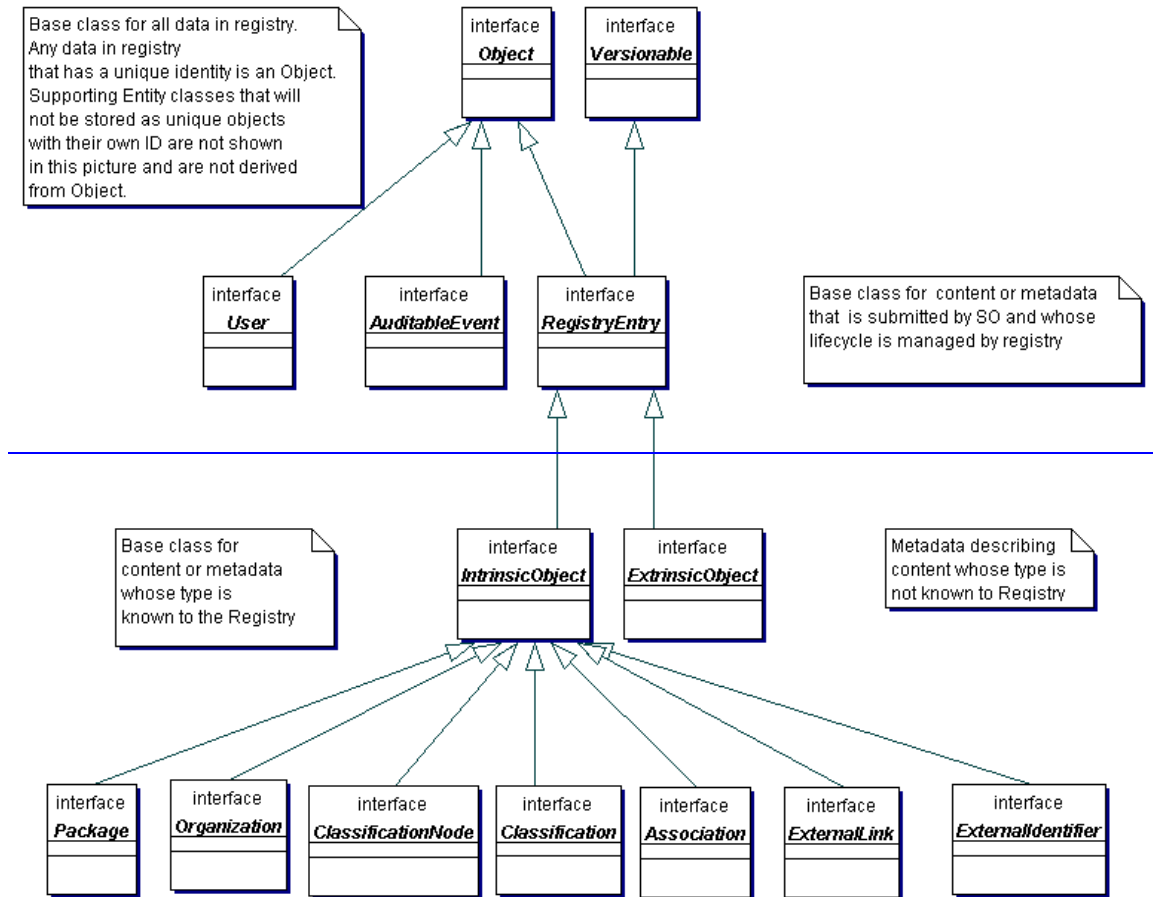
704
705 The reader is again reminded that the information model is not modeling actual
706 repository items.



707



708



709
710

Figure 222: Information Model *Inheritance View*

711 **6.1 Attribute and Methods of Information Model Classes**

712 Information model classes are defined primarily in terms of the attributes they
 713 carry. These attributes provide state information on instances of these classes.
 714 Implementations of a registry often map class attributes to attributes in an XML
 715 store or columns in a relational store.

716 Information model classes may also have methods defined for them. These
 717 methods provide additional behavior for the class they are defined within.
 718 Methods are currently used in mapping to SQL stored procedures in the SQL
 719 query capability defined in [ebRS].

720
 721
 722 Since the model supports inheritance between classes, it is usually the case that
 723 a class in the model inherits attributes and methods from its base classes, in
 724 addition to defining its own specialized attributes and methods.

725 **6.2 Data Types**

726 This following table lists the various data types used by the attributes within
 727 information model classes:
 728

<u>Data Type</u>	<u>Primitive Type</u>	<u>Description</u>	<u>Length</u>
<u>Boolean</u>		Used for a true or false value	
<u>String4</u>	<u>String</u>	Used for 4 character long strings	4 characters
<u>String8</u>	<u>String</u>	Used for 8 character long strings	8 characters
<u>ShortName</u>	<u>String</u>	A short text string	64 characters
<u>LongName</u>	<u>String</u>	A long text string	128 characters
<u>FreeFormText</u>	<u>String</u>	A very long text string for free-form text	256 characters
<u>UUID</u>	<u>String</u>	DCE 128 Bit Universally unique Ids used for referencing another object	64 characters
<u>URI</u>	<u>String</u>	Used for URL and URN values	256 characters
<u>Integer</u>		Used for integer values	4 bytes
<u>Timestamp</u>		Used for a time stamp value such as Date	

729

730 **7.16.3 InterfaceClass ObjectRegistryObject**731 **All Known Subinterfaces:**732 **Direct Known Subclasses:**

733 Association, AuditableEvent, Classification, ClassificationNode,
 734 ExternalIdentifier, ExternalLink, Organization, RegistryEntry,
 735 UserAssociation, Classification, ClassificationNode, ExternalLink,
 736 ExtrinsicObject, IntrinsicObject, RegistryEntry, Organization, Package,
 737 User, AuditableEvent, ExternalIdentifier, Submission

738

739 **ObjectRegistryObject** provides a common base **interfaceclass** for almost all
 740 objects in the information model. Information model **εClasses** whose
 741 **instancesinstances** have a unique identity and an independent life cycle are
 742 descendants of the **ObjectRegistryObject εClass**.

743

744 Note that Slot and PostalAddress are not descendants of the
 745 **ObjectRegistryObject εClass** because their **instancesinstances** do not have an
 746 independent existence and unique identity. They are always a part of some other
 747 **εClass's instance** (e.g. Organization has a PostalAddress).

748 **6.3.1 Attribute Summary**

749 The following is the first of many tables that summarize the attributes of a class.
 750 The columns in the table are described as follows:

<u>Column</u>	<u>Description</u>
<u>Attribute</u>	<u>The name of the attribute</u>
<u>Data Type</u>	<u>The data type for the attribute</u>
<u>Required</u>	<u>Specifies whether the attribute is required to be specified</u>
<u>Default</u>	<u>Specifies the default value in case the attribute is omitted</u>
<u>Specified By</u>	<u>Indicates whether the attribute is specified by the client or specified by the registry. In some cases it may be both</u>
<u>Mutable</u>	<u>Specifies whether an attribute may be changed once it has been set to a certain value</u>

751
 752

<u>Attribute</u>	<u>Data Type</u>	<u>Required</u>	<u>Default Value</u>	<u>Specified By</u>	<u>Mutable</u>
<u>accessControlPolicy</u>	<u>UUID</u>	<u>No</u>		<u>Registry</u>	<u>No</u>
<u>description</u>	<u>FreeFormText</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>
<u>id</u>	<u>UUID</u>	<u>Yes</u>		<u>Client or registry</u>	<u>No</u>
<u>name</u>	<u>LongName</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>
<u>objectType</u>	<u>LongName</u>	<u>Yes</u>		<u>Registry</u>	<u>No</u>

753

754 **6.3.2 Attribute accesControlPolicy**

755 Each RegistryObject instance has an AccessControlPolicy instance associated
 756 with it. An AccessControlPolicy instance defines the *Security Model* associated
 757 with the RegistryObject in terms of “who is permitted to do what” with that
 758 RegistryObject.

759 **6.3.3 Attribute description**

760 Each RegistryObject instance may have textual description in a human readable
 761 and user-friendly manner.

762 **6.3.4 Attribute id**

763 Each RegistryObject instance must have a universally unique ID. Registry
 764 objects use the id of other RegistryObject instances for the purpose of
 765 referencing those objects.

766
 767 Note that some classes in the information model do not have a need for a unique
 768 id. Such classes do not inherit from RegistryObject class. Examples include
 769 Entity classes such as TelephoneNumber, PostalAddress and PersonName.

770

771 The id attribute of various derived classes of RegistryObject fall into two
772 categories:

773

774 1. UUID based Id

775 2. Attribute based Id

776

777 6.3.4.1 UUID based Id

778 Most classes derived from RegistryObject have an id that is a Universally Unique
779 ID as defined by [UUID]. Such UUID based Id attributes may be specified by the
780 client. If the UUID based ID is not specified, it must be generated by the registry
781 when a new RegistryObject instance is first submitted to the registry.

782 6.3.4.2 Attribute based Id

783 A few classes derived from RegistryObject have an Id that is not a UUID but is
784 instead composed of multiple attributes of that object. This is very similar to the
785 concept of a multi-column primary key in relational databases, or multi-attribute
786 key instances in XML Schema.

787

788 Examples of classes that use attribute based Id are Classification, Association
789 and ExternalIdentifier. The reason these objects do not use UUIDs and instead
790 use attribute based Id is that they do not have an independent lifecycle separate
791 from their primary RegistryObject.

792

793 Attribute based Ids are not UUIDs and therefore are not constrained by the 64 bit
794 limit of the UUID data type. Instead they can be of arbitrary length.

795 6.3.5 Attribute name

796 Each RegistryObject instance may have human readable name. The name does
797 not need to be unique with respect other RegistryObject instances.

798 6.3.6 Attribute objectType

799 Each RegistryObject instance has an objectType. The objectType for almost all
800 objects in the information model is the name of their class. For example the
801 objectType for a Classification is "Classification". The only exception to this rule
802 is that the objectType for an ExtrinsicObject instance is user defined and
803 indicates the type of repository item associated with the ExtrinsicObject.

804 6.3.6.1 Pre-defined Object Types

805 The following table lists pre-defined object types. Note that for an ExtrinsicObject
806 there are many types defined based on the type of repository item the
807 ExtrinsicObject catalogs. In addition there are object types defined for
808 IntrinsicObject sub-classes that may have concrete instances.

809

810
811
812
813

These pre-defined object types are defined as a *ClassificationScheme*. While the scheme may easily be extended a *Registry* MUST support the object types listed below.

<u>Name</u>	<u>description</u>
<u>Unknown</u>	An ExtrinsicObject that catalogues content whose type is unspecified or unknown.
<u>CPA</u>	An ExtrinsicObject of this type catalogues an XML document <i>Collaboration Protocol Agreement (CPA)</i> representing a technical agreement between two parties on how they plan to communicate with each other using a specific protocol.
<u>CPP</u>	An ExtrinsicObject of this type catalogues an document called <i>Collaboration Protocol Profile (CPP)</i> that provides information about a <i>Party</i> participating in a <i>Business</i> transaction.
<u>Process</u>	An ExtrinsicObject of this type catalogues a process description document.
<u>Role</u>	An ExtrinsicObject of this type catalogues an XML description of a <i>Role</i> in a <i>Collaboration Protocol Profile (CPP)</i> .
<u>ServiceInterface</u>	An ExtrinsicObject of this type catalogues an XML description of a service interface as defined by [ebCPP].
<u>SoftwareComponent</u>	An ExtrinsicObject of this type catalogues a software component (e.g., an EJB or <i>Class</i> library).
<u>Transport</u>	An ExtrinsicObject of this type catalogues an XML description of a transport configuration as defined by [ebCPP].
<u>UMLModel</u>	An ExtrinsicObject of this type catalogues a <i>UML</i> model.
<u>XMLSchema</u>	An ExtrinsicObject of this type catalogues an XML schema (<i>DTD</i> , XML Schema, RELAX grammar, etc.).
<u>Package</u>	A Package object
<u>ExternalLink</u>	An ExternalLink object
<u>ExternalIdentifier</u>	An ExternalIdentifier object
<u>Association</u>	An Association object
<u>Classification</u>	A Classification object
<u>ClassificationNode</u>	A ClassificationNode object
<u>AuditableEvent</u>	An AuditableEvent object
<u>User</u>	A User object

<u>Organization</u>	An Organization object
---------------------	------------------------

814

-

815 **6.3.7 Method Summary**

816 In addition to its attributes, the RegistryObject class also defines the following
 817 methods. These methods are used to navigate relationship links from a
 818 RegistryObject instance to other objects.

819

Method Summary for RegistryObject	
<u>Collection</u> ¹	<u>getAssociatedObjects()</u> _____ Gets the collection of RegistryObject instances associated with this object.
<u>Collection</u>	<u>getAssociations()</u> _____ Gets all Associations where this object is the source of the Association.
<u>Collection</u>	<u>getAuditTrail()</u> _____ Gets the complete audit trail of all requests that effected a state change in this object as an ordered Collection of AuditableEvent objects.
<u>Collection</u>	<u>getClassificationNodes()</u> _____ Gets the ClassificationNodes that classify this object.
<u>Collection</u>	<u>getClassifications()</u> _____ Gets the Classification that classify this object.
<u>Collection</u>	<u>getExternalIdentifiers()</u> _____ Gets the collection of ExternalIdentifiers associated with this object.
<u>Collection</u>	<u>getExternalLinks()</u> _____ Gets the ExternalLinks associated with this object.
<u>Collection</u>	<u>getOrganizations(String type)</u> _____ Gets the Organizations associated with this object. If a non-null type is specified it is used as a filter to match only specified type of organizations as indicated by the associationType attribute in the Association() instance linking the object to the Organization.
<u>Collection</u>	<u>getPackages()</u> _____ Gets the Packages that this object is a member of.
<u>Collection</u>	<u>getSlots()</u> _____ Gets the Slots associated with this object.

¹ A Collection represents a collection of multiple RegistryObject instances

820
821
822
823

823 **7.2Interface Versionable**

824 **All Known Subinterfaces:**

825 [Association](#), [Classification](#), [ClassificationNode](#), [ExternalLink](#),
 826 [ExtrinsicObject](#), [IntrinsicObject](#), [RegistryEntry](#), [Organization](#), [Package](#),
 827 [ExternalIdentifier](#)

828

829 The Versionable interface defines the behavior common to ~~e~~Cclasses that are
 830 capable of creating versions of their ~~i~~instances. At present all RegistryEntry
 831 ~~e~~Cclasses are ~~required~~REQUIRED to implement the Versionable interface.

832

Method Summary of Versionable	
-int	getMajorVersion() _____ Gets the major revision number for this version of the Versionable object. Maps to attribute named majorVersion.
-int	getMinorVersion() _____ Gets the minor revision number for this version of the Versionable object. Maps to attribute named minorVersion.
-void	setMajorVersion(int majorVersion) _____ Set Gets the major revision number for this version of the Versionable object.
-void	setMinorVersion(int minorVersion) _____ Sets the minor revision number for this version of the Versionable object.

833 -

834 **7.36.4 InterfaceClass RegistryEntry**

835 **Super Classes:**

836 [RegistryObject](#)

837

838 **All Superinterfaces:**

839 [Object](#)[RegistryObject](#), [Versionable](#)

840 **Direct Known Subclasses:**

841 [ClassificationScheme](#), [ExtrinsicObject](#), [Package](#)**All Known Subinterfaces:**

842 [Association](#), [Classification](#), [ClassificationNode](#), [ExternalLink](#),
 843 [ExtrinsicObject](#), [IntrinsicObject](#), [Organization](#), [Package](#), [ExternalIdentifier](#)

844

845 RegistryEntry is a common base ~~e~~Cclass for classes in the information model that
 846 require additional metadata beyond the minimal metadata provided by

847 RegistryObject class. The additional metadata is described by the attributes of
 848 the RegistryEntry class below.

849 **6.4.1 Attribute Summary**~~all metadata describing submitted content whose~~
 850 ~~life cycle is managed by the rRegistry. Metadata describing content~~
 851 ~~submitted to the rRegistry is further specialized by the~~
 852 **ExtrinsicObject and IntrinsicObject subclasses of RegistryEntry.**

853

<u>Attribute</u>	<u>Data Type</u>	<u>Required</u>	<u>Default Value</u>	<u>Specified By</u>	<u>Mutable</u>
<u>expiration</u>	<u>Timestamp</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>
<u>majorVersion</u>	<u>Integer</u>	<u>Yes</u>	<u>1</u>	<u>Registry</u>	<u>Yes</u>
<u>minorVersion</u>	<u>Integer</u>	<u>Yes</u>	<u>0</u>	<u>Registry</u>	<u>Yes</u>
<u>stability</u>	<u>LongName</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>
<u>status²</u>	<u>LongName</u>	<u>Yes</u>		<u>Registry</u>	<u>Yes</u>
<u>userVersion</u>	<u>ShortName</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>

854

855

856 Note that attributes inherited by RegistryEntry class from the RegistryObject
 857 class are not shown in the table above.

858

859

860 **6.4.2 Attribute expiration**

861 Each RegistrEntry instance may have an expirationDate. This attribute defines a
 862 time limit upon the stability guarantee provided by the stability attribute. Once the
 863 expirationDate has been reached the stability attribute in effect becomes
 864 STABILITY_DYNAMIC implying that content can change at any time and in any
 865 manner. A null value implies that there is no expiration on stability attribute.

866 **6.4.3 Attribute majorVersion**

867 Each RegistrEntry instance must have a major revision number for the current
 868 version of the RegistryEntry instance. This number is assigned by the registry
 869 when the object is created. This number may be updated by the registry when an
 870 object is updated.

871 **6.4.4 Attribute minorVersion**

872 Each RegistrEntry instance must have a minor revision number for the current
 873 version of the RegistryEntry instance. This number is assigned by the registry

² Was Integer in RIM 1.0 for some reason.

874 when the object is created. This number may be updated by the registry when an
 875 object is updated.

876 **6.4.5 Attribute stability**

877 Each RegistrEntry instance may have a stability indicator. The stability indicator
 878 is provided by the submitter as a guarantee of the level of stability for the content.

879 **6.4.5.1 Pre-defined RegistryEntry Stability Enumerations**

880 The following table lists pre-defined choices for RegistryEntry stability attribute.
 881 These pre-defined stability types are defined as a *Classification* scheme. While
 882 the scheme may easily be extended, a *Registry* MAY support the stability types
 883 listed below.

<u>Name</u>	<u>Description</u>
<u>Dynamic</u>	<u>Stability of a RegistryEntry that indicates that the content is dynamic and may be changed arbitrarily by submitter at any time.</u>
<u>DynamicCompatible</u>	<u>Stability of a RegistryEntry that indicates that the content is dynamic and may be changed in a backward compatible way by submitter at any time.</u>
<u>Static</u>	<u>Stability of a RegistryEntry that indicates that the content is static and will not be changed by submitter.</u>

885

886 **6.4.6 Attribute status**

887 Each RegistryEntry instance must have a life cycle status indicator. The status is
 888 assigned by the registry.

889 **6.4.6.1 Pre-defined RegistryObject Status Types**

890 The following table lists pre-defined choices for RegistryObject status attribute.
 891 These pre-defined status types are defined as a *Classification* scheme. While the
 892 scheme may easily be extended, a *Registry* MUST support the status types listed
 893 below.

894

<u>Name</u>	<u>Description</u>
<u>Submitted</u>	<u>Status of a RegistryObject that catalogues content that has been submitted to the <i>Registry</i>.</u>
<u>Approved</u>	<u>Status of a RegistryObject that catalogues content that has been submitted to the <i>Registry</i> and has been subsequently approved.</u>
<u>Deprecated</u>	<u>Status of a RegistrvObiect that catalogues content that has</u>

	been submitted to the <i>Registry</i> and has been subsequently deprecated.
<u>Withdrawn</u>	Status of a RegistryObject that catalogues content that has been withdrawn from the <i>Registry</i> .

895

896 **6.4.7 Attribute userVersion**

897 Each RegistrEntry instance may have a userVersion. The userVersion is similar
 898 to the majorVersion-minorVersion tuple. They both provide an indication of the
 899 version of the object. The majorVersion-minorVersion tuple is provided by the
 900 registry while userVersion provides a user specified version for the object.-

901 **7.3.1 Pre-defined RegistryEntry Status Types**

902 The following table lists pre-defined choices for RegistryEntry status attribute.
 903 These pre-defined status types are defined as a *Classification* scheme. While the
 904 scheme may easily be extended, a *rRegistry* ~~must~~ **MUST** support the status
 905 types listed below.

906

Name	Description
Submitted	Status of a RegistryEntry that catalogues content that has been submitted to the <i>Registry</i> .
Approved	Status of a RegistryEntry that catalogues content that has been submitted to the <i>Registry</i> and has been subsequently approved.
Deprecated	Status of a RegistryEntry that catalogues content that has been submitted to the <i>Registry</i> and has been subsequently deprecated.
Withdrawn	Status of a RegistryEntry that catalogues content that has been withdrawn from the <i>Registry</i> .

907 **7.3.2 Pre-dDefined Object Types**

908 The following table lists pre-defined object types. Note that for an ExtrinsicObject
 909 there are many types defined based on the type of repository item the
 910 ExtrinsicObject catalogs. In addition there there are object types defined for
 911 IntrinsicObject sub-classes that may have concrete *i* instances.

912

913 These pre-defined object types are defined as a *Classification* scheme. While the
 914 scheme may easily be extended a *rRegistry* ~~must~~ **MUST** support the object types
 915 listed below.

916

name	description
Unknown	An ExtrinsicObject that catalogues content whose type is unspecified or unknown.
CPA	An ExtrinsicObject of this type catalogues an XML document <i>Collaboration Protocol Agreement (CPA)</i> representing a technical agreement between two parties on how they plan to communicate with each other using a specific protocol.
CPP	An ExtrinsicObject of this type catalogues an XML document called <i>Collaboration Protocol Profile (CPP)</i> that provides information about a <i>pParty</i> participating in a <i>Bbusiness</i> transaction.
Process	An ExtrinsicObject of this type catalogues a process description document.
Role	An ExtrinsicObject of this type catalogues an XML description of a <i>Role</i> in a <i>Collaboration Protocol Profile (CPP)</i> .
ServiceInterface	An ExtrinsicObject of this type catalogues an XML description of a service interface as defined by [CPAebCPP].
SoftwareComponent	An ExtrinsicObject of this type catalogues a software component (e.g., an EJB or <i>cClass</i> library).
Transport	An ExtrinsicObject of this type catalogues an XML description of a transport configuration as defined by [CPAebCPP].
UMLModel	An ExtrinsicObject of this type catalogues a <i>UML</i> model.
XMLSchema	An ExtrinsicObject of this type catalogues an XML schema (<i>DTD</i> , XML Schema, RELAX grammar, etc.).
Package	A Package object
ExternalLink	An ExternalLink object
ExternalIdentifier	An ExternalIdentifier object
Association	An Association object
Classification	A Classification object
ClassificationNode	A ClassificationNode object
AuditableEvent	An AuditableEvent object
User	A User object
Organization	An Organization object

917

918 **7.3.3 Pre-defined RegistryEntry Stability Enumerations**

919 The following table lists pre-defined choices for RegistryEntry stability attribute.
 920 These pre-defined stability types are defined as a *Classification* scheme. While
 921 the scheme may easily be extended, a *rRegistry* **must** *MAY* support the stability
 922 types listed below.

923

Name	Description
Dynamic	Stability of a RegistryEntry that indicates that the content is dynamic and may be changed arbitrarily by submitter at any time.
DynamicCompatible	Stability of a RegistryEntry that indicates that the content is dynamic and may be changed in a backward compatible way by submitter at any time.
Static	Stability of a RegistryEntry that indicates that the content is static and will not be changed by submitter.

924

925

926 **7.4.6.5 InterfaceClass Slot**

927

928 Slot *instances* provide a dynamic way to add arbitrary attributes to
 929 *RegistryEntryRegistryObject instances*. This ability to add attributes
 930 dynamically to *RegistryEntryRegistryObject instances* enables
 931 extensibility within the *RegistryInformation Model*.

932

933 In this model, a *RegistryEntryRegistryObject* may have 0 or more Slots. A slot
 934 is composed of a name, a slotType and a collection of values.

935 **6.5.1 Attribute Summary** The name of slot is locally unique within the
 936 *RegistryEntry instance*. Similarly, the value of a Slot is locally
 937 unique within a slot *instance*. Since a Slot represent an extensible
 938 attribute whose value may be a collection, therefore a Slot is allowed
 939 to have a collection of values rather than a single value. The slotType
 940 attribute may optionally specify a type or category for the slot.

941

Attribute	Data Type	Required	Default Value	Specified By	Mutable
name	LongName	Yes		Client	No
slotType	LongName	Yes		Client	No
values	Collection of ShortName	Yes		Client	No

942

943 **6.5.2 Attribute name**

944 Each Slot instance must have a name. The name is the primary means for
 945 identifying a Slot instance within a RegistryObject. Consequently, the name of a
 946 Slot instance must be locally unique within the RegistryObject Instance.

947 **6.5.3 Attribute slotType**

948 Each Slot instance may have a slotType that allows different slots to be grouped
 949 together.

950 **6.5.4 Attribute values**

951 A Slot instance must have a Collection of values. Since a Slot represent an
 952 extensible attribute whose value may be a collection, therefore a Slot is allowed
 953 to have a collection of values rather than a single
 954 value.

955

956

957 **7.56.6 InterfaceClass ExtrinsicObject**958 **All Super interfaceClasses:**959 RegistryEntry, ObjectRegistryObject, Versionable

960

961

962 ExtrinsicObjects provide metadata that describes submitted content whose type
 963 is not intrinsically known to the *rRegistry* and therefore **mustMUST** be described
 964 by means of additional attributes (e.g., mime type).

965

966 Since the registry can contain arbitrary content without intrinsic knowledge about
 967 that content, ExtrinsicObjects require special metadata attributes to provide some
 968 knowledge about the object (e.g. mime type).

969

970 Examples of content described by ExtrinsicObject include *Collaboration Protocol*
 971 *Profiles (CPP)*, *Bbusiness Pprocess* descriptions, and schemas.

972 **6.6.1 Attribute Summary**

973

<u>Attribute</u>	<u>Data Type</u>	<u>Required</u>	<u>Default Value</u>	<u>Specified By</u>	<u>Mutable</u>
contentURI	URI	Yes		Registry	No
isOpaque	Boolean	No	false	Client	No
mimeType	LongName	Yes		Client	No

974

975

Note that attributes inherited from RegistryEntry and RegistryObject are not shown in the table above.

976

977

6.6.2 Attribute contentURI

978

Each ExtrinsicObject instance must have a contentURI attribute. The contentURI is a URI to the repository item that is catalogued by this ExtrinsicObject instance.

979

980

The contentURI is assigned by the Registry and must be resolvable by the registry.

981

982

6.6.3 Attribute isOpaque

983

Each ExtrinsicObject instance may have an isOpaque attribute defined. This attribute determines whether the content catalogued by this ExtrinsicObject is

984

985

opaque to (not readable by) the Registry. In some situations, a Submitting

986

Organization may submit content that is encrypted and not even readable by the Registry.

987

988

6.6.4 Attribute mimeType

989

Each ExtrinsicObject instance may have a mimeType attribute defined. The

990

mimeType provides information on the type of repository item catalogued by the ExtrinsicObject instance.

991

992

993

-

994

Note that methods inherited from the base interfaces of this interface are not shown.

995

996

7.6 Interface IntrinsicObject

997

All Superinterfaces:

998

RegistryEntry, ObjectRegistryObject, Versionable

999

All Known Subinterfaces:

1000

Association, Classification, ClassificationNode, ExternalLink, Organization,

1001

Package, ExternalIdentifier

1002

1003

IntrinsicObject serve as a common base eClass for derived eClasses that

1004

catalogue submitted content whose type is known to the Registry and defined by

1005

the ebXML rRegistry specifications.

1006

1007

This interface currently does not define any attributes or methods. Note that

1008

methods inherited from the base interfaces of this interface are not shown.

1009

1010 **7.76.7 InterfaceClass Package**

1011 **All Super Classes/interfaces:**

1012 IntrinsicObject, RegistryEntry, ObjectRegistryObject, Versionable

1013
 1014 Packages allow for grouping of logically related registry
 1015 entriesRegistryEntriesRegistryObject instances may be grouped into a
 1016 Packageeven if individual member objects belong to different Submitting
 1017 Organizations. It is anticipated that Registry Services will allow operations to be
 1018 performed on an entire Package of objects in the future.

1019 **6.7.1 Attribute Summary**

1020
 1021 The Package class defines no new attributes other than those that are inherited
 1022 from RegistryEntry and RegistryObject base classes. The inherited attributes are
 1023 not shown here.

1024 **6.7.2 Method Summary**

1025 In addition to its attributes, the Package class also defines the following methods.

1026

Method Summary of Package	
Collection	getMemberObjects() Get the collection of RegistryEntriesRegistryObject instances that are members of this Package. Maps to attribute named memberObjects.

1027

1028 **7.86.8 InterfaceClass ExternalIdentifier**

1029 **All Super Classes/interfaces:**

1030 IntrinsicObject, RegistryEntry, ObjectRegistryObject, Versionable

1031
 1032 ExternalIdentifier instancesinstances provide the additional identifier information
 1033 to RegistryEntryRegistryObject such as DUNS number, Social Security Number,
 1034 or an alias name of the organization. The attribute name inherited from
 1035 ObjectRegistryObject is used to contain the identification scheme (Social
 1036 Security Number, etc), and the attribute value contains the actual information.
 1037 Each RegistryEntryRegistryObject may contain have 0 or more association(s)
 1038 with ExternalIdentifier instances.

1039 **6.8.1 Attribute Summary**

1040

Attribute	Data Type	Required	Default	Specified	Mutable
-----------	-----------	----------	---------	-----------	---------

			Value	By	
registryObject	UUID	Yes		Client	No
value	ShortName	Yes		Client	Yes

1041 **See Also:**

1042

1043 -

1044 Note that attributesmethods inherited from the base interfacesclasses of this
1045 interfaceclass are not shown.

1046 6.8.2 Attribute registryObject

1047 Each ExternalIdentifier instance must have a RegistryObject attribute that
1048 references the parent RegistryObject for which this is an ExternalIdentifier.

1049 6.8.3 Attribute value

1050 Each ExternalIdentifier instance must have a value attribute which provides the
1051 identifier value for this ExternalIdentifier (e.g. social security number).

1052 6.8.4 Inherited Attribute id

1053 The id attribute for an ExternalIdentifier is an attribute based id composed of the
1054 value of the registryObject and the name attributes in that order, where each
1055 attribute value is separated by a ‘.’.

1056

1057 The pattern is as follows:

1058 urn:uuid:<RegistryObject id>:<name>

1059

1060 An example is as follows:

1061

1062 urn:uuid:a2345678-1234-1234-123456789012:Social Security Number

1063 6.8.5 Inherited Attribute name

1064 An ExternalIdentifier instance for a RegistryObject instance must have a unique
1065 name among all other ExternalIdentifier instances for that RegistryObject
1066 instance.

1067 7.9.6.9 InterfaceClass ExternalLink

1068 **All Super Classes/interfaces:**

1069 IntrinsicObject, RegistryEntry, ObjectRegistryObject, Versionable

1070

1071 ExternalLinks use URIs to associate content in the #Registry with content that
1072 may reside outside the #Registry. For example, an organization submitting a
1073 *DTD* could use an ExternalLink to associate the *DTD* with the organization's
1074 home page.

1075 **6.9.1 Attribute Summary**

1076

<u>Attribute</u>	<u>Data Type</u>	<u>Required</u>	<u>Default Value</u>	<u>Specified By</u>	<u>Mutable</u>
<u>externalURI</u>	<u>URI</u>	<u>Yes</u>		<u>Client</u>	<u>Yes</u>

1077

1078 Note that attributes inherited from the base classes of this class are not shown.1079 **6.9.2 Attribute externalURI**

1080 Each ExternalLink instance must have an externalURI attribute defined. The
 1081 externalURI attribute provides a URI to the external resource pointed to by this
 1082 ExternalLink instance.

1083 **6.9.3 Method Summary**

1084 In addition to its attributes, the ExternalLink class also defines the following
 1085 methods.

1086

1087

Method Summary of ExternalLink	
Collection	<u>getLinkedObjects ()</u> Gets the collection of <u>objectRegistryObjects</u> that <u>are linked</u> <u>by this</u> <u>ExternalLink to content outside the registryuse this external link.</u> Maps to attribute named linkedObjects.

1088

1089 Note that methods inherited from the base interfacesclasses of this interfaceclass
 1090 are not shown.

1091 **87 Registry Audit Trail**

1092 This chaptersection describes the information model eElements that support the
 1093 audit trail capability of the *Registry*. Several eClasses in this chaptersection are
 1094 eEntity eClasses that are used as wrappers to model a set of related attributes.
 1095 These eEntity eClasses do not have any associated behavior. They are
 1096 analogous to the “struct” construct in the C programming language.

1097

1098 The getAuditTrail() method of a RegistryEntryRegistryObject returns an ordered
 1099 Collection of AuditableEvents. These AuditableEvents constitute the audit trail for
 1100 the RegistryEntryRegistryObject. AuditableEvents include a timestamp for the
 1101 eEvent. Each AuditableEvent has a reference to a User identifying the specific
 1102 user that performed an action that resulted in an AuditableEvent. Each User is
 1103 affiliated with an Organization, which is usually the sSubmitting Organization.

1104 8.17.1 InterfaceClass AuditableEvent

1105 All Superinterfac Classes:

1106 ObjectRegistryObject

1107

1108 AuditableEvent #instancesinstances provide a long-term record of eEvents that
1109 effect a change of state in a RegistryEntryRegistryObject. A

1110 RegistryEntryRegistryObject is associated with an ordered Collection of

1111 AuditableEvent #instancesinstances that provide a complete audit trail for that

1112 ObjectRegistryObject.

1113

1114 AuditableEvents are usually a result of a client-initiated request. AuditableEvent

1115 #instancesinstances are generated by the Registry Sservice to log such eEvents.

1116

1117 Often such eEvents effect a change in the life cycle of a

1118 RegistryEntryRegistryObject. For example a client request could Create, Update,

1119 Deprecate or Delete a RegistryEntryRegistryObject. No AuditableEvent is

1120 created for requests that do not alter the state of a RegistryEntryRegistryObject.

1121 Specifically, read-only requests do not generate an AuditableEvent. No

1122 AuditableEvent is generated for a RegistryEntryRegistryObject when it is

1123 classified, assigned to a Package or associated with another

1124 ObjectRegistryObject.

1125 7.1.1 Attribute Summary

1126

<u>Attribute</u>	<u>Data Type</u>	<u>Required</u>	<u>Default Value</u>	<u>Specified By</u>	<u>Mutable</u>
<u>eventType</u>	<u>LongName</u>	<u>Yes</u>		<u>Registry</u>	<u>No</u>
<u>registryObject</u>	<u>UUID</u>	<u>Yes</u>		<u>Registry</u>	<u>No</u>
<u>timestamp</u>	<u>Timestamp</u>	<u>Yes</u>		<u>Registry</u>	<u>No</u>
<u>user</u>	<u>UUID</u>	<u>Yes</u>		<u>Registry</u>	<u>No</u>

1127

1128 Note that attributes inherited from the base classes of this class are not shown.

1129 7.1.2 Attribute eventType

1130 Each AuditableEvent must have an eventType attribute which identifies the type

1131 of event recorded by the AuditableEvent.

1132

1133 8.1.17.1.2.1 Pred-defined Auditable Event Types

1134 The following table lists pre-defined auditable eevent types. These pre-defined

1135 eevent types are defined as a pre-defined ClassificationScheme tionScheme tionScheme with

1136 name "EventType".- While this scheme may easily be extended, a Registry
 1137 mustMUST support the event types listed below.
 1138

Name	description
Created	An event that created a RegistryEntryRegistryObject.
Deleted	An event that deleted a RegistryEntryRegistryObject.
Deprecated	An event that deprecated a RegistryEntryRegistryObject.
Updated	An event that updated the state of a RegistryEntryRegistryObject.
Versioned	An event that versioned a RegistryEntryRegistryObject.

1139 7.1.3 Attribute RegistryObject

1140 Each AuditableEvent must have a registryObject attribute that identifies the
 1141 RegistryObject instance that was affected by this event.

1142 7.1.4 Attribute timestamp

1143 Each AuditableEvent must have a timestamp attribute that records the date and
 1144 time that this event occurred.

1145 7.1.5 Attribute user

1146 Each AuditableEvent must have a timestamp attribute that identifies the User that
 1147 sent the request that generated this event affecting the RegistryObject instance.-

Method Summary of AuditableEvent

1148 -

1149 Note that methods inherited from the base interfaces of this interface are not
 1150 shown.

1151 8.27.2 InterfaceClass User

1152 All Super interfacesClasses:
 1153 ObjectRegistryObject

1154
 1155 User instancesinstances are used in an AuditableEvent to keep track of the
 1156 identity of the requestor that sent the request that generated the AuditableEvent.

1157 7.2.1 Attribute Summary

1158

Attribute	Data Type	Required	Default	Specified	Mutable
-----------	-----------	----------	---------	-----------	---------

			Value	By	
address	PostalAddress	Yes		Client	Yes
email	LongName	Yes		Client	Yes
organization	UUID	Yes		Client	No
personName	PersonName	Yes		Client	No
telephoneNumbers	Collection of TelephoneNumber	Yes		Client	Yes
url	URI	No		Client	Yes

1159

1160

1161

Note that attributes inherited from the base classes of this class are not shown.

1162

7.2.2 Attribute address

1163

1164

Each User instance must have an address attribute that provides the postal address for that user.

1165

7.2.3 Attribute email

1166

1167

Each User instance must have an email attribute that provides the email address for that user.

1168

7.2.4 Attribute organization

1169

1170

Each User instance must have an organization attribute that references the Organization instance for the organization that the user is affiliated with.

1171

7.2.5 Attribute personName

1172

1173

Each User instance must have a personName attribute that provides the human name for that user.

1174

7.2.6 Attribute telephoneNumbers

1175

1176

1177

Each User instance must have a telephoneNumbers attribute that contains the Collection of TelephoneNumber instances for each telephone number defined for that user.

1178

7.2.7 Attribute url

1179

1180

1181

Each User instance may have a url attribute that provides the URL address for the web page associated with that user.

1182

-

1183

8.37.3 InterfaceClass Organization

1184

All Super Cinterfaseslasses:

OASIS/ebXML Registry Information Model

1185 ~~IntrinsicObject, RegistryEntry, ObjectRegistryObject, Versionable~~

1186

1187 Organization ~~#instancesinstances~~ provide information on organizations such as a
 1188 *Submitting Organization*. Each Organization ~~#Instance~~ may have a reference to a
 1189 parent Organization.

1190 7.3.1 Attribute Summary

1191

<u>Attribute</u>	<u>Data Type</u>	<u>Required</u>	<u>Default Value</u>	<u>Specified By</u>	<u>Mutable</u>
<u>address</u>	<u>PostalAddress</u>	<u>Yes</u>		<u>Client</u>	<u>Yes</u>
<u>parent</u>	<u>UUID</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>
<u>primaryContact</u>	<u>UUID</u>	<u>Yes</u>		<u>Client</u>	<u>No</u>
<u>telephoneNumbers</u>	<u>Collection of TelephoneNumber</u>	<u>Yes</u>		<u>Client</u>	<u>Yes</u>

1192

1193 Note that attributes inherited from the base classes of this class are not shown.

1194 7.3.2 Attribute address

1195 Each Organization instance must have an address attribute that provides the
 1196 postal address for that organization.

1197 7.3.3 Attribute parent

1198 Each Organization instance may have a parent attribute that references the
 1199 parent Organization instance, if any, for that organization.

1200 7.3.4 Attribute primaryContact

1201 Each Organization instance must have a primaryContact attribute that references
 1202 the User instance for the user that is the primary contact for that organization.

1203 7.3.5 Attribute telephoneNumbers

1204 Each Organization instance must have a telephoneNumbers attribute that
 1205 contains the Collection of TelephoneNumber instances for each telephone
 1206 number defined for that organization. In addition it may have a contact attribute
 1207 defining the primary contact within the organization. An Organization also has an
 1208 address attribute.

1209 **See Also:**

1210

1211 -

1212 ~~Note that methods inherited from the base interfaces of this interface are not~~
 1213 ~~shown.~~

1214

1215 **7.4 Class PostalAddress**1216 **Class PostalAddress**

1217

1218

1219

1220

1221 PostalAddress is a simple reusable eEntity eClass that defines attributes of a
1222 postal address.

1223 **7.4.1 Attribute Summary**

1224

<u>Attribute</u>	<u>Data Type</u>	<u>Required</u>	<u>Default Value</u>	<u>Specified By</u>	<u>Mutable</u>
<u>City</u>	<u>ShortName</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>
<u>country</u>	<u>ShortName</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>
<u>postalCode</u>	<u>ShortName</u>	<u>No</u>		<u>Client</u>	<u>No</u>
<u>state</u>	<u>ShortName</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>
<u>street</u>	<u>ShortName</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>

1225

1226 **7.4.2 Attribute city**

1227 Each PostalAddress may have a city attribute identifying the city for that address.

1228 **7.4.3 Attribute country**

1229 Each PostalAddress may have a country attribute identifying the country for that
1230 address.

1231 **7.4.4 Attribute postalCode**

1232 Each PostalAddress may have a postalCode attribute identifying the postal code
1233 (e.g. zip code) for that address.

1234 **7.4.5 Attribute state**

1235 Each PostalAddress may have a state attribute identifying the state, province or
1236 region for that address.

1237 **7.4.6 Attribute street**

1238 Each PostalAddress may have a street attribute identifying the street address for
1239 that address.

1240 **7.4.7 Method Summary**

1241 In addition to its attributes, the PostalAddress class also defines the following
 1242 methods.

1243

Method Summary of ExternalLink	
<u>Collection</u>	<u>getSlots()</u> Gets the collection of Slots for this object. Each PostalAddress may have multiple Slot instances where a Slot is a dynamically defined attribute. The use of Slots allows the client to extend PostalAddress class by defining additional dynamic attributes using slots to handle locale specific needs.

1244

1245

1246 -

1247 **8.57.5 Class TelephoneNumber**

1248

1249

1250

1251 A simple reusable eEntity eClass that defines attributes of a telephone number.

1252 **7.5.1 Attribute Summary**

1253

<u>Attribute</u>	<u>Data Type</u>	<u>Required</u>	<u>Default Value</u>	<u>Specified By</u>	<u>Mutable</u>
<u>areaCode</u>	<u>String4</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>
<u>countryCode</u>	<u>String4</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>
<u>extension</u>	<u>String8</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>
<u>number</u>	<u>String8</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>
<u>phoneType</u>	<u>LongName</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>
<u>url</u>	<u>URI</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>

1254

1255

1256 **7.5.2 Attribute areaCode**

1257 Each TelephoneNumber instance may have an areaCode attribute that provides
 1258 the area code for that telephone number.

1259 **7.5.3 Attribute countryCode**

1260 Each TelephoneNumber instance may have an countryCode attribute that
 1261 provides the country code for that telephone number.

1262 **7.5.4 Attribute extension**

1263 Each TelephoneNumber instance may have an extension attribute that provides
 1264 the extension number, if any, for that telephone number.

1265 **7.5.5 Attribute number**

1266 Each TelephoneNumber instance may have an number attribute that provides
 1267 the local number (without area code, country code and extension) for that
 1268 telephone number.

1269 **7.5.6 Attribute phoneType**

1270 Each TelephoneNumber instance may have an areaCode attribute that provides
 1271 the area code for that telephone number.

1272 **7.5.7 Attribute url**

1273 Each TelephoneNumber instance may have a url attribute that provides the url, if
 1274 any, associated with that telephone number. It is an anticipated that it will be
 1275 possible to dial telephone numbers via URLs sometime in the future. Do we need
 1276 this or should we remove it??-

1277 **8.67.6 Class PersonName**

1278

 1279 A simple eEntity eClass for a person's name.

1280 **7.6.1 Attribute Summary**

1281

<u>Attribute</u>	<u>Data Type</u>	<u>Required</u>	<u>Default Value</u>	<u>Specified By</u>	<u>Mutable</u>
<u>firstName</u>	<u>ShortName</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>
<u>lastName</u>	<u>ShortName</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>
<u>middleName</u>	<u>ShortName</u>	<u>No</u>		<u>Client</u>	<u>Yes</u>

1282

1283 **7.6.2 Attribute firstName**

1284 Each PersonName may have a firstName attribute that is the first name of the
 1285 person.

1286 **7.6.3 Attribute lastName**

1287 Each PersonName may have a lastName attribute that is the last name of the
 1288 person.

1289 **7.6.4 Attribute middleName**

1290 Each PersonName may have a middleName attribute that is the middle name of
 1291 the person.

1292

1293

1294 **98 Registry-ObjectEntry Naming**

1295 A RegistryEntryRegistryObject has a name that may or may not be unique within
 1296 the *Registry*.

1297

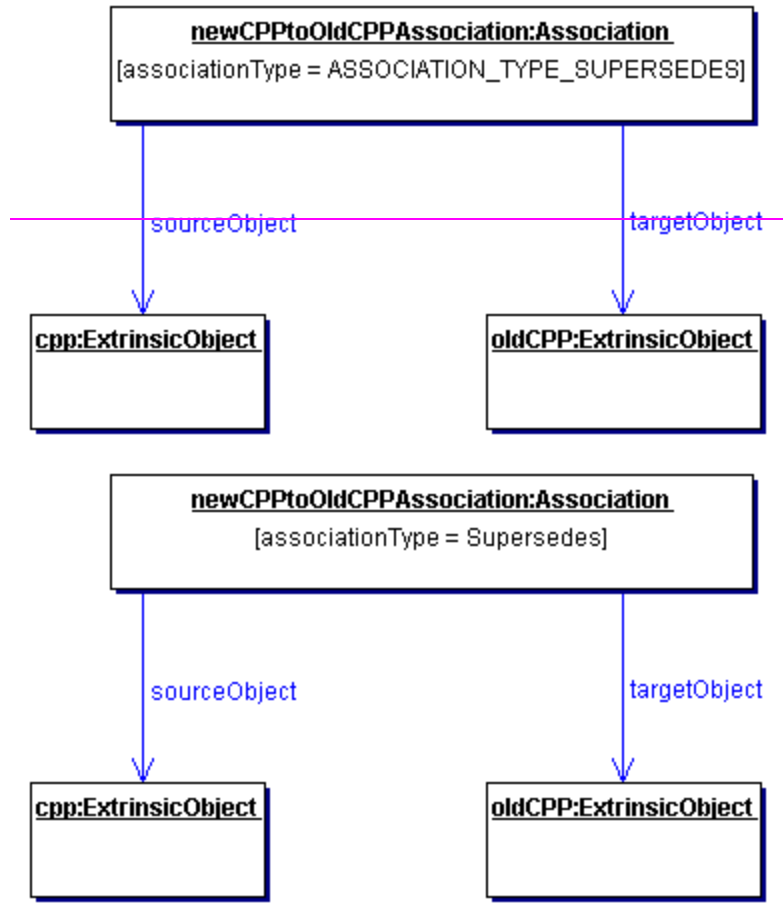
1298 In addition a RegistryEntryRegistryObject may have any number of context
 1299 sensitive alternate names that are valid only in the context of a particular
 1300 Classification scheme. Alternate contextual naming will be addressed in a later
 1301 version of the Registry Information Model.

1302 <FSN: We should remove this chapter??>

1303

1304 **109 Association of Registry-Entries**

1305 A RegistryEntryRegistryObject may be associated with 0 or more
 1306 objectRegistryObjects. The information model defines an Association Class. An
 1307 Instance of the Association Class represents an association between a source
 1308 RegistryEntryRegistryObject and another target ObjectRegistryObject. An
 1309 example of such an association is between an ExtrinsicObject instances that
 1310 catalogues a new *Collaboration Protocol Profile (CPP)* and another
 1311 ExtrinsicObject instance that catalogues an older *Collaboration Protocol Profile*
 1312 where the newer *CPP* supersedes the older *CPP* as shown in Figure 3Figure
 1313 3Figure 3Figure 3Figure 3.



1314

1315

1316

1317

Figure 333: Example of Registry-Entry Association

1318 **10.19.1 InterfaceClass Association**

1319 All Super interfacesClasses:

1320 [IntrinsicObject](#), [RegistryEntry](#), [ObjectRegistryObject](#), [Versionable](#)

1321 All Known Subinterfaces:

1322 [Classification](#)

1323

1324

1325 Association [instances](#) are used to define many-to-many associations
 1326 between [ObjectRegistryObjects](#) in the information model.

1327

1328 An *instance* of the Association [class](#) represents an association between two
 1329 [ObjectRegistryObjects](#).

1330

9.1.1 Attribute Summary

1331

<u>Attribute</u>	<u>Data Type</u>	<u>Required</u>	<u>Default Value</u>	<u>Specified By</u>	<u>Mutable</u>
<u>associationType</u>	<u>LongName</u>	<u>Yes</u>		<u>Client</u>	<u>No</u>
<u>sourceObject</u>	<u>UUID</u>	<u>Yes</u>		<u>Client</u>	<u>No</u>
<u>targetObject</u>	<u>UUID</u>	<u>Yes</u>		<u>Client</u>	<u>No</u>

1332

1333

Note that attributes inherited from the base classes of this class are not shown.

1334

9.1.2 Attribute associationType

1335

1336

1337

Each Association must have an associationType attribute that identifies the type of that association. This MUST be the name attribute of an association type as defined by 1.1.1.

1338

9.1.2.1 Pre-defined Association Types

1339

1340

1341

1342

The following table lists pre-defined association types. These pre-defined association types are defined as a *Classification* scheme. While the scheme may easily be extended a *Registry* MUST support the association types listed below.

<u>name</u>	<u>description</u>
<u>RelatedTo</u>	<u>Defines that source RegistryObject is related to target RegistryObject.</u>
<u>HasMember</u>	<u>Defines that the source Package object has the target RegistryObject object as a member. Reserved for use in Packaging of RegistryEntries.</u>
<u>ExternallyLinks</u>	<u>Defines that the source ExternalLink object externally links the target RegistryObject object. Reserved for use in associating ExternalLinks with RegistryEntries.</u>
<u>ExternallyIdentifies</u>	<u>Defines that the source ExternalIdentifier object identifies the target RegistryObject object. Reserved for use in associating ExternalIdentifiers with RegistryEntries.</u>
<u>ContainedBy</u>	<u>Defines that source RegistryObject is contained by the target RegistryObject.</u>
<u>Contains</u>	<u>Defines that source RegistryObject contains the target RegistryObject.</u>
<u>Extends</u>	<u>Defines that source RegistryObject inherits from or specializes the target RegistryObject.</u>
<u>Implements</u>	<u>Defines that source RegistryObject implements the functionality defined by the target RegistryObject.</u>
<u>InstanceOf</u>	<u>Defines that source RegistryObject is an <i>Instance</i> of</u>

	<u>target RegistryObject.</u>
<u>SupersededBy</u>	<u>Defines that the source RegistryObject is superseded by the target RegistryObject.</u>
<u>Supersedes</u>	<u>Defines that the source RegistryObject supersedes the target RegistryObject.</u>
<u>UsedBy</u>	<u>Defines that the source RegistryObject is used by the target RegistryObject in some manner.</u>
<u>Uses</u>	<u>Defines that the source RegistryObject uses the target RegistryObject in some manner.</u>
<u>ReplacedBy</u>	<u>Defines that the source RegistryObject is replaced by the target RegistryObject in some manner.</u>
<u>Replaces</u>	<u>Defines that the source RegistryObject replaces the target RegistryObject in some manner.</u>

1343

-

1344 **9.1.3 Attribute sourceObject**

1345 Each Association must have a sourceObject attribute that references the
 1346 RegistryObject instance that is the source or owner of that association.

1347 **9.1.4 Attribute targetObject**

1348 Each Association must have an targetObject attribute that references the
 1349 RegistryObject instance that is the target of that association.

1350 **9.1.5 Inherited Attribute id**

1351 The id attribute for an Association is an attribute based id composed of the value
 1352 of the sourceObject, targetObject and associationType attributes in that order,
 1353 where each attribute value is separated by a ':'.

1354

1355 The pattern is as follows:1356 urn:uuid:< sourceObject id>:< targetObject id>:<associationType>

1357

1358 An example is as follows:

1359

1360 urn:uuid:a2345678-1234-1234-123456789012: a2345678-1234-1234-1361 123456789013:Implements

1362

1363 -

1364 **10.1.1 Pre-defined Association Types**

1365 The following table lists pre-defined association types. These pre-defined
 1366 association types are defined as a Classification scheme. While the scheme may

OASIS/ebXML Registry Information Model

Page 5

1367
1368
1369

easily be extended a Registry must MUST support the association types listed below.

name	description
RelatedTo	Defines that source <u>objectRegistryObject</u> is an <u>related to instance of target objectRegistryObject</u> .
PHasMemberackages	Defines that the source Package object <u>haspackages</u> the target RegistryEntry object as a member. Reserved for use in Packaging of Registry Entries.
ExternallyLinks	Defines that the source ExternalLink object <u>externally links</u> the target RegistryEntry object. Reserved for use in associating ExternalLinks with Registry Entries.
ExternallyIdentifies	Defines that the source ExternalIdentifier object <u>identifies</u> the target RegistryEntry object. Reserved for use in associating ExternalIdentifiers with Registry Entries.
ContainedBy	Defines that source <u>objectRegistryObject</u> is contained by the target <u>objectRegistryObject</u> .
Contains	Defines that source <u>objectRegistryObject</u> contains the target <u>objectRegistryObject</u> .
Extends	Defines that source <u>objectRegistryObject</u> inherits from or specializes the target <u>objectRegistryObject</u> .
Implements	Defines that source <u>objectRegistryObject</u> implements the functionality defined by the target <u>objectRegistryObject</u> .
InstanceOf	Defines that source <u>objectRegistryObject</u> is an <u>iinstance</u> of target <u>objectRegistryObject</u> .
SupersededBy	Defines that the source <u>objectRegistryObject</u> is superseded by the target <u>objectRegistryObject</u> .
Supersedes	Defines that the source <u>objectRegistryObject</u> supersedes the target <u>objectRegistryObject</u> .
UsedBy	Defines that the source <u>objectRegistryObject</u> is used by the target <u>objectRegistryObject</u> in some manner.
Uses	Defines that the source <u>objectRegistryObject</u> uses the target <u>objectRegistryObject</u> in some manner.
ReplacedBy	Defines that the source <u>objectRegistryObject</u> is replaced by the target <u>objectRegistryObject</u> in some manner.
Replaces	Defines that the source <u>objectRegistryObject</u> replaces the target <u>objectRegistryObject</u> in some manner.

1370
1371

In some association types, such as Extends and

1372 ~~Implements, although the association is between~~
 1373 ~~RegistryObjects, the actual relationship~~
 1374 ~~specified by that type is between repository~~
 1375 ~~items pointed by RegistryObjects.~~

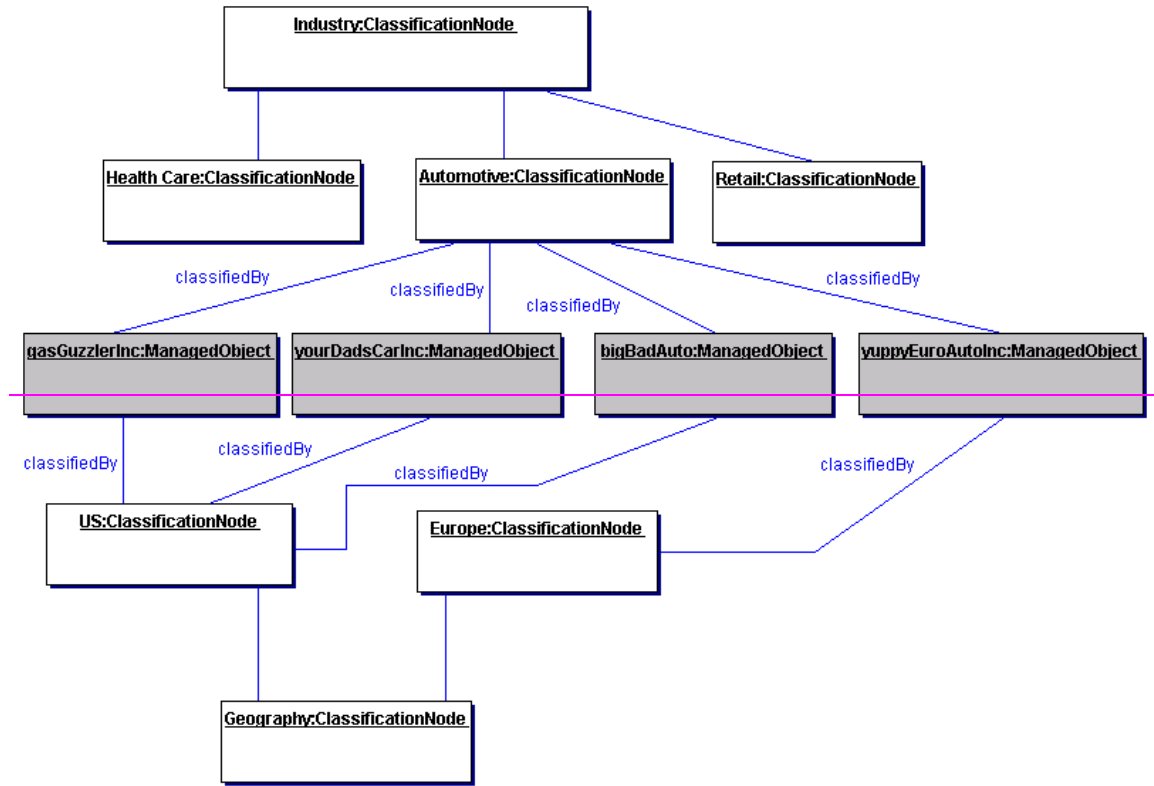
1376 **1110 Classification of Registry Object Entries**

1377 This section describes the how the information model supports ~~e~~Classification of
 1378 ~~RegistryEntryRegistryObject~~s. It is a simplified version of the OASIS
 1379 classification model [OAS].

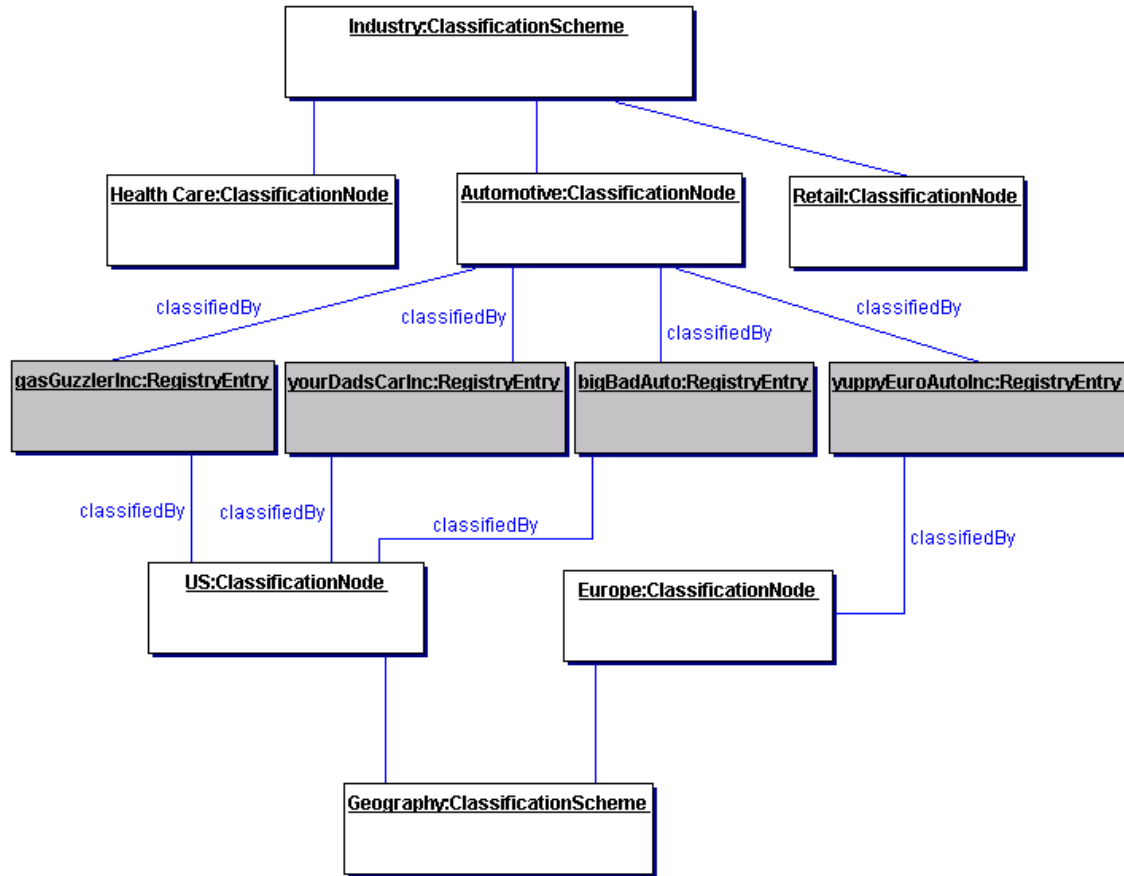
1380
 1381 A ~~RegistryEntryRegistryObject~~ may be classified in many ways. For example the
 1382 ~~RegistryEntryRegistryObject~~ for the same *Collaboration Protocol Profile (CPP)*
 1383 may be classified by its industry, by the products it sells and by its geographical
 1384 location.

1385
 1386 A general ~~e~~Classification scheme can be viewed as a ~~e~~Classification tree. In the
 1387 example shown in ~~Figure 4Figure 4Figure 4Figure 4Figure 4~~,
 1388 ~~RegistryEntriesRegistryObject~~ instances representing *Collaboration Protocol*
 1389 *Profiles* are shown as shaded boxes. Each *Collaboration Protocol Profile*
 1390 represents an automobile manufacturer. Each *Collaboration Protocol Profile* is
 1391 classified by the ClassificationNode named "Automotive" under the ~~root~~
 1392 ~~ClassificationNodeClassificationScheme~~ instance with name "Industry".
 1393 Furthermore, the US Automobile manufacturers are classified by the US
 1394 ClassificationNode under the ~~Geography~~ ClassificationScheme with name
 1395 "Geography"Node. Similarly, a European automobile manufacturer is classified
 1396 by the "Europe" ClassificationNode under the ~~Geography~~ ClassificationScheme
 1397 with name "Geography"Node.

1398
 1399 The example shows how a ~~RegistryEntryRegistryObject~~ may be classified by
 1400 multiple ~~e~~ClassificationNode instances under multiple ClassificationScheme
 1401 instances. A ~~e~~Classification scheme is defined by a ClassificationNode that is the
 1402 root of a ~~e~~Classification tree (e.g. Industry, Geography).



1403



1404

1405

Figure 44: Example showing a Classification Tree

1406

[Note] It is important to point out that the dark nodes (gasGuzzlerInc, yourDadsCarInc etc.) are not part of the *eClassification* tree. The leaf nodes of the *eClassification* tree are Health Care, Automotive, Retail, US and Europe. The dark nodes are associated with the *eClassification* tree via a *ClassificationInstance* that is not shown in the picture

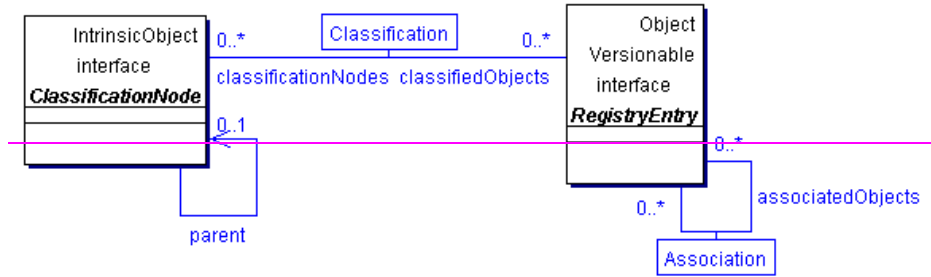
1414

1415

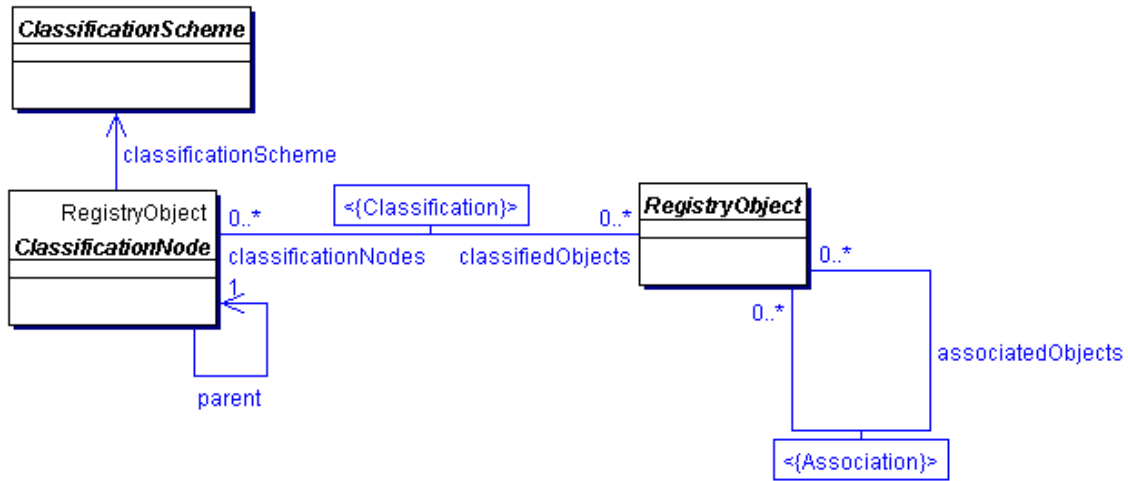
In order to support a general *eClassification* scheme that can support single level as well as multi-level *eClassifications*, the information model defines the *eClasses* and relationships shown in [Figure 5](#) [Figure 5](#) [Figure 5](#) [Figure 5](#) [Figure 5](#).

1416

1417



1418



1419

1420

Figure 55: Information Model Classification View

1421

A Classification is somewhat like a specialized form of an Association. Figure

1422

6Figure 6Figure 6Figure 6 shows an example of an ExtrinsicObject

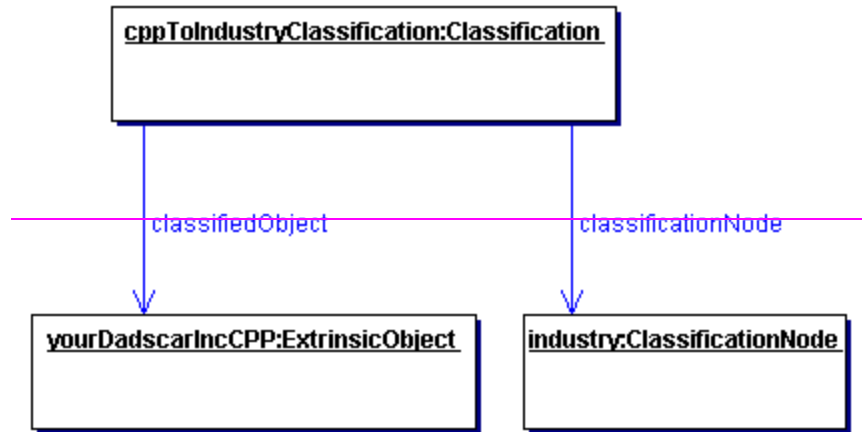
1423

Instance for a Collaboration Protocol Profile (CPP) object that is classified by a

1424

ClassificationNode representing the Industry that it belongs to.

1425



1426

1427

1428

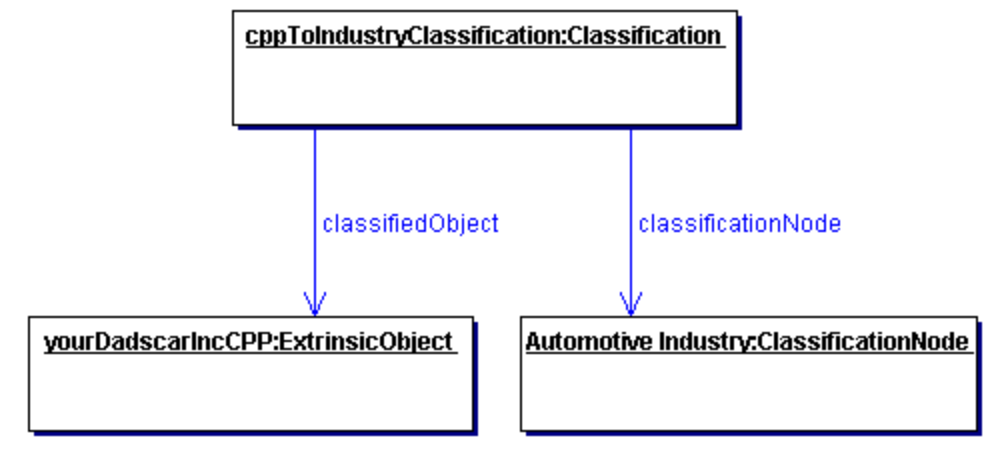


Figure 666: Classification Instance Diagram

1429 **10.1 Class ClassificationScheme**

1430 A ClassificationScheme instance defines the root of a classification hierarchy
 1431 where the nodes of the tree are composed of ClassificationNodes instances.
 1432 Currently the ClassificationScheme class does not define any attributes other
 1433 than the attributes inherited from RegistryObject and RegistryEntry base classes.
 1434

1435 <FSN: Do we need any attributes or methods for this class?? Also do we need to
 1436 change the name of GetRootClassificationNodesRequest/Response to
 1437 GetClassificationSchemeRequest/Response in the RS 1.1??>

1438 **11.1-10.2 InterfaceClass ClassificationNode**

1439 **Base All Superinterfacesclasses:**

1440 ~~IntrinsicObject, RegistryEntry, ObjectRegistryObject, Versionable~~

1441

1442 ClassificationNode ~~#instances~~instances are used to define tree structures where
 1443 each node in the tree is a ClassificationNode. Such ~~e~~Classification trees are
 1444 constructed with ClassificationNode instances under a ClassificationScheme
 1445 instance, and are used to define ~~e~~Classification schemes or ontologies.

1446 **See Also:**
 1447 [Classification](#)

1448 **10.2.1 Attribute Summary**

1449

<u>Attribute</u>	<u>Data Type</u>	<u>Required</u>	<u>Default Value</u>	<u>Specified By</u>	<u>Mutable</u>
parent	UUID	Yes		Client	No
code	ShortName	No		Client	No

1450

1451 [Note that attributes inherited from the base classes of this class are not shown.](#)

1452 **10.2.2 Attribute parent**

1453 Each ClassificationNode must have a parent attribute. The parent attribute either
 1454 references a parent ClassificationNode or a ClassificationScheme instance in
 1455 case of first level ClassificationNode instances.

1456 **10.2.3 Attribute code**

1457 Each ClassificationNode may have a code attrubite. The code attribute contains
 1458 a code within a standard coding scheme as described in section 10.5.

1459 **10.2.4 Method Summary**

1460 In addition to its attributes, the Package class also defines the following methods.

1461

<u>Method Summary of ClassificationNode</u>	
ClassificationScheme	getClassificationScheme() Get the ClassificationScheme that this this ClassificationNode belongs to.
Collection	getClassifiedObjects() Get the collection of RegistryObjectsEntries classified by this ClassificationNode. Maps to attribute named classifiedObjects.
String	getPath() Gets the path from the ClassificationScheme root ancestor of this ClassificationNode. The path conforms to the [XPATH] expression syntax (e.g "/Geography/Asia/Japan"). Maps to attribute named path.

1462

1463 Note that methods inherited from the base [interfacesclasses](#) of this [interfaceclass](#)
1464 are not shown.

1465

1466 In [Figure 4Figure 4Figure 4Figure 4](#), several [instancesinstances](#) of
1467 ClassificationNode are defined (all light colored boxes). A ClassificationNode has
1468 ~~exactly zero or one ClassificationNodes for its~~ parent and zero or more
1469 ClassificationNodes for its immediate children. [The parent of a](#)
1470 [ClassificationNodes may be another ClassificationNodes or a](#)
1471 [ClassificationScheme in case of first level ClassificationNodes.If a](#)
1472 [ClassificationNode has no parent then it is the root of a eClassification tree. Note](#)
1473 [that the entire eClassification tree is recursively defined by a single information](#)
1474 [model eElement ClassificationNode.](#)
1475

1476 **11.210.3 InterfaceClass Classification**

1477 **Base CAll Superinterfaceslasses:**

1478 [IntrinsicObject](#), [RegistryEntry](#), [ObjectRegistryObject](#), [Versionable](#)

1479

1480 Classification [instancesinstances](#) are used to classify repository item by
1481 associating their [RegistryEntryRegistryObject](#) [instance](#) with a ClassificationNode
1482 [instance](#) within a [eClassification](#) scheme.

1483

1484 In [Figure 4Figure 4Figure 4Figure 4](#), Classification [instancesinstances](#)
1485 are not explicitly shown but are implied as associations between the
1486 [RegistryEntriesRegistryObject instances](#) (shaded leaf node) and the associated
1487 ClassificationNode.

1488 **10.3.1 Attribute Summary**

1489

<u>Attribute</u>	<u>Data Type</u>	<u>Required</u>	<u>Default Value</u>	<u>Specified By</u>	<u>Mutable</u>
classificationNode	UUID	Yes		Client	No
classifiedObject	UUID	Yes		Client	No

1490 [Note that attributes inherited from the base classes of this class are not shown.](#)

1491 **10.3.2 Attribute classificationNode**

1492 [Each Classification instance must have a classificationNode attribute that](#)
1493 [references the ClassificationNode instance that is used to classify a](#)
1494 [RegistryObject specified by the classifiedObject attribute. This is similar to the](#)
1495 [targetObject attribute in an Association instance.](#)

1496 **10.3.3 Attribute classifiedObject**

1497 Each Classification instance must have a classifiedObject attribute that
 1498 references the RegistryObject instance that is classified by this Classification.
 1499 This is similar to the sourceObject attribute in an Association instance.
 1500

1501 **10.3.4 Inherited Attribute id**

1502 The id attribute for a Classification is an attribute based id composed of the
 1503 value of the classifiedObject and the classificationNode attributes in that order,
 1504 where each attribute value is separated by a ':'.
 1505

1506 The pattern is as follows:

1507 urn:uuid:<classifiedObject id>:< classificationNode id>
 1508

1509 An example is as follows:

1510 urn:uuid:a2345678-1234-1234-123456789012: a2345678-1234-1234-123456789013
 1511

1512 Note that methods inherited from the base interfaces of this interface are not
 1513 shown.

1514 **11.2.110.3.5 Context Sensitive Classification**

1515 Consider the case depicted in ~~Figure 7~~~~Figure 7~~~~Figure 7~~~~Figure 7~~~~Figure 7~~ where a
 1516 *Collaboration Protocol Profile* for ACME Inc. is classified by the Japan
 1517 ClassificationNode under the Geography *eClassification* scheme. In the absence
 1518 of the context for this *eClassification* its meaning is ambiguous. Does it mean
 1519 that ACME is located in Japan, or does it mean that ACME ships products to
 1520 Japan, or does it have some other meaning? To address this ambiguity a
 1521 Classification may optionally be associated with another ClassificationNode (in
 1522 this example named isLocatedIn) that provides the missing context for the
 1523 Classification. Another *Collaboration Protocol Profile* for MyParcelService may be
 1524 classified by the Japan ClassificationNode where this Classification is associated
 1525 with a different ClassificationNode (e.g. named shipsTo) to indicate a different
 1526 context than the one used by ACME Inc.

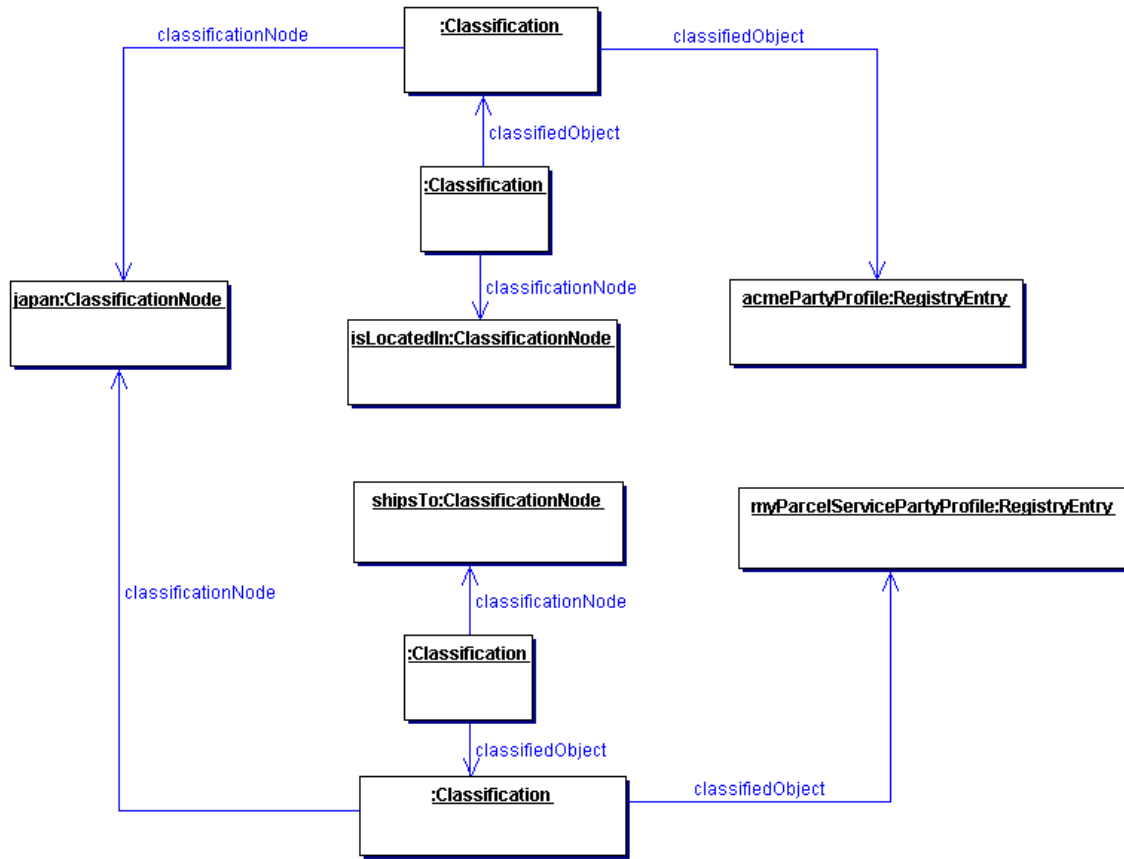


Figure 777: Context Sensitive Classification

1527

1528

1529 Thus, in order to support the possibility of Classification within multiple contexts,
 1530 a Classification is itself classified by any number of Classifications that bind the
 1531 first Classification to ClassificationNodes that provide the missing contexts.
 1532

1533 In summary, the generalized support for ϵ Classification schemes in the
 1534 information model allows:

- 1535 ○ A RegistryEntryRegistryObject to be classified by defining a Classification
- 1536 that associates it with a ClassificationNode in a ϵ Classification tree.
- 1537 ○ A RegistryEntryRegistryObject to be classified along multiple facets by
- 1538 having multiple ϵ Classifications that associate it with multiple
- 1539 ClassificationNodes.
- 1540 ○ A ϵ Classification defined for a RegistryEntryRegistryObject to be qualified
- 1541 by the contexts in which it is being classified.

1542 4.3.10.4 Example of Classification Schemes

1543 The following table lists some examples of possible ϵ Classification schemes
 1544 enabled by the information model. These schemes are based on a subset of

1545 contextual concepts identified by the ebXML Business Process and Core
 1546 Components Project Teams. This list is meant to be illustrative not prescriptive.
 1547
 1548

Classification Scheme (Context)	Usage Example	Standard Classification Schemes
Industry	Find all Parties in Automotive industry	NAICS
Process	Find a ServiceInterface that implements a Process	
Product / Services	Find a <i>Bbusiness</i> that sells a product <i>or offers a service</i>	UNSPSC
Locale	Find a Supplier located in -Japan	ISO 3166
Temporal	Find Supplier that can ship with 24 hours	
Role	Find All Suppliers that have a <i>#Role</i> of "Seller"	

1549 **Table 1: Sample Classification Schemes**

1550 **11.4.10.5 Standardized Taxonomy Support**

1551 Standardized taxonomies also referred to as ontologies, *classification schemes*,
 1552 *or-or* coding schemes exist in various industries to provide a structured coded
 1553 vocabulary. The ebXML *#Registry* does not define support for specific
 1554 taxonomies. Instead it provides a general capability to link Registry *EntryItems* to
 1555 codes defined by various taxonomies.

1556
 1557 The information model provides two alternatives for using standardized
 1558 taxonomies for *eClassification* of Registry *EntryItems*.

1559 **11.4.110.5.1 Full-featured Taxonomy Based Classification**

1560 The information model provides a full-featured taxonomy based *eClassification*
 1561 alternative based on *Classification*, *ClassificationScheme* and *ClassificationNode*
 1562 *#instancesinstances*. This alternative requires that a standard taxonomy be
 1563 imported into the *Registry* as a *eClassification* tree consisting of
 1564 *ClassificationNode #instancesinstances rooted under a ClassificationScheme*
 1565 *instance*. This specification does not prescribe the transformation tools
 1566 necessary to convert standard taxonomies into ebXML *Registry eClassification*
 1567 trees. However, the transformation *mustMUST* ensure that:

- 1568 *1-o* The name attribute of the *root*
 1569 *ClassificationNodeClassificationScheme instance* is the *name* of the
 1570 standard taxonomy (e.g. NAICS, ICD-9, SNOMED).
 1571 *2-o* All codes in the standard taxonomy are preserved in the *code*
 1572 attribute of a *ClassificationNode*.

1573 3-o The intended structure of the standard taxonomy is preserved in
 1574 the ClassificationNode tree, thus allowing polymorphic browse and drill
 1575 down discovery. This means that whenis searching for entries classified
 1576 by Asia, a client will find entries classified by descendants of Asia (e.g.
 1577 Japan and Korea).

1578 11.4.210.5.2 Light Weight Taxonomy Based Classification

1579 <FSN: This section will be reworked based on the classification sub-team
 1580 proposal??>

1581
 1582 The information model also provides a lightweight alternative for classifying
 1583 RegistryEntryRegistryObject #instancesinstances by codes defined by standard
 1584 taxonomies, where the submitter does not wish to import an entire taxonomy as a
 1585 native eClassification scheme.

1586
 1587 In this alternative the submitter adds one or more taxonomy related Slots to the
 1588 RegistryEntryRegistryObject-for-a-submitted-repository-item. Each Slot's name
 1589 identifies a standardized taxonomy while the Slot's value is the code within the
 1590 specified taxonomy. Such taxonomy related Sslots mustMUST be defined with a
 1591 slotType of Classification.

1592
 1593 For example if a RegistryEntryRegistryObject has a Slot with name "NAICS", a
 1594 slotType of "Classification" and a value "51113" it implies that the
 1595 RegistryEntryRegistryObject is classified by the code for "Book Publishers" in the
 1596 NAICS taxonomy. Note that in this example, there is no need to import the entire
 1597 NAICS taxonomy, nor is there any need to create #instancesinstances of
 1598 ClassificationNode or Classification.

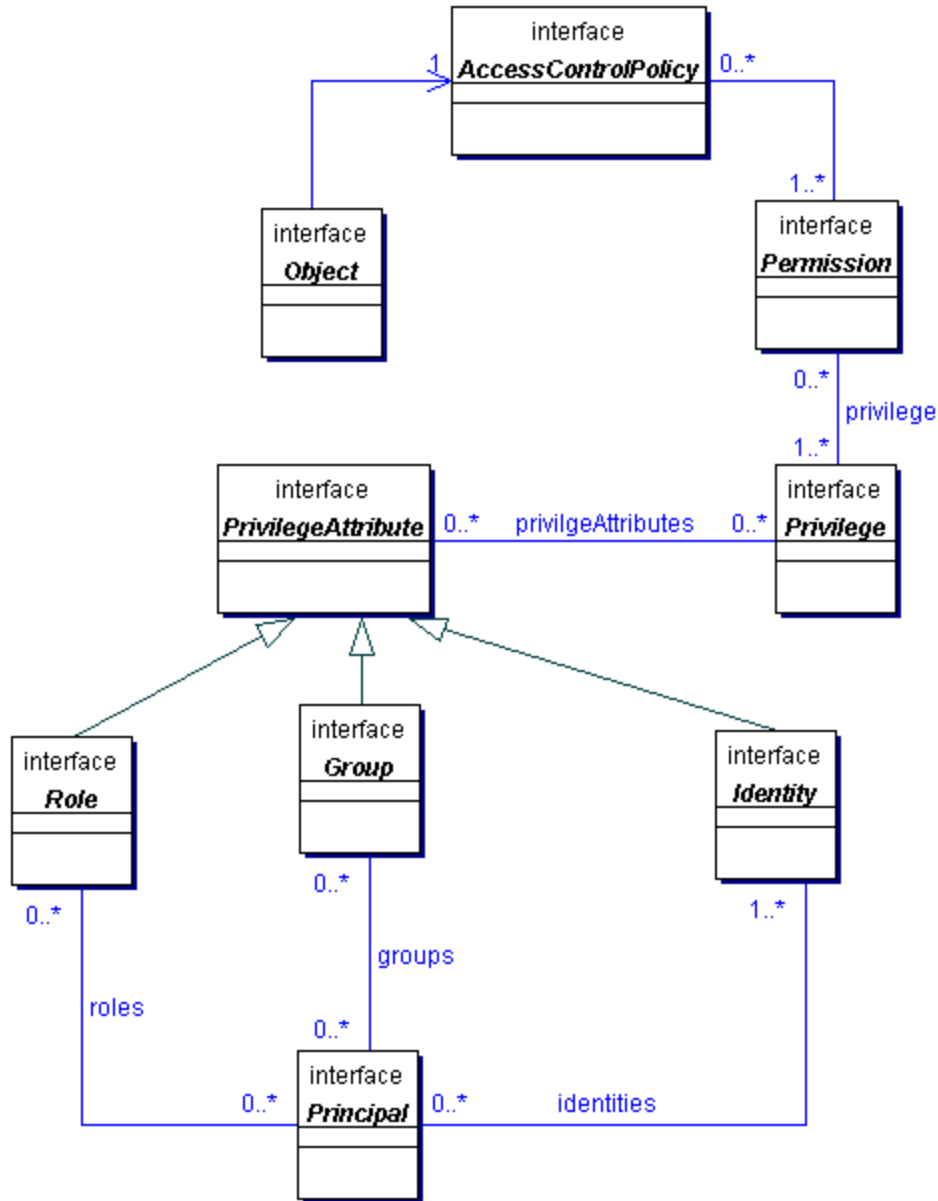
1599
 1600 The following points are noteworthy in this light weight eClassification alternative:
 1601 o Validation of the name and the value of the Classification" is responsibility
 1602 of the SO and not of the ebXML *Registry* itself.
 1603 o Discovery is based on exact match on slot name and slot value rather
 1604 than the flexible "browse and drill down discovery" available to the heavy
 1605 weight eClassification alternative.
 1606

1607 1211 Information Model: Security View

1608 This chaptersection describes the aspects of the information model that relate to
 1609 the security features of the *Registry*.

1610
 1611 <FSN: Thos chapter will be updated based on output from the security sub-
 1612 team?? It is therefore not updated in format to the new format for RIM 1.1>
 1613

1614 ~~Figure 8~~~~Figure 8~~~~Figure 8~~~~Figure 8~~~~Figure 8~~ shows the view of the objects in the
 1615 *Registry* from a security perspective. It shows object relationships as a *UML*
 1616 *Class* diagram. It does not show *Class* attributes or *Class* methods that will be
 1617 described in subsequent sections. It is meant to be illustrative not prescriptive.
 1618



1619
 1620

Figure 888: Information Model: Security View

1621 **12.111.1 InterfaceClass AccessControlPolicy**

1622 Every [ObjectRegistryObject](#) is associated with exactly one AccessControlPolicy
 1623 which defines the policy rules that govern access to operations or methods
 1624 performed on that [ObjectRegistryObject](#). Such policy rules are defined as a
 1625 collection of Permissions.

1626
 1627
 1628
 1629

Method Summary of AccessControlPolicy

Collection	getPermissions() Gets the Permissions defined for this AccessControlPolicy. Maps to attribute named <code>permissions</code> .
------------	--

1630

1631 **12.211.2 InterfaceClass Permission**

1632
 1633 The Permission object is used for authorization and access control to
 1634 [ObjectRegistryObjects](#) in the *Registry*. The Permissions for a
 1635 [ObjectRegistryObject](#) are defined in an AccessControlPolicy object.
 1636

1637 A Permission object authorizes access to a method in a [ObjectRegistryObject](#) if
 1638 the requesting Principal has any of the Privileges defined in the Permission.

1639 **See Also:**

1640 [Privilege](#), [AccessControlPolicy](#)

1641

Method Summary of Permission

String	getMethodName() Gets the method name that is accessible to a Principal with specified Privilege by this Permission. Maps to attribute named <code>methodName</code> .
Collection	getPrivileges() Gets the Privileges associated with this Permission. Maps to attribute named <code>privileges</code> .

1642

1643 **12.311.3 InterfaceClass Privilege**

1644
 1645 A Privilege object contains zero or more PrivilegeAttributes. A PrivilegeAttribute
 1646 can be a Group, a Role, or an Identity.
 1647

1648 A requesting Principal ~~must~~**MUST** have all of the PrivilegeAttributes specified in a
 1649 Privilege in order to gain access to a method in a protected
 1650 [ObjectRegistryObject](#). Permissions defined in the [ObjectRegistryObject](#)'s
 1651 AccessControlPolicy define the Privileges that can authorize access to specific
 1652 methods.

1653
 1654 This mechanism enables the flexibility to have object access control policies that
 1655 are based on any combination of Roles, Identities or Groups.

1656 **See Also:**

1657 [PrivilegeAttribute](#), [Permission](#)

1658

1659

1660

Method Summary of Privilege	
Collection	getPrivilegeAttributes() Gets the PrivilegeAttributes associated with this Privilege. Maps to attribute named <code>privilegeAttributes</code> .

1661

1662 **12.411.4 InterfaceClass PrivilegeAttribute**

1663 **All Known Subinterfacesclasses:**

1664 [Group](#), [Identity](#), [Role](#)

1665

1666
 1667 PrivilegeAttribute is a common base [eClass](#) for all types of security attributes that
 1668 are used to grant specific access control privileges to a Principal. A Principal may
 1669 have several different types of PrivilegeAttributes. Specific combination of
 1670 PrivilegeAttributes may be defined as a Privilege object.

1671 **See Also:**

1672 [Principal](#), [Privilege](#)

1673 **12.511.5 InterfaceClass Role**

1674 **All Superinterfacesclasses:**

1675 [PrivilegeAttribute](#)

1676

1677 A security Role PrivilegeAttribute. For example a hospital may have *Roles* such
 1678 as Nurse, Doctor, Administrator etc. Roles are used to grant Privileges to
 1679 Principals. For example a Doctor *rRole* may be allowed to write a prescription but
 1680 a Nurse *rRole* may not.

1681 **12.611.6 InterfaceClass Group**

1682 **All Superinterfacesclasses:**

[OASIS/ebXML Registry Information Model](#)

1683 [PrivilegeAttribute](#)

1684

1685 A security Group PrivilegeAttribute. A Group is an aggregation of users that may
 1686 have different [#Roles](#). For example a hospital may have a Group defined for
 1687 Nurses and Doctors that are participating in a specific clinical trial (e.g.
 1688 AspirinTrial group). Groups are used to grant Privileges to Principals. For
 1689 example the members of the AspirinTrial group may be allowed to write a
 1690 prescription for Aspirin (even though Nurse [#Role](#) as a rule may not be allowed to
 1691 write prescriptions).

1692 **[12.711.7](#) InterfaceClass Identity**1693 **All Super[interfacesclasses](#):**1694 [PrivilegeAttribute](#)

1695

1696 A security Identity PrivilegeAttribute. This is typically used to identify a person, an
 1697 organization, or software service. Identity attribute may be in the form of a digital
 1698 certificate.

1699 **[12.811.8](#) InterfaceClass Principal**

1700

1701 Principal is a completely generic term used by the security community to include
 1702 both people and software systems. The Principal object is an entity that has a set
 1703 of PrivilegeAttributes. These PrivilegeAttributes include at least one identity, and
 1704 optionally a set of [#role](#) memberships, group memberships or security
 1705 clearances. A principal is used to authenticate a requestor and to authorize the
 1706 requested action based on the PrivilegeAttributes associated with the Principal.

1707 **See Also:**1708 PrivilegeAttributes, [Privilege](#), [Permission](#)

1709

Method Summary of Principal	
Collection	getGroups() Gets the Groups associated with this Principal. Maps to attribute named <code>groups</code> .
Collection	getIdentities() Gets the Identities associated with this Principal. Maps to attribute named <code>identities</code> .
Collection	getRoles() Gets the Roles associated with this Principal. Maps to attribute named <code>roles</code> .

1710

1711

1711 **1312References**

- 1712 [ebGLOSSGLS]—ebXML Glossary,
 1713 [_http://www.ebxml.org/documents/199909/terms_of_reference.htm](http://www.ebxml.org/documents/199909/terms_of_reference.htm)
- 1714 [ebTA]—ebXML Technical Architecture Specification
 1715 http://www.ebxml.org/specdrafts/ebXML_TA_v1.0.4.pdf
- 1716 [OAS] OASIS Information Model
 1717 <http://xsun.sdct.itl.nist.gov/www.nist.gov/itl/div897/ctg/regrep/OasisRegrep>
 1718 [Spec.pdf](http://www.nist.gov/itl/div897/ctg/regrep/OasisRegrepSpec.pdf)
[easis-work.html](http://www.nist.gov/itl/div897/ctg/regrep/OasisRegrepSpec.pdf#easis-work.html)
- 1719 [ISO] ISO 11179 Information Model
 1720 <http://208.226.167.205/SC32/jtc1sc32.nsf/576871ad2f11bba78525662100>
 1721 [5419d7/b83fc7816a6064c68525690e0065f913?OpenDocument](http://208.226.167.205/SC32/jtc1sc32.nsf/576871ad2f11bba785256621005419d7/b83fc7816a6064c68525690e0065f913?OpenDocument)
- 1722 [BRA97] IETF (Internet Engineering Task Force). RFC 2119: Key words for use
 1723 in RFCs to Indicate Requirement Levels
 1724 <http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc2119.html>
 1725 [DM]—Registry and Repository: Business Domain Model
 1726 <http://www.ebxml.org/specdrafts/RegRepv1-0.pdf>
- 1727 [ebRS]—ebXML Registry Services Specification
 1728 http://www.ebxml.org/specdrafts/ebXML_RS_v1.0.pdf
 1729 http://www.ebxml.org/project_teams/registry/private/RegistryServicesSpec
 1730 [ificationv0.83.pdf](http://www.ebxml.org/project_teams/registry/private/RegistryServicesSpecificationv0.83.pdf)
- 1731 [BPMebBPSS]—ebXML Business Process Metamodel Specification Schema
 1732 <http://www.ebxml.org/specdrafts/Busv2-0.pdf>
- 1733 [ebCPPCPA]—ebXML Collaboration-Protocol Profile and Agreement
 1734 Trading Partner Specification
 1735 http://www.ebxml.org/specdrafts/project_teams/trade_partner/private/
 1736
- 1737 [CTB] Context table informal document from Core Components
 1738 http://www.ebxml.org/project_teams/core_components/ContextTable.doc
- 1739 [UUID] DCE 128 bit Universal Unique Identifier
 1740 http://www.opengroup.org/onlinepubs/009629399/apdxa.htm#tagcjh_20
 1741 <http://www.opengroup.org/publications/catalog/c706.htm>
 1742 <http://www.w3.org/TR/REC-xml>
- 1743 [XPATH] XML Path Language (XPath) Version 1.0
 1744 <http://www.w3.org/TR/xpath>

1745

1746 **1413Disclaimer**

1747 The views and specification expressed in this document are those of the authors
1748 and are not necessarily those of their employers. The authors and their
1749 employers specifically disclaim responsibility for any problems arising from
1750 correct or incorrect implementation or use of this design.

1751

1751 **1514 Contact Information**

1752

1753 Team Leader

1754 Name:

[Lisa Carnahan](#)[Scott Nieman](#)

1755 Company:

[NIST](#)[Norstan Consulting](#)

1756 Street:

[100 Bureau Drive STOP 8970 5101 Shady](#)1757 [Oak Road](#)

1758 City, State, Postal Code:

[Gaithersburg, MD 20899-8970 Minnetonka,](#)1759 [MN-55343](#)

1760 Country:

USA

1761 Phone:

[301-975-3362 952-352-5889](#)

1762 Email:

[lisa.carnahan@nist.gov](#)[Scott.Nieman@Norstan](#)

1763

1764 [Editor](#) [Vice Team Lead](#)

1765 Name:

[Sally Fuger](#)[Yutaka Yoshida](#)

1766 Company:

[<Need Sally's contact info??>Sun](#)1767 [Microsystems](#)

1768 Street:

[901 San Antonio Road, MS UMPK17-102](#)

1769 City, State, Postal Code:

[Palo Alto, CA 94303](#)

1770 Country:

USA

1771 Phone:

[650.786.5488](#)

1772 Email:

[Yutaka.Yoshida@eng.sun.com](#)1773 [sfuger@aiag.org](#)

1774

1775 [Technical](#) Editor

1776 Name:

Farrukh S. Najmi

1777 Company:

Sun Microsystems

1778 Street:

1 Network Dr., MS BUR02-302

1779 City, State, Postal Code:

Burlington, MA, 01803-0902

1780 Country:

USA

1781 Phone:

781.442.0703

1782 Email:

[najmi@east.sun.com](#)

1783

1784

1784 **Copyright Statement**1785 Copyright © UN/CEFACT and OASIS, 2001. All Rights Reserved1786 Copyright © ebXML 2000 & 2001 All Rights Reserved.

1787

1788 This document and translations of it MAY be copied and furnished to others, and
1789 derivative works that comment on or otherwise explain it or assist in its
1790 implementation MAY be prepared, copied, published and distributed, in whole or
1791 in part, without restriction of any kind, provided that the above copyright notice
1792 and this paragraph are included on all such copies and derivative works.

1793 However, this document itself MAY not be modified in any way, such as by
1794 removing the copyright notice or references to ebXML, UN/CEFACT, or OASIS,
1795 except as required to translate it into languages other than English.

1796

1797 The limited permissions granted above are perpetual and will not be revoked by
1798 ebXML or its successors or assigns.

1799

1800 This document and the information contained herein is provided on an "AS IS"
1801 basis and ebXML DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED,
1802 INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE
1803 INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
1804 WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR
1805 PURPOSE.

1806 Copyright © ebXML 2000. All Rights Reserved.

1807

1808 ~~-This document and translations of it may be copied and furnished to others, and~~
1809 ~~derivative works that comment on or otherwise explain it or assist in its~~
1810 ~~implementation may be prepared, copied, published and distributed, in whole or~~
1811 ~~in part, without restriction of any kind, provided that the above copyright notice~~
1812 ~~and this paragraph are included on all such copies and derivative works.~~

1813 ~~However, this document itself may not be modified in any way, such as by~~
1814 ~~removing the copyright notice or references to the Internet Society or other~~
1815 ~~Internet organizations, except as needed for the purpose of developing Internet~~
1816 ~~standards in which case the procedures for copyrights defined in the Internet~~
1817 ~~Standards process must be followed, or as required to translate it into languages~~
1818 ~~other than English.~~

1819

1820 ~~-The limited permissions granted above are perpetual and will not be revoked by~~
1821 ~~ebXML or its successors or assigns.~~

1822

1823 ~~-This document and the information contained herein is provided on an~~
1824 ~~"AS IS" basis and ebXML DISCLAIMS ALL WARRANTIES, EXPRESS OR~~
1825 ~~IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE~~
1826 ~~USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR~~

OASIS/ebXML Registry Information Model

1827 ~~ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A~~
1828 ~~PARTICULAR PURPOSE.~~ |