



Creating A Single Global Electronic Market

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

OASIS/ebXML Registry Information Model v2.010

Bug Fixes To Approved OASIS Standard

OASIS/ebXML Registry Technical Committee

MayApril 2002

1 Status of this Document

Distribution of this document is unlimited.

This version:

<http://www.oasis-open.org/committees/regrep/documents/2.01/specs/ebRIM.pdf>

Latest version:

<http://www.oasis-open.org/committees/regrep/documents/2.01/specs/ebRIM.pdf>

20 **2 OASIS/ebXML Registry Technical Committee**

21 ~~Prior to being approved as an OASIS Standard, this document, in its current~~
22 ~~form, was an approved Committee Specification of the OASIS ebXML Registry~~
23 ~~Technical Committee. It builds upon version 1.0 which was approved by the~~
24 ~~OASIS/ebXML Registry Technical Committee as a DRAFT Specification of the~~
25 ~~TC~~This document has no standing and currently represents works-in-progress of
26 the OASIS ebXML Registry TC. A future version of this document will be finalized
27 and approved by the Registry TC as version 2.1.

28
29 At the time of v2.0 committee approval, the following were members of the
30 OASIS/ebXML Registry Technical Committee:

31
32 Kathryn Breining, Boeing
33 Lisa Carnahan, US NIST (TC Chair)
34 Joseph M. Chiusano, LMI
35 Suresh Damodaran, Sterling Commerce
36 Mike DeNicola Fujitsu
37 Anne Fischer, Drummond Group
38 Sally Fuger, AIAG
39 Jong Kim InnoDigital
40 Kyu-Chul Lee, Chungnam National University
41 Joel Munter, Intel
42 Farrukh Najmi, Sun Microsystems
43 Joel Neu, Vitria Technologies
44 Sanjay Patil, IONA
45 Neal Smith, ChevronTexaco
46 Nikola Stojanovic, Encoda Systems, Inc.
47 Prasad Yendluri, webMethods
48 Yutaka Yoshida, Sun Microsystems
49

50 **2.1 Contributors**

51 The following persons contributed to the content of this document, but are not
52 voting members of the OASIS/ebXML Registry Technical Committee.

53
54 Len Gallagher, NIST
55 Sekhar Vajjhala, Sun Microsystems

56

57

57 **Table of Contents**

58

59	<u>1 STATUS OF THIS DOCUMENT.....1</u>
60	<u>2 OASIS/EBXML REGISTRY TECHNICAL COMMITTEE.....2</u>
61	2.1 CONTRIBUTORS.....2
62	<u>3 INTRODUCTION13</u>
63	3.1 SUMMARY OF CONTENTS OF DOCUMENT.....13
64	3.2 GENERAL CONVENTIONS13
65	3.2.1 Naming Conventions.....13
66	3.3 AUDIENCE.....14
67	3.4 RELATED DOCUMENTS14
68	<u>4 DESIGN OBJECTIVES14</u>
69	4.1 GOALS14
70	<u>5 SYSTEM OVERVIEW15</u>
71	5.1 ROLE OF EBXML <i>REGISTRY</i>15
72	5.2 REGISTRY SERVICES15
73	5.3 WHAT THE REGISTRY INFORMATION MODEL DOES15
74	5.4 HOW THE REGISTRY INFORMATION MODEL WORKS15
75	5.5 WHERE THE REGISTRY INFORMATION MODEL MAY BE IMPLEMENTED15
76	5.6 CONFORMANCE TO AN EBXML <i>REGISTRY</i>16
77	<u>6 REGISTRY INFORMATION MODEL: HIGH LEVEL PUBLIC VIEW.....16</u>
78	6.1 REGISTRYOBJECT18
79	6.2 SLOT18
80	6.3 ASSOCIATION.....18
81	6.4 EXTERNALIDENTIFIER18
82	6.5 EXTERNALLINK18
83	6.6 CLASSIFICATIONSCHEME.....18
84	6.7 CLASSIFICATIONNODE.....19
85	6.8 CLASSIFICATION19
86	6.9 REGISTRYPACKAGE.....19
87	6.10 AUDITABLEEVENT.....19
88	6.11 USER.....19
89	6.12 POSTALADDRESS19
90	6.13 EMAILADDRESS19
91	6.14 ORGANIZATION.....20
92	6.15 SERVICE.....20
93	6.16 SERVICEBINDING.....20
94	6.17 SPECIFICATIONLINK20

95	7	REGISTRY INFORMATION MODEL: DETAIL VIEW	20
96	7.1	ATTRIBUTE AND METHODS OF INFORMATION MODEL CLASSES	21
97	7.2	DATA TYPES	22
98	7.3	INTERNATIONALIZATION (I18N) SUPPORT	22
99	7.3.1	Class <i>InternationalString</i>	22
100	7.3.2	Class <i>LocalizedString</i>	23
101	7.4	CLASS REGISTRYOBJECT	23
102	7.4.1	Attribute <i>Summary</i>	24
103	7.4.2	Attribute <i>accessControlPolicy</i>	24
104	7.4.3	Attribute <i>description</i>	24
105	7.4.4	Attribute <i>id</i>	25
106	7.4.5	Attribute <i>name</i>	25
107	7.4.6	Attribute <i>objectType</i>	25
108	7.4.7	Method <i>Summary</i>	26
109	7.5	CLASS REGISTRYENTRY	27
110	7.5.1	Attribute <i>Summary</i>	27
111	7.5.2	Attribute <i>expiration</i>	28
112	7.5.3	Attribute <i>majorVersion</i>	28
113	7.5.4	Attribute <i>minorVersion</i>	28
114	7.5.5	Attribute <i>stability</i>	28
115	7.5.6	Attribute <i>status</i>	29
116	7.5.7	Attribute <i>userVersion</i>	29
117	7.5.8	Method <i>Summary</i>	29
118	7.6	CLASS SLOT	30
119	7.6.1	Attribute <i>Summary</i>	30
120	7.6.2	Attribute <i>name</i>	30
121	7.6.3	Attribute <i>slotType</i>	30
122	7.6.4	Attribute <i>values</i>	30
123	7.7	CLASS EXTRINSICOBJECT	31
124	7.7.1	Attribute <i>Summary</i>	31
125	7.7.2	Attribute <i>isOpaque</i>	31
126	7.7.3	Attribute <i>mimeType</i>	31
127	7.8	CLASS REGISTRYPACKAGE	31
128	7.8.1	Attribute <i>Summary</i>	32
129	7.8.2	Method <i>Summary</i>	32
130	7.9	CLASS EXTERNALIDENTIFIER	32
131	7.9.1	Attribute <i>Summary</i>	32
132	7.9.2	Attribute <i>identificationScheme</i>	33
133	7.9.3	Attribute <i>registryObject</i>	33
134	7.9.4	Attribute <i>value</i>	33
135	7.10	CLASS EXTERNALLINK	33
136	7.10.1	Attribute <i>Summary</i>	33
137	7.10.2	Attribute <i>externalURI</i>	33
138	7.10.3	Method <i>Summary</i>	33
139	8	REGISTRY AUDIT TRAIL	34

140	<u>8.1 CLASS AUDITABLEEVENT.....</u>	<u>34</u>
141	<u>8.1.1 Attribute Summary.....</u>	<u>34</u>
142	<u>8.1.2 Attribute eventType.....</u>	<u>35</u>
143	<u>8.1.3 Attribute registryObject.....</u>	<u>35</u>
144	<u>8.1.4 Attribute timestamp.....</u>	<u>35</u>
145	<u>8.1.5 Attribute user.....</u>	<u>35</u>
146	<u>8.2 CLASS USER.....</u>	<u>35</u>
147	<u>8.2.1 Attribute Summary.....</u>	<u>35</u>
148	<u>8.2.2 Attribute address.....</u>	<u>36</u>
149	<u>8.2.3 Attribute emailAddresses.....</u>	<u>36</u>
150	<u>8.2.4 Attribute organization.....</u>	<u>36</u>
151	<u>8.2.5 Attribute personName.....</u>	<u>36</u>
152	<u>8.2.6 Attribute telephoneNumbers.....</u>	<u>36</u>
153	<u>8.2.7 Attribute url.....</u>	<u>36</u>
154	<u>8.3 CLASS ORGANIZATION.....</u>	<u>36</u>
155	<u>8.3.1 Attribute Summary.....</u>	<u>37</u>
156	<u>8.3.2 Attribute address.....</u>	<u>37</u>
157	<u>8.3.3 Attribute parent.....</u>	<u>37</u>
158	<u>8.3.4 Attribute primaryContact.....</u>	<u>37</u>
159	<u>8.3.5 Attribute telephoneNumbers.....</u>	<u>37</u>
160	<u>8.4 CLASS POSTALADDRESS.....</u>	<u>37</u>
161	<u>8.4.1 Attribute Summary.....</u>	<u>37</u>
162	<u>8.4.2 Attribute city.....</u>	<u>38</u>
163	<u>8.4.3 Attribute country.....</u>	<u>38</u>
164	<u>8.4.4 Attribute postalCode.....</u>	<u>38</u>
165	<u>8.4.5 Attribute state.....</u>	<u>38</u>
166	<u>8.4.6 Attribute street.....</u>	<u>38</u>
167	<u>8.4.7 Attribute streetNumber.....</u>	<u>38</u>
168	<u>8.4.8 Method Summary.....</u>	<u>38</u>
169	<u>8.5 CLASS TELEPHONENUMBER.....</u>	<u>38</u>
170	<u>8.5.1 Attribute Summary.....</u>	<u>39</u>
171	<u>8.5.2 Attribute areaCode.....</u>	<u>39</u>
172	<u>8.5.3 Attribute countryCode.....</u>	<u>39</u>
173	<u>8.5.4 Attribute extension.....</u>	<u>39</u>
174	<u>8.5.5 Attribute number.....</u>	<u>39</u>
175	<u>8.5.6 Attribute phoneType.....</u>	<u>39</u>
176	<u>8.6 CLASS EMAILADDRESS.....</u>	<u>39</u>
177	<u>8.6.1 Attribute Summary.....</u>	<u>39</u>
178	<u>8.6.2 Attribute address.....</u>	<u>40</u>
179	<u>8.6.3 Attribute type.....</u>	<u>40</u>
180	<u>8.7 CLASS PERSONNAME.....</u>	<u>40</u>
181	<u>8.7.1 Attribute Summary.....</u>	<u>40</u>
182	<u>8.7.2 Attribute firstName.....</u>	<u>40</u>
183	<u>8.7.3 Attribute lastName.....</u>	<u>40</u>
184	<u>8.7.4 Attribute middleName.....</u>	<u>40</u>
185	<u>8.8 CLASS SERVICE.....</u>	<u>40</u>

186	8.8.1 Attribute Summary.....	40
187	8.8.2 Method Summary.....	41
188	8.9 CLASS SERVICEBINDING.....	41
189	8.9.1 Attribute Summary.....	41
190	8.9.2 Attribute accessURI.....	41
191	8.9.3 Attribute targetBinding.....	41
192	8.9.4 Method Summary.....	42
193	8.10 CLASS SPECIFICATIONLINK.....	42
194	8.10.1 Attribute Summary.....	42
195	8.10.2 Attribute specificationObject.....	42
196	8.10.3 Attribute usageDescription.....	42
197	8.10.4 Attribute usageParameters.....	43
198	9 ASSOCIATION OF REGISTRY OBJECTS.....	44
199	9.1 EXAMPLE OF AN ASSOCIATION.....	44
200	9.2 SOURCE AND TARGET OBJECTS.....	44
201	9.3 ASSOCIATION TYPES.....	44
202	9.4 INTRAMURAL ASSOCIATION.....	45
203	9.5 EXTRAMURAL ASSOCIATION.....	45
204	9.6 CONFIRMATION OF AN ASSOCIATION.....	46
205	9.6.1 Confirmation of Intramural Associations.....	46
206	9.6.2 Confirmation of Extramural Associations.....	47
207	9.6.3 Deleting an Extramural Associations.....	47
208	9.7 VISIBILITY OF UNCONFIRMED ASSOCIATIONS.....	47
209	9.8 POSSIBLE CONFIRMATION STATES.....	47
210	9.9 CLASS ASSOCIATION.....	48
211	9.9.1 Attribute Summary.....	48
212	9.9.2 Attribute associationType.....	48
213	9.9.3 Attribute sourceObject.....	49
214	9.9.4 Attribute targetObject.....	49
215	9.9.5 Attribute isConfirmedBySourceOwner.....	49
216	9.9.6 Attribute isConfirmedByTargetOwner.....	50
217	10 CLASSIFICATION OF REGISTRYOBJECT.....	50
218	10.1 CLASS CLASSIFICATIONSCHEME.....	53
219	10.1.1 Attribute Summary.....	53
220	10.1.2 Attribute isInternal.....	53
221	10.1.3 Attribute nodeType.....	53
222	10.2 CLASS CLASSIFICATIONNODE.....	54
223	10.2.1 Attribute Summary.....	54
224	10.2.2 Attribute parent.....	54
225	10.2.3 Attribute code.....	54
226	10.2.4 Attribute code.....	54
227	10.2.5 Method Summary.....	55
228	10.2.6 Canonical Path Syntax.....	55
229	10.3 CLASS CLASSIFICATION.....	56

230	<i>10.3.1 Attribute Summary</i>	56
231	<i>10.3.2 Attribute classificationScheme</i>	57
232	<i>10.3.3 Attribute classificationNode</i>	57
233	<i>10.3.4 Attribute classifiedObject</i>	57
234	<i>10.3.5 Attribute nodeRepresentation</i>	57
235	<i>10.3.6 Context Sensitive Classification</i>	57
236	<i>10.3.7 Method Summary</i>	59
237	10.4 EXAMPLE OF CLASSIFICATION SCHEMES.....	60
238	<u>11 INFORMATION MODEL: SECURITY VIEW</u>	60
239	11.1 CLASS ACCESSCONTROLPOLICY.....	61
240	11.2 CLASS PERMISSION.....	62
241	11.3 CLASS PRIVILEGE.....	62
242	11.4 CLASS PRIVILEGEATTRIBUTE.....	63
243	11.5 CLASS ROLE.....	63
244	<i>11.5.1 A security Role PrivilegeAttribute</i>	63
245	11.6 CLASS GROUP.....	63
246	<i>11.6.1 A security Group PrivilegeAttribute</i>	63
247	11.7 CLASS IDENTITY.....	64
248	<i>11.7.1 A security Identity PrivilegeAttribute</i>	64
249	11.8 CLASS PRINCIPAL.....	64
250	<u>12 REFERENCES</u>	65
251	<u>13 DISCLAIMER</u>	65
252	<u>14 CONTACT INFORMATION</u>	66
253	<u>COPYRIGHT STATEMENT</u>	67
254	<u>1 STATUS OF THIS DOCUMENT</u>	1
255	<u>2 OASIS/EBXML REGISTRY TECHNICAL COMMITTEE</u>	2
256	<u>2.1 CONTRIBUTORS</u>	2
257	<u>3 INTRODUCTION</u>	8
258	<u>3.1 SUMMARY OF CONTENTS OF DOCUMENT</u>	8
259	<u>3.2 GENERAL CONVENTIONS</u>	8
260	<u>3.2.1 Naming Conventions</u>	8
261	<u>3.3 AUDIENCE</u>	9
262	<u>3.4 RELATED DOCUMENTS</u>	9
263	<u>4 DESIGN OBJECTIVES</u>	9
264	<u>4.1 GOALS</u>	9
265	<u>5 SYSTEM OVERVIEW</u>	10
266	<u>5.1 ROLE OF EBXML REGISTRY</u>	10

267	<u>5.2</u>	<u>REGISTRY SERVICES</u>	10
268	<u>5.3</u>	<u>WHAT THE REGISTRY INFORMATION MODEL DOES</u>	10
269	<u>5.4</u>	<u>HOW THE REGISTRY INFORMATION MODEL WORKS</u>	10
270	<u>5.5</u>	<u>WHERE THE REGISTRY INFORMATION MODEL MAY BE IMPLEMENTED</u>	10
271	<u>5.6</u>	<u>CONFORMANCE TO AN EBXML REGISTRY</u>	11
272	<u>6</u>	<u>REGISTRY INFORMATION MODEL: HIGH LEVEL PUBLIC VIEW</u>	11
273	<u>6.1</u>	<u>REGISTRYOBJECT</u>	12
274	<u>6.2</u>	<u>SLOT</u>	12
275	<u>6.3</u>	<u>ASSOCIATION</u>	12
276	<u>6.4</u>	<u>EXTERNALIDENTIFIER</u>	12
277	<u>6.5</u>	<u>EXTERNALLINK</u>	12
278	<u>6.6</u>	<u>CLASSIFICATIONSCHEME</u>	12
279	<u>6.7</u>	<u>CLASSIFICATIONNODE</u>	13
280	<u>6.8</u>	<u>CLASSIFICATION</u>	13
281	<u>6.9</u>	<u>REGISTRYPACKAGE</u>	13
282	<u>6.10</u>	<u>AUDITABLEEVENT</u>	13
283	<u>6.11</u>	<u>USER</u>	13
284	<u>6.12</u>	<u>POSTALADDRESS</u>	13
285	<u>6.13</u>	<u>EMAILADDRESS</u>	13
286	<u>6.14</u>	<u>ORGANIZATION</u>	14
287	<u>6.15</u>	<u>SERVICE</u>	14
288	<u>6.16</u>	<u>SERVICEBINDING</u>	14
289	<u>6.17</u>	<u>SPECIFICATIONLINK</u>	14
290	<u>7</u>	<u>REGISTRY INFORMATION MODEL: DETAIL VIEW</u>	14
291	<u>7.1</u>	<u>ATTRIBUTE AND METHODS OF INFORMATION MODEL CLASSES</u>	15
292	<u>7.2</u>	<u>DATA TYPES</u>	16
293	<u>7.3</u>	<u>INTERNATIONALIZATION (I18N) SUPPORT</u>	16
294	<u>7.3.1</u>	<u>Class InternationalString</u>	16
295	<u>7.3.2</u>	<u>Class LocalizedString</u>	17
296	<u>7.4</u>	<u>CLASS REGISTRYOBJECT</u>	17
297	<u>7.4.1</u>	<u>Attribute Summary</u>	17
298	<u>7.4.2</u>	<u>Attribute accessControlPolicy</u>	18
299	<u>7.4.3</u>	<u>Attribute description</u>	18
300	<u>7.4.4</u>	<u>Attribute id</u>	18
301	<u>7.4.5</u>	<u>Attribute name</u>	18
302	<u>7.4.6</u>	<u>Attribute objectType</u>	18
303	<u>7.4.7</u>	<u>Method Summary</u>	20
304	<u>7.5</u>	<u>CLASS REGISTRYENTRY</u>	20
305	<u>7.5.1</u>	<u>Attribute Summary</u>	21
306	<u>7.5.2</u>	<u>Attribute expiration</u>	21
307	<u>7.5.3</u>	<u>Attribute majorVersion</u>	21
308	<u>7.5.4</u>	<u>Attribute minorVersion</u>	21
309	<u>7.5.5</u>	<u>Attribute stability</u>	22
310	<u>7.5.6</u>	<u>Attribute status</u>	22

311	<u>7.5.7</u>	<u>Attribute userVersion</u>	23
312	<u>7.5.8</u>	<u>Method Summary</u>	23
313	<u>7.6</u>	<u>CLASS SLOT</u>	23
314	<u>7.6.1</u>	<u>Attribute Summary</u>	23
315	<u>7.6.2</u>	<u>Attribute name</u>	24
316	<u>7.6.3</u>	<u>Attribute slotType</u>	24
317	<u>7.6.4</u>	<u>Attribute values</u>	24
318	<u>7.7</u>	<u>CLASS EXTRINSICOBJECT</u>	24
319	<u>7.7.1</u>	<u>Attribute Summary</u>	24
320	<u>7.7.2</u>	<u>Attribute isOpaque</u>	25
321	<u>7.7.3</u>	<u>Attribute mimeType</u>	25
322	<u>7.8</u>	<u>CLASS REGISTRYPACKAGE</u>	25
323	<u>7.8.1</u>	<u>Attribute Summary</u>	25
324	<u>7.8.2</u>	<u>Method Summary</u>	25
325	<u>7.9</u>	<u>CLASS EXTERNALIDENTIFIER</u>	25
326	<u>7.9.1</u>	<u>Attribute Summary</u>	26
327	<u>7.9.2</u>	<u>Attribute identificationScheme</u>	26
328	<u>7.9.3</u>	<u>Attribute registryObject</u>	26
329	<u>7.9.4</u>	<u>Attribute value</u>	26
330	<u>7.10</u>	<u>CLASS EXTERNALLINK</u>	26
331	<u>7.10.1</u>	<u>Attribute Summary</u>	26
332	<u>7.10.2</u>	<u>Attribute externalURI</u>	27
333	<u>7.10.3</u>	<u>Method Summary</u>	27
334	8	<u>REGISTRY AUDIT TRAIL</u>	27
335	<u>8.1</u>	<u>CLASS AUDITABLEEVENT</u>	27
336	<u>8.1.1</u>	<u>Attribute Summary</u>	28
337	<u>8.1.2</u>	<u>Attribute eventType</u>	28
338	<u>8.1.3</u>	<u>Attribute registryObject</u>	28
339	<u>8.1.4</u>	<u>Attribute timestamp</u>	28
340	<u>8.1.5</u>	<u>Attribute user</u>	28
341	<u>8.2</u>	<u>CLASS USER</u>	29
342	<u>8.2.1</u>	<u>Attribute Summary</u>	29
343	<u>8.2.2</u>	<u>Attribute address</u>	29
344	<u>8.2.3</u>	<u>Attribute emailAddresses</u>	29
345	<u>8.2.4</u>	<u>Attribute organization</u>	29
346	<u>8.2.5</u>	<u>Attribute personName</u>	29
347	<u>8.2.6</u>	<u>Attribute telephoneNumbers</u>	30
348	<u>8.2.7</u>	<u>Attribute url</u>	30
349	<u>8.3</u>	<u>CLASS ORGANIZATION</u>	30
350	<u>8.3.1</u>	<u>Attribute Summary</u>	30
351	<u>8.3.2</u>	<u>Attribute address</u>	30
352	<u>8.3.3</u>	<u>Attribute parent</u>	30
353	<u>8.3.4</u>	<u>Attribute primaryContact</u>	30
354	<u>8.3.5</u>	<u>Attribute telephoneNumbers</u>	30
355	<u>8.4</u>	<u>CLASS POSTALADDRESS</u>	31

356	8.4.1	Attribute Summary	31
357	8.4.2	Attribute city	31
358	8.4.3	Attribute country	31
359	8.4.4	Attribute postalCode	31
360	8.4.5	Attribute state	31
361	8.4.6	Attribute street	31
362	8.4.7	Attribute streetNumber	31
363	8.4.8	Method Summary	32
364	8.5	CLASS TELEPHONENUMBER	32
365	8.5.1	Attribute Summary	32
366	8.5.2	Attribute areaCode	32
367	8.5.3	Attribute countryCode	32
368	8.5.4	Attribute extension	32
369	8.5.5	Attribute number	33
370	8.5.6	Attribute phoneType	33
371	8.6	CLASS EMAILADDRESS	33
372	8.6.1	Attribute Summary	33
373	8.6.2	Attribute address	33
374	8.6.3	Attribute type	33
375	8.7	CLASS PERSONNAME	33
376	8.7.1	Attribute Summary	33
377	8.7.2	Attribute firstName	33
378	8.7.3	Attribute lastName	34
379	8.7.4	Attribute middleName	34
380	8.8	CLASS SERVICE	34
381	8.8.1	Attribute Summary	34
382	8.8.2	Method Summary	34
383	8.9	CLASS SERVICEBINDING	34
384	8.9.1	Attribute Summary	35
385	8.9.2	Attribute accessURI	35
386	8.9.3	Attribute targetBinding	35
387	8.9.4	Method Summary	35
388	8.10	CLASS SPECIFICATIONLINK	35
389	8.10.1	Attribute Summary	36
390	8.10.2	Attribute specificationObject	36
391	8.10.3	Attribute usageDescription	36
392	8.10.4	Attribute usageParameters	36
393	9	ASSOCIATION OF REGISTRY OBJECTS	37
394	9.1	EXAMPLE OF AN ASSOCIATION	37
395	9.2	SOURCE AND TARGET OBJECTS	37
396	9.3	ASSOCIATION TYPES	37
397	9.4	INTRAMURAL ASSOCIATION	38
398	9.5	EXTRAMURAL ASSOCIATION	38
399	9.6	CONFIRMATION OF AN ASSOCIATION	39
400	9.6.1	Confirmation of Intramural Associations	39

401	<u>9.6.2</u> — <u>Confirmation of Extramural Associations</u>	40
402	<u>9.7</u> — <u>VISIBILITY OF UNCONFIRMED ASSOCIATIONS</u>	40
403	<u>9.8</u> — <u>POSSIBLE CONFIRMATION STATES</u>	40
404	<u>9.9</u> — <u>CLASS ASSOCIATION</u>	40
405	<u>9.9.1</u> — <u>Attribute Summary</u>	41
406	<u>9.9.2</u> — <u>Attribute associationType</u>	41
407	<u>9.9.3</u> — <u>Attribute sourceObject</u>	42
408	<u>9.9.4</u> — <u>Attribute targetObject</u>	42
409	<u>10</u> — <u>CLASSIFICATION OF REGISTRYOBJECT</u>	43
410	<u>10.1</u> — <u>CLASS CLASSIFICATIONSCHEME</u>	46
411	<u>10.1.1</u> — <u>Attribute Summary</u>	46
412	<u>10.1.2</u> — <u>Attribute isInternal</u>	46
413	<u>10.1.3</u> — <u>Attribute nodeType</u>	46
414	<u>10.2</u> — <u>CLASS CLASSIFICATIONNODE</u>	47
415	<u>10.2.1</u> — <u>Attribute Summary</u>	47
416	<u>10.2.2</u> — <u>Attribute parent</u>	47
417	<u>10.2.3</u> — <u>Attribute code</u>	47
418	<u>10.2.4</u> — <u>Method Summary</u>	47
419	<u>10.2.5</u> — <u>Canonical Path Syntax</u>	48
420	<u>10.3</u> — <u>CLASS CLASSIFICATION</u>	49
421	<u>10.3.1</u> — <u>Attribute Summary</u>	49
422	<u>10.3.2</u> — <u>Attribute classificationScheme</u>	50
423	<u>10.3.3</u> — <u>Attribute classificationNode</u>	50
424	<u>10.3.4</u> — <u>Attribute classifiedObject</u>	50
425	<u>10.3.5</u> — <u>Attribute nodeRepresentation</u>	50
426	<u>10.3.6</u> — <u>Context Sensitive Classification</u>	50
427	<u>10.3.7</u> — <u>Method Summary</u>	52
428	<u>10.4</u> — <u>EXAMPLE OF CLASSIFICATION SCHEMES</u>	53
429	<u>11</u> — <u>INFORMATION MODEL: SECURITY VIEW</u>	53
430	<u>11.1</u> — <u>CLASS ACCESSCONTROLPOLICY</u>	54
431	<u>11.2</u> — <u>CLASS PERMISSION</u>	55
432	<u>11.3</u> — <u>CLASS PRIVILEGE</u>	55
433	<u>11.4</u> — <u>CLASS PRIVILEGEATTRIBUTE</u>	56
434	<u>11.5</u> — <u>CLASS ROLE</u>	56
435	<u>11.5.1</u> — <u>A security Role PrivilegeAttribute</u>	56
436	<u>11.6</u> — <u>CLASS GROUP</u>	56
437	<u>11.6.1</u> — <u>A security Group PrivilegeAttribute</u>	56
438	<u>11.7</u> — <u>CLASS IDENTITY</u>	57
439	<u>11.7.1</u> — <u>A security Identity PrivilegeAttribute</u>	57
440	<u>11.8</u> — <u>CLASS PRINCIPAL</u>	57
441	<u>12</u> — <u>REFERENCES</u>	58
442	<u>13</u> — <u>DISCLAIMER</u>	58

443	<u>14</u> — <u>CONTACT INFORMATION</u>	59
444	<u>COPYRIGHT STATEMENT</u>	60

445 **Table of Figures**

446	<u>Figure 1: Information Model High Level Public View</u>	<u>17</u>
447	<u>Figure 2: Information Model <i>Inheritance</i> View</u>	<u>21</u>
448	<u>Figure 3: Example of RegistryObject Association</u>	<u>44</u>
449	<u>Figure 4: Example of Intramural Association</u>	<u>45</u>
450	<u>Figure 5: Example of Extramural Association</u>	<u>46</u>
451	<u>Figure 6: Example showing a <i>Classification Tree</i></u>	<u>51</u>
452	<u>Figure 7: Information Model <i>Classification</i> View</u>	<u>52</u>
453	<u>Figure 8: Classification <i>Instance</i> Diagram</u>	<u>52</u>
454	<u>Figure 9: Context Sensitive <i>Classification</i></u>	<u>58</u>
455	<u>Figure 10: Information Model: Security View</u>	<u>61</u>
456	<u>Figure 1: Information Model High Level Public View</u>	<u>11</u>
457	<u>Figure 2: Information Model <i>Inheritance</i> View</u>	<u>15</u>
458	<u>Figure 3: Example of RegistryObject Association</u>	<u>37</u>
459	<u>Figure 4: Example of Intramural Association</u>	<u>38</u>
460	<u>Figure 5: Example of Extramural Association</u>	<u>39</u>
461	<u>Figure 6: Example showing a <i>Classification Tree</i></u>	<u>44</u>
462	<u>Figure 7: Information Model <i>Classification</i> View</u>	<u>45</u>
463	<u>Figure 8: Classification <i>Instance</i> Diagram</u>	<u>45</u>
464	<u>Figure 9: Context Sensitive <i>Classification</i></u>	<u>51</u>
465	<u>Figure 10: Information Model: Security View</u>	<u>54</u>

466 **Table of Tables**

467	<u>Table 1: Sample <i>Classification</i> Schemes</u>	<u>60</u>
468	<u>Table 1: Sample <i>Classification</i> Schemes</u>	<u>53</u>
469		
470		

470 **3 Introduction**

471 **3.1 Summary of Contents of Document**

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

473

474 A separate document, ebXML Registry Services Specification [ebRS], describes
475 how to build *Registry Services* that provide access to the information content in
476 the ebXML *Registry*.

477 **3.2 General Conventions**

478 The following conventions are used throughout this document:

479

480 UML diagrams are used as a way to concisely describe concepts. They are not
481 intended to convey any specific *Implementation* or methodology requirements.

482

483 The term "*repository item*" is used to refer to an object that has resides in a
484 repository for storage and safekeeping (e.g., an XML document or a DTD). Every
485 repository item is described in the Registry by a RegistryObject instance.

486

487 The term "*RegistryEntry*" is used to refer to an object that provides metadata
488 about a *repository item*.

489

490 The information model does not deal with the actual content of the repository. All
491 *Elements* of the information model represent metadata about the content and not
492 the content itself.

493

494 *Capitalized Italic* words are defined in the ebXML Glossary.

495

496 The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD,
497 SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in
498 this document, are to be interpreted as described in RFC 2119 [Bra97].

499

500 Software practitioners MAY use this document in combination with other ebXML
501 specification documents when creating ebXML compliant software.

502 **3.2.1 Naming Conventions**

503

504 In order to enforce a consistent capitalization and naming convention in this
505 document, "Upper Camel Case" (*UCC*) and "Lower Camel Case" (*LCC*)

506 Capitalization styles are used in the following conventions:

- 507 ○ Element name is in *UCC* convention
508 (example: <UpperCamelCaseElement/>)
- 509 ○ Attribute name is in *LCC* convention

- 510 (example: <UpperCamelCaseElement
511 lowerCamelCaseAttribute="whatEver"/>)
512 ○ Class, Interface names use UCC convention
513 (examples: ClassificationNode, Versionable)
514 ○ Method name uses LCC convention
515 (example: getName(), setName()).
516

517 Also, *Capitalized Italics* words are defined in the ebXML Glossary [ebGLOSS].

518 **3.3 Audience**

519 The target audience for this specification is the community of software
520 developers who are:

- 521 ○ Implementers of ebXML *Registry Services*
522 ○ Implementers of ebXML *Registry Clients*

523 **3.4 Related Documents**

524 The following specifications provide some background and related information to
525 the reader:

- 526
527 a) ebXML Registry Services Specification [ebRS] - defines the actual
528 *Registry Services* based on this information model
529 b) ebXML Collaboration-Protocol Profile and Agreement Specification
530 [ebCPP] - defines how profiles can be defined for a *Party* and how two
531 *Parties'* profiles may be used to define a *Party* agreement
532

533 **4 Design Objectives**

534 **4.1 Goals**

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

- 536 ○ Communicate what information is in the *Registry* and how that information
537 is organized
538 ○ Leverage as much as possible the work done in the OASIS [OAS] and the
539 ISO 11179 [ISO] Registry models
540 ○ Align with relevant works within other ebXML working groups
541 ○ Be able to evolve to support future ebXML *Registry* requirements
542 ○ Be compatible with other ebXML specifications
543

544 **5 System Overview**

545 **5.1 Role of ebXML Registry**

546

547 The *Registry* provides a stable store where information submitted by a
548 *Submitting Organization* is made persistent. Such information is used to facilitate
549 ebXML-based *Business to Business* (B2B) partnerships and transactions.

550 Submitted content may be *XML* schema and documents, process descriptions,
551 ebXML *Core Components*, context descriptions, *UML* models, information about
552 parties and even software components.

553 **5.2 Registry Services**

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

558 **5.3 What the Registry Information Model Does**

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

563 The Registry information model:

- 564 ○ Defines what types of objects are stored in the *Registry*
- 565 ○ Defines how stored objects are organized in the *Registry*

566

567 **5.4 How the Registry Information Model Works**

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

573 [Note]The information model is meant to be
574 illustrative and does not prescribe any
575 specific *Implementation* choices.

576

577 **5.5 Where the Registry Information Model May Be Implemented**

578 The Registry Information Model MAY be implemented within an ebXML *Registry*
579 in the form of a relational database schema, object database schema or some

580 other physical schema. It MAY also be implemented as interfaces and *Classes*
581 within a *Registry Implementation*.

582 **5.6 Conformance to an ebXML Registry**

583 If an *Implementation* claims *Conformance* to this specification then it supports all
584 required information model *Classes* and interfaces, their attributes and their
585 semantic definitions that are visible through the ebXML *Registry Services*.

586 **6 Registry Information Model: High Level Public View**

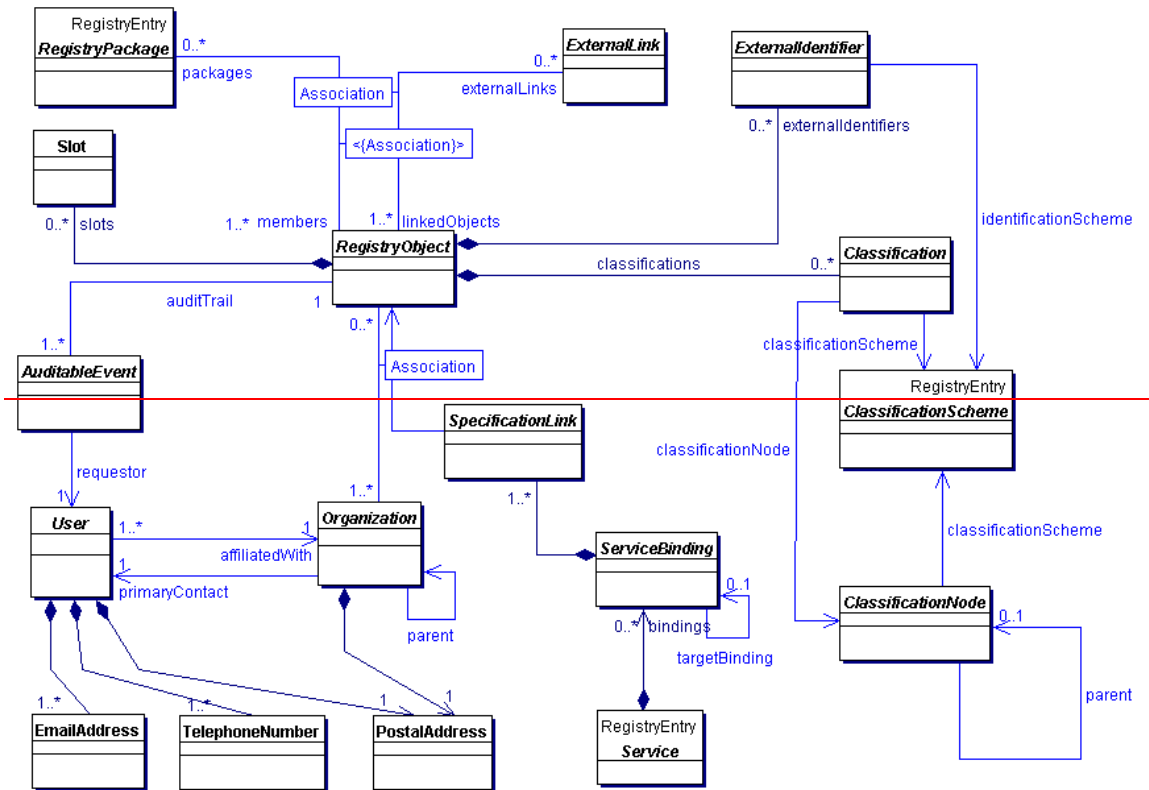
587 This section provides a high level public view of the most visible objects in the
588 *Registry*.

589

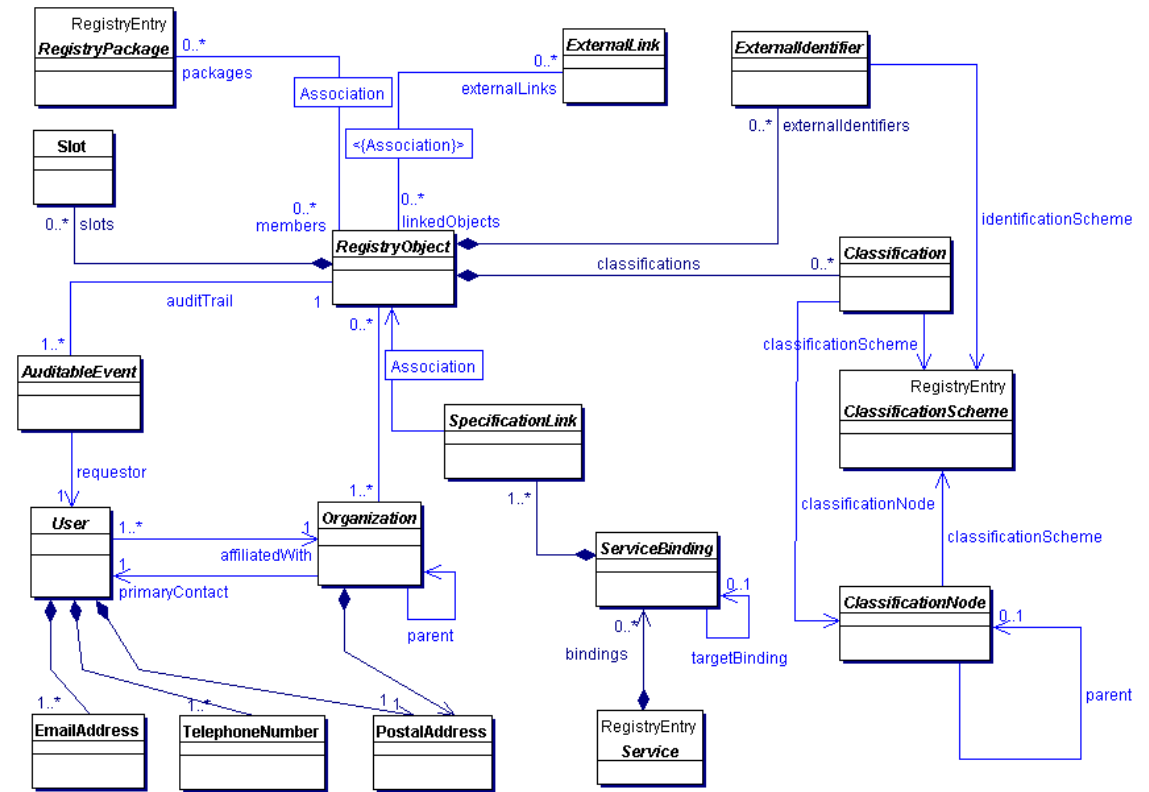
590 Figure 1~~Figure 4~~ shows the high level public view of the objects in the *Registry* |
591 and their relationships as a *UML Class Diagram*. It does not show *Inheritance*,
592 *Class* attributes or *Class* methods.

593 The reader is again reminded that the information model is not modeling actual
594 repository items.

595



596



597

598

Figure 1: Information Model High Level Public View

599 **6.1 RegistryObject**

600 The RegistryObject class is an abstract base class used by most classes in the
601 model. It provides minimal metadata for registry objects. It also provides methods
602 for accessing related objects that provide additional dynamic metadata for the
603 registry object.

604 **6.2 Slot**

605 Slot instances provide a dynamic way to add arbitrary attributes to
606 RegistryObject instances. This ability to add attributes dynamically to
607 RegistryObject instances enables extensibility within the Registry Information
608 Model. For example, if a company wants to add a “copyright” attribute to each
609 RegistryObject instance that it submits, it can do so by adding a slot with name
610 “copyright” and value containing the copyrights statement.

611 **6.3 Association**

612 Association instances are RegistryObject instances that are used to define many-
613 to-many associations between objects in the information model. Associations are
614 described in detail in section 9.

615 **6.4 ExternalIdentifier**

616 ExternalIdentifier instances provide additional identifier information to a
617 RegistryObject instance, such as DUNS number, Social Security Number, or an
618 alias name of the organization.

619 **6.5 ExternalLink**

620 ExternalLink instances are RegistryObject instances that model a named URI to
621 content that is not managed by the *Registry*. Unlike managed content, such
622 external content may change or be deleted at any time without the knowledge of
623 the *Registry*. A RegistryObject instance may be associated with any number of
624 ExternalLinks.

625 Consider the case where a *Submitting Organization* submits a repository item
626 (e.g., a *DTD*) and wants to associate some external content to that object (e.g.,
627 the *Submitting Organization's* home page). The ExternalLink enables this
628 capability. A potential use of the ExternalLink capability may be in a GUI tool that
629 displays the ExternalLinks to a RegistryObject. The user may click on such links
630 and navigate to an external web page referenced by the link.

631 **6.6 ClassificationScheme**

632 ClassificationScheme instances are RegistryEntry instances that describe a
633 structured way to classify or categorize RegistryObject instances. The structure
634 of the classification scheme may be defined internal or external to the registry,
635 resulting in a distinction between internal and external classification schemes. A
636 very common example of a classification scheme in science is the *Classification*
637 *of living things* where living things are categorized in a tree like structure. Another

638 example is the Dewey Decimal system used in libraries to categorize books and
639 other publications. ClassificationScheme is described in detail in section 10.

640 **6.7 ClassificationNode**

641 ClassificationNode instances are RegistryObject instances that are used to
642 define tree structures under a ClassificationScheme, where each node in the tree
643 is a ClassificationNode and the root is the ClassificationScheme. *Classification*
644 trees constructed with ClassificationNodes are used to define the structure of
645 *Classification* schemes or ontologies. ClassificationNode is described in detail in
646 section 10.

647 **6.8 Classification**

648 Classification instances are RegistryObject instances that are used to classify
649 other RegistryObject instances. A Classification instance identifies a
650 ClassificationScheme instance and taxonomy value defined within the
651 classification scheme. Classifications can be internal or external depending on
652 whether the referenced classification scheme is internal or external.
653 Classification is described in detail in section 10.

654 **6.9 RegistryPackage**

655 RegistryPackage instances are RegistryEntry instances that group logically
656 related RegistryObject instances together.

657 **6.10 AuditableEvent**

658 AuditableEvent instances are RegistryObject instances that are used to provide
659 an audit trail for RegistryObject instances. AuditableEvent is described in detail in
660 section 8.

661 **6.11 User**

662 User instances are RegistryObject instances that are used to provide information
663 about registered users within the *Registry*. User objects are used in audit trail for
664 RegistryObject instances. User is described in detail in section 8.

665 **6.12 PostalAddress**

666 PostalAddress is a simple reusable *Entity Class* that defines attributes of a postal
667 address.

668 **6.13 EmailAddress**

669 EmailAddress is a simple reusable *Entity Class* that defines attributes of an email
670 address.

671 **6.14 Organization**

672 Organization instances are RegistryObject instances that provide information on
673 organizations such as a *Submitting Organization*. Each Organization instance
674 may have a reference to a parent Organization.

675 **6.15 Service**

676 Service instances are RegistryEntry instances that provide information on
677 services (e.g., web services).

678 **6.16 ServiceBinding**

679 ServiceBinding instances are RegistryObject instances that represent technical
680 information on a specific way to access a specific interface offered by a Service
681 instance. A Service has a collection of ServiceBindings.
682

683 **6.17 SpecificationLink**

684 A SpecificationLink provides the linkage between a ServiceBinding and one of its
685 technical specifications that describes how to use the service with that
686 ServiceBinding. For example, a ServiceBinding may have a SpecificationLink
687 instance that describes how to access the service using a technical specification
688 in the form of a WSDL document or a CORBA IDL document.
689

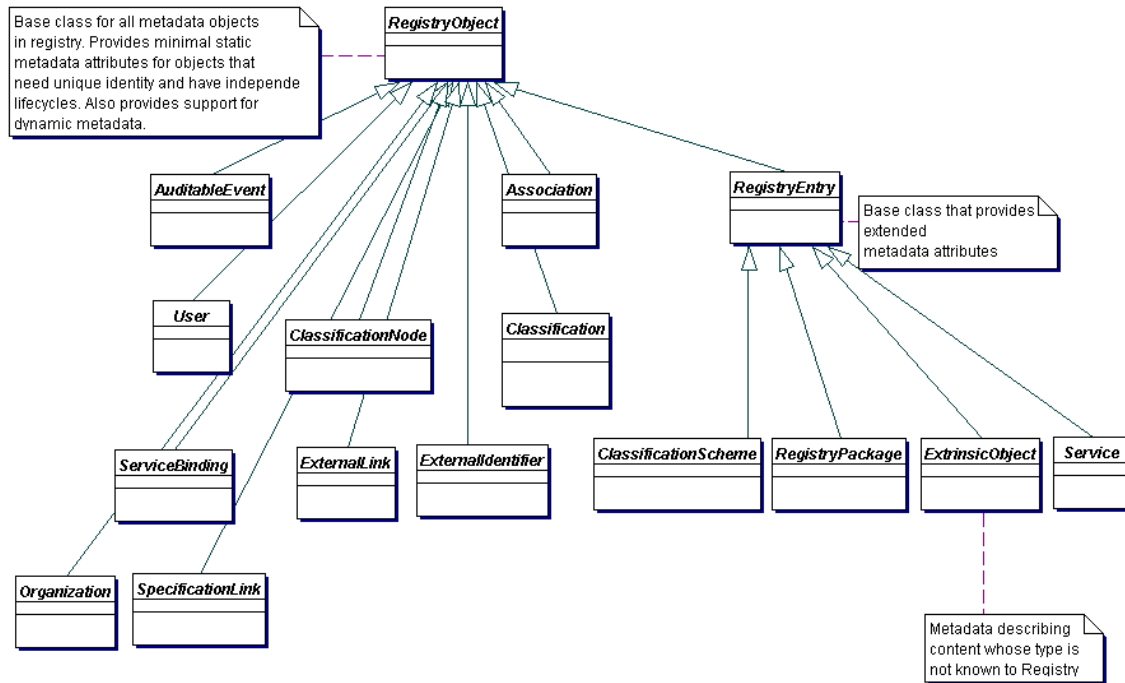
690 **7 Registry Information Model: Detail View**

691 This section covers the information model *Classes* in more detail than the Public
692 View. The detail view introduces some additional *Classes* within the model that
693 were not described in the public view of the information model.
694

695 [Figure 2](#) ~~Figure-2~~ shows the *Inheritance* or “is a” relationships between the
696 *Classes* in the information model. Note that it does not show the other types of
697 relationships, such as “has a” relationships, since they have already been shown
698 in a previous figure. *Class* attributes and *class* methods are also not shown.
699 Detailed description of methods and attributes of most interfaces and *Classes* will
700 be displayed in tabular form following the description of each *Class* in the model.
701

702 The class Association will be covered in detail separately in section 9. The
703 classes ClassificationScheme, Classification, and ClassificationNode will be
704 covered in detail separately in section 10.
705

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



708
709
710

Figure 2: Information Model *Inheritance View*

711 **7.1 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
717 Information model classes may also have methods defined for them. These
718 methods provide additional behavior for the class they are defined within.
719 Methods are currently used in mapping to filter query and the SQL query
720 capabilities defined in [ebRS].

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

7.2 Data Types

725
726 The following table lists the various data types used by the attributes within
727 information model classes:
728

Data Type	XML Schema Data Type	Description	Length
Boolean	boolean	Used for a true or false value	
String4	string	Used for 4 character long strings	4 characters
String8	string	Used for 8 character long strings	8 characters
String16	string	Used for 16 character long strings	16 characters
String32	string	Used for 32 character long strings	32 characters
String	string	Used for unbounded Strings	unbounded
ShortName	string	A short text string	64 characters
LongName	string	A long text string	128 characters
FreeFormText	string	A very long text string for free-form text	256 characters
UUID	string	DCE 128 Bit Universally unique Ids used for referencing another object	64 characters
URI	string	Used for URL and URN values	256 characters
Integer	integer	Used for integer values	4 bytes
DateTime	dateTime	Used for a timestamp value such as Date	

729

7.3 Internationalization (I18N) Support

731 Some information model classes have String attributes that are I18N capable and
732 may be localized into multiple native languages. Examples include the name and
733 description attributes of the RegistryObject class in 7.4.

734

735 The information model defines the InternationalString and the LocalizedString
736 interfaces to support I18N capable attributes within the information model
737 classes. These classes are defined below.

7.3.1 Class InternationalString

739 This class is used as a replacement for the String type whenever a String
740 attribute needs to be I18N capable. An instance of the InternationalString class
741 composes within it a Collection of LocalizedString instances, where each String
742 is specific to a particular locale. The InternationalString class provides set/get

743 methods for adding or getting locale specific String values for the
744 InternationalString instance.

745 [7.3.1.1 Attribute Summary](#)

746

<u>Attribute</u>	<u>Data Type</u>	<u>Required</u>	<u>Default Value</u>	<u>Specified By</u>	<u>Mutable</u>
localized-Strings	Collection of Localized-String	No		Client	Yes

747

748 [7.3.1.2 Attribute localizedStrings](#)

749 [Each InternationalString instance may have localizedString attribute that is a](#)
750 [Collection of zero or more LocalizedString instances.](#)

751 **7.3.2 Class LocalizedString**

752 This class is used as a simple wrapper class that associates a String with its
753 locale. The class is needed in the InternationalString class where a Collection of
754 LocalizedString instances are kept. Each LocalizedString instance has a charset
755 and lang attribute as well as a value attribute of type String.

756 [7.3.2.1 Attribute Summary](#)

757

<u>Attribute</u>	<u>Data Type</u>	<u>Required</u>	<u>Default Value</u>	<u>Specified By</u>	<u>Mutable</u>
lang	language	No	en-us	Client	Yes
charset	string	No	UTF-8	Client	Yes
value	string	Yes		Client	Yes

758

759 [7.3.2.2 Attribute lang](#)

760 [Each LocalizedString instance may have a lang attribute that specifies the](#)
761 [language used by that LocalizedString.](#)

762 [7.3.2.3 Attribute charset](#)

763 [Each LocalizedString instance may have a charset attribute that specifies the](#)
764 [name of the character set used by that LocalizedString.](#)

765 [7.3.2.4 Attribute value](#)

766 [Each LocalizedString instance must have a value attribute that specifies the](#)
767 [string value used by that LocalizedString.](#)

768 **7.4 Class RegistryObject**

769 **Direct Known Subclasses:**

770 [Association](#), [AuditableEvent](#), [Classification](#), [ClassificationNode](#),
771 [ExternalIdentifier](#), [ExternalLink](#), [Organization](#), [RegistryEntry](#), [User](#),
772 [Service](#), [ServiceBinding](#), [SpecificationLink](#)

773

774 RegistryObject provides a common base class for almost all objects in the
775 information model. Information model *Classes* whose instances have a unique
776 identity are descendants of the RegistryObject *Class*.

777

778 Note that Slot, PostalAddress, and a few other classes are not descendants of
779 the RegistryObject Class because their instances do not have an independent
780 existence and unique identity. They are always a part of some other Class's
781 Instance (e.g., Organization has a PostalAddress).

782 7.4.1 Attribute Summary

783 The following is the first of many tables that summarize the attributes of a class.

784 The columns in the table are described as follows:

785

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

786

Attribute	Data Type	Required	Default Value	Specified By	Mutable
accessControlPolicy	UUID	No		Registry	No
description	International-String	No		Client	Yes
id	UUID	Yes		Client or registry	No
name	International-String	No		Client	Yes
objectType	LongName	Yes		Registry	No

787 7.4.2 Attribute accessControlPolicy

788 Each RegistryObject instance may have an accessControlPolicy instance
789 associated with it. An accessControlPolicy instance defines the *Security Model*
790 associated with the RegistryObject in terms of “who is permitted to do what” with
791 that RegistryObject.

792 7.4.3 Attribute description

793 Each RegistryObject instance may have textual description in a human readable
794 and user-friendly manner. This attribute is I18N capable and therefore of type
795 InternationalString.

796 **7.4.4 Attribute id**

797 Each RegistryObject instance must have a universally unique ID. Registry
798 objects use the id of other RegistryObject instances for the purpose of
799 referencing those objects.

800

801 Note that some classes in the information model do not have a need for a unique
802 id. Such classes do not inherit from RegistryObject class. Examples include
803 Entity classes such as TelephoneNumber, PostalAddress, EmailAddress and
804 PersonName.

805

806 All classes derived from RegistryObject have an id that is a Universally Unique ID
807 as defined by [UUID]. Such UUID based id attributes may be specified by the
808 client. If the UUID based id is not specified, then it must be generated by the
809 registry when a new RegistryObject instance is first submitted to the registry.

810 **7.4.5 Attribute name**

811 Each RegistryObject instance may have human readable name. The name does
812 not need to be unique with respect to other RegistryObject instances. This
813 attribute is I18N capable and therefore of type InternationalString.

814 **7.4.6 Attribute objectType**

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

820 **7.4.6.1 Pre-defined Object Types**

821 The following table lists pre-defined object types. Note that for an ExtrinsicObject
822 there are many types defined based on the type of repository item the
823 ExtrinsicObject catalogues. In addition there are object types defined for all leaf
824 sub-classes of RegistryObject.

825

826

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

830

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 document called <i>Collaboration Protocol Profile (CPP)</i> that provides information about a <i>Party</i> participating in a <i>Business</i> transaction. See [ebCPP] for details.
Process	An ExtrinsicObject of this type catalogues a process description document.
SoftwareComponent	An ExtrinsicObject of this type catalogues a software component (e.g., an EJB or <i>Class</i> library).
UMLModel	An ExtrinsicObject of this type catalogues a <i>UML</i> model.
XMLSchema	An ExtrinsicObject of this type catalogues an <i>XML</i> schema (<i>DTD</i> , <i>XML</i> Schema, RELAX grammar, etc.).
RegistryPackage	A RegistryPackage object
ExternalLink	An ExternalLink object
ExternalIdentifier	An ExternalIdentifier object
Association	An Association object
ClassificationScheme	A ClassificationScheme object
Classification	A Classification object
ClassificationNode	A ClassificationNode object
AuditableEvent	An AuditableEvent object
User	A User object
Organization	An Organization object
Service	A Service object
ServiceBinding	A ServiceBinding object
SpecificationLink	A SpecificationLink object

831

832 **7.4.7 Method Summary**

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

836

Method Summary for RegistryObject	
Collection	getAssociations() Gets all Associations where this object is the source of the Association.
Collection	getAuditTrail() Gets the complete audit trail of all requests that effected a state change in this object as an ordered Collection of

	AuditableEvent objects.
Collection	getClassifications() Gets the Classification that classify this object.
Collection	getExternalIdentifiers() Gets the collection of ExternalIdentifiers associated with this object.
Collection	getExternalLinks() Gets the ExternalLinks associated with this object.
Collection	getOrganizations(<u>LongNameString</u> type) 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.
Collection	getRegistryPackages() Gets the RegistryPackages that this object is a member of.
Collection	getSlots() Gets the Slots associated with this object.

837

838

839 7.5 Class RegistryEntry

840 **Super Classes:**

841 [RegistryObject](#)

842

843 **Direct Known Subclasses:**

844 [ClassificationScheme](#), [ExtrinsicObject](#), [RegistryPackage](#)

845

846 RegistryEntry is a common base *Class* for classes in the information model that
847 require additional metadata beyond the minimal metadata provided by
848 RegistryObject class. RegistryEntry is used as a base class for high level coarse
849 grained objects in the registry. Their life cycle typically requires more
850 management (e.g. may require approval, deprecation). They typically have
851 relatively fewer instances but serve as a root of a composition hierarchy
852 consisting of numerous objects that are sub-classes of RegistryObject but not
853 RegistryEntry.

854

855 The additional metadata is described by the attributes of the RegistryEntry class
856 below.

857 7.5.1 Attribute Summary

858

Attribute	Data Type	Required	Default Value	Specified By	Mutable
expiration	DateTime	No		Client	Yes

majorVersion	Integer	Yes	1	Registry	Yes
minorVersion	Integer	Yes	0	Registry	Yes
stability	LongName	No		Client	Yes
status	LongName	Yes		Registry	Yes
userVersion	ShortName	No		Client	Yes

859

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

862 7.5.2 Attribute expiration

863 Each RegistryEntry instance may have an expirationDate. This attribute defines a
864 time limit upon the stability indication provided by the stability attribute. Once the
865 expirationDate has been reached the stability attribute in effect becomes
866 STABILITY_DYNAMIC implying that the repository item can change at any time
867 and in any manner. A null value implies that there is no expiration on stability
868 attribute.

869 7.5.3 Attribute majorVersion

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

874 7.5.4 Attribute minorVersion

875 Each RegistryEntry instance must have a minor revision number for the current
876 version of the RegistryEntry instance. This number is assigned by the registry
877 when the object is created. This number may be updated by the registry when an
878 object is updated.

879 7.5.5 Attribute stability

880 Each RegistryEntry instance may have a stability indicator. The stability indicator
881 is provided by the submitter as an indication of the level of stability for the
882 repository item.

883 7.5.5.1 Pre-defined RegistryEntry Stability Enumerations

884 The following table lists pre-defined choices for RegistryEntry stability attribute.
885 These pre-defined stability types are defined as a *ClassificationScheme*. While
886 the scheme may easily be extended, a *Registry* MAY support the stability types
887 listed below.

888

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.

889

890 **7.5.6 Attribute status**

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

893 **7.5.6.1 Pre-defined RegistryObject Status Types**

894 The following table lists pre-defined choices for RegistryObject status attribute.
895 These pre-defined status types are defined as a *ClassificationScheme*.

896

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

897

898 **7.5.7 Attribute userVersion**

899 Each RegistryEntry instance may have a userVersion. The userVersion is similar
900 to the majorVersion-minorVersion tuple. They both provide an indication of the
901 version of the object. The majorVersion-minorVersion tuple is provided by the
902 registry while userVersion provides a user specified version for the object.
903

904 **7.5.8 Method Summary**

905 In addition to its attributes, the RegistryEntry class also defines the following
906 methods.

Method Summary for RegistryEntry	
Organization	getSubmittingOrganization() Gets the Organization instance of the organization that submitted the given RegistryEntry instance. This method returns a non-null result for every RegistryEntry. For privilege

	assignment, the organization returned by this method is regarded as the owner of the RegistryEntry instance.
Organization	getResponsibleOrganization() Gets the Organization instance of the organization responsible for definition, approval, and/or maintenance of the repository item referenced by the given RegistryEntry instance. This method may return a null result if the submitting organization of this RegistryEntry does not identify a responsible organization or if the registration authority does not assign a responsible organization.

907

908 7.6 Class Slot

909 Slot instances provide a dynamic way to add arbitrary attributes to
910 RegistryObject instances. This ability to add attributes dynamically to
911 RegistryObject instances enables extensibility within the information model.

912

913 A RegistryObject may have 0 or more Slots. A slot is composed of a name, a
914 slotType and a collection of values.

915 7.6.1 Attribute Summary

916

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

917

918 7.6.2 Attribute name

919 Each Slot instance must have a name. The name is the primary means for
920 identifying a Slot instance within a RegistryObject. Consequently, the name of a
921 Slot instance must be locally unique within the RegistryObject *Instance*.

922 7.6.3 Attribute slotType

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

925 7.6.4 Attribute values

926 A Slot instance must have a Collection of values. The collection of values may be
927 empty. Since a Slot represent an extensible attribute whose value may be a

928 collection, therefore a Slot is allowed to have a collection of values rather than a
 929 single value.
 930

931 **7.7 Class ExtrinsicObject**

932 **Super Classes:**

933 [RegistryEntry](#), [RegistryObject](#)

934
 935

936 ExtrinsicObjects provide metadata that describes submitted content whose type
 937 is not intrinsically known to the *Registry* and therefore **MUST** be described by
 938 means of additional attributes (e.g., mime type).
 939

940 Since the registry can contain arbitrary content without intrinsic knowledge about
 941 that content, ExtrinsicObjects require special metadata attributes to provide some
 942 knowledge about the object (e.g., mime type).
 943

944 Examples of content described by ExtrinsicObject include *Collaboration Protocol*
 945 *Profiles* [ebCPP], *Business Process* descriptions, and schemas.

946 **7.7.1 Attribute Summary**

947

Attribute	Data Type	Required	Default Value	Specified By	Mutable
isOpaque	Boolean	No		Client	No
mimeType	LongName	No		Client	No

948

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

951 **7.7.2 Attribute isOpaque**

952 Each ExtrinsicObject instance may have an isOpaque attribute defined. This
 953 attribute determines whether the content catalogued by this ExtrinsicObject is
 954 opaque to (not readable by) the *Registry*. In some situations, a *Submitting*
 955 *Organization* may submit content that is encrypted and not even readable by the
 956 *Registry*.

957 **7.7.3 Attribute mimeType**

958 Each ExtrinsicObject instance may have a mimeType attribute defined. The
 959 mimeType provides information on the type of repository item catalogued by the
 960 ExtrinsicObject instance.
 961

962 **7.8 Class RegistryPackage**

963 **Super Classes:**

OASIS/ebXML Registry Information Model

964 [RegistryEntry](#), [RegistryObject](#)

965
 966 RegistryPackage instances allow for grouping of logically related RegistryObject
 967 instances even if individual member objects belong to different Submitting
 968 Organizations.

969 **7.8.1 Attribute Summary**

970
 971 The RegistryPackage class defines no new attributes other than those that are
 972 inherited from RegistryEntry and RegistryObject base classes. The inherited
 973 attributes are not shown here.

974 **7.8.2 Method Summary**

975 In addition to its attributes, the RegistryPackage class also defines the following
 976 methods.

977

Method Summary of RegistryPackage	
Collection	getMemberObjects() Get the collection of RegistryObject instances that are members of this RegistryPackage.

978

979 **7.9 Class ExternalIdentifier**

980 **Super Classes:**

981 [RegistryObject](#)

982

983 ExternalIdentifier instances provide the additional identifier information to
 984 RegistryObject such as DUNS number, Social Security Number, or an alias
 985 name of the organization. The attribute *identificationScheme* is used to
 986 reference the identification scheme (e.g., "DUNS", "Social Security #"), and the
 987 attribute *value* contains the actual information (e.g., the DUNS number, the social
 988 security number). Each RegistryObject may contain 0 or more ExternalIdentifier
 989 instances.

990 **7.9.1 Attribute Summary**

991

Attribute	Data Type	Required	Default Value	Specified By	Mutable
identificationScheme	UUID	Yes		Client	Yes
registryObject	UUID	Yes		Client	No
value	ShortName	Yes		Client	Yes

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

993 **7.9.2 Attribute identificationScheme**

994 Each ExternalIdentifier instance must have an identificationScheme attribute that
 995 references a ClassificationScheme. This ClassificationScheme defines the
 996 namespace within which an identifier is defined using the value attribute for the
 997 RegistryObject referenced by the RegistryObject attribute.

998 **7.9.3 Attribute registryObject**

999 Each ExternalIdentifier instance must have a RegistryObject attribute that
 1000 references the parent RegistryObject for which this is an ExternalIdentifier.

1001 **7.9.4 Attribute value**

1002 Each ExternalIdentifier instance must have a value attribute that provides the
 1003 identifier value for this ExternalIdentifier (e.g., the actual social security number).

1004 **7.10 Class ExternalLink**

1005 **Super Classes:**

1006 [RegistryObject](#)

1007

1008 ExternalLinks use URIs to associate content in the *Registry* with content that may
 1009 reside outside the *Registry*. For example, an organization submitting a *DTD*
 1010 could use an ExternalLink to associate the *DTD* with the organization's home
 1011 page.

1012 **7.10.1 Attribute Summary**

1013

Attribute	Data Type	Required	Default Value	Specified By	Mutable
externalURI	URI	Yes		Client	Yes

1014

1015 **7.10.2 Attribute externalURI**

1016 Each ExternalLink instance must have an externalURI attribute defined. The
 1017 externalURI attribute provides a URI to the external resource pointed to by this
 1018 ExternalLink instance. If the URI is a URL then a registry must validate the URL
 1019 to be resolvable at the time of submission before accepting an ExternalLink
 1020 submission to the registry.

1021 **7.10.3 Method Summary**

1022 In addition to its attributes, the ExternalLink class also defines the following
 1023 methods.

1024

Method Summary of ExternalLink

Collection [getLinkedObjects\(\)](#)

Gets the collection of RegistryObjects that are linked by this ExternalLink to content outside the registry.

1025

1026 **8 Registry Audit Trail**

1027 This section describes the information model *Elements* that support the audit trail
 1028 capability of the *Registry*. Several *Classes* in this section are *Entity Classes* that
 1029 are used as wrappers to model a set of related attributes. They are analogous to
 1030 the “struct” construct in the C programming language.

1031

1032 The getAuditTrail() method of a RegistryObject returns an ordered Collection of
 1033 AuditableEvents. These AuditableEvents constitute the audit trail for the
 1034 RegistryObject. AuditableEvents include a timestamp for the *Event*. Each
 1035 AuditableEvent has a reference to a User identifying the specific user that
 1036 performed an action that resulted in an AuditableEvent. Each User is affiliated
 1037 with an Organization, which is usually the *Submitting Organization*.

1038 **8.1 Class AuditableEvent**

1039 **Super Classes:**

1040 [RegistryObject](#)

1041

1042 AuditableEvent instances provide a long-term record of *Events* that effect a
 1043 change in a RegistryObject. A RegistryObject is associated with an ordered
 1044 Collection of AuditableEvent instances that provide a complete audit trail for that
 1045 RegistryObject.

1046

1047 AuditableEvents are usually a result of a client-initiated request. AuditableEvent
 1048 instances are generated by the *Registry Service* to log such *Events*.

1049

1050 Often such *Events* effect a change in the life cycle of a RegistryObject. For
 1051 example a client request could Create, Update, Deprecate or Delete a
 1052 RegistryObject. An AuditableEvent is created if and only if a request creates or
 1053 alters the content or ownership of a RegistryObject. Read-only requests do not
 1054 generate an AuditableEvent. No AuditableEvent is generated for a
 1055 RegistryObject when it is classified, assigned to a RegistryPackage or associated
 1056 with another RegistryObject.

1057 **8.1.1 Attribute Summary**

1058

Attribute	Data Type	Required	Default Value	Specified By	Mutable
eventType	LongName	Yes		Registry	No
registryObject	UUID	Yes		Registry	No
timestamp	DateTime	Yes		Registry	No

user	UUID	Yes		Registry	No
------	------	-----	--	----------	----

1059

1060 **8.1.2 Attribute eventType**

1061 Each AuditableEvent must have an eventType attribute which identifies the type
1062 of event recorded by the AuditableEvent.

1063 **8.1.2.1 Pre-defined Auditable Event Types**

1064 The following table lists pre-defined auditable event types. These pre-defined
1065 event types are defined as a pre-defined *ClassificationScheme* with name
1066 "EventType". A *Registry* MUST support the event types listed below.

1067

Name	description
Created	An <i>Event</i> that created a RegistryObject.
Deleted	An <i>Event</i> that deleted a RegistryObject.
Deprecated	An <i>Event</i> that deprecated a RegistryObject.
Updated	An <i>Event</i> that updated the state of a RegistryObject.
Versioned	An <i>Event</i> that versioned a RegistryObject.

1068 **8.1.3 Attribute registryObject**

1069 Each AuditableEvent must have a registryObject attribute that identifies the
1070 RegistryObject instance that was affected by this event.

1071 **8.1.4 Attribute timestamp**

1072 Each AuditableEvent must have a timestamp attribute that records the date and
1073 time that this event occurred.

1074 **8.1.5 Attribute user**

1075 Each AuditableEvent must have a user attribute that identifies the User that sent
1076 the request that generated this event affecting the RegistryObject instance.

1077

1078

1079 **8.2 Class User**

1080 **Super Classes:**

1081 [RegistryObject](#)

1082

1083 User instances are used in an AuditableEvent to keep track of the identity of the
1084 requestor that sent the request that generated the AuditableEvent.

1085 **8.2.1 Attribute Summary**

1086

Attribute	Data Type	Required	Default Value	Specified By	Mutable
address	PostalAddress	Yes		Client	Yes
emailAddresses	Collection of EmailAddress	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

1087

1088 **8.2.2 Attribute address**

1089 Each User instance must have an address attribute that provides the postal
1090 address for that user.

1091 **8.2.3 Attribute emailAddresses**

1092 Each User instance has an attribute emailAddresses that is a Collection of
1093 EmailAddress instances. Each EmailAddress provides an email address for that
1094 user. A User must have at least one email address.

1095 **8.2.4 Attribute organization**

1096 Each User instance must have an organization attribute that references the
1097 Organization instance for the organization that the user is affiliated with.

1098 **8.2.5 Attribute personName**

1099 Each User instance must have a personName attribute that provides the human
1100 name for that user.

1101 **8.2.6 Attribute telephoneNumbers**

1102 Each User instance must have a telephoneNumbers attribute that contains the
1103 Collection of TelephoneNumber instances for each telephone number defined for
1104 that user. A User must have at least one telephone number.

1105 **8.2.7 Attribute url**

1106 Each User instance may have a url attribute that provides the URL address for the web
1107 page associated with that user.

1108 **8.3 Class Organization**1109 **Super Classes:**1110 [RegistryObject](#)

1111

1112 Organization instances provide information on organizations such as a
 1113 *Submitting Organization*. Each *Organization Instance* may have a reference to a
 1114 parent Organization.

1115 **8.3.1 Attribute Summary**

1116

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

1117

1118 **8.3.2 Attribute address**

1119 Each Organization instance must have an address attribute that provides the
 1120 postal address for that organization.

1121 **8.3.3 Attribute parent**

1122 Each Organization instance may have a parent attribute that references the
 1123 parent Organization instance, if any, for that organization.

1124 **8.3.4 Attribute primaryContact**

1125 Each Organization instance must have a primaryContact attribute that references
 1126 the User instance for the user that is the primary contact for that organization.

1127 **8.3.5 Attribute telephoneNumbers**

1128 Each Organization instance must have a telephoneNumbers attribute that
 1129 contains the Collection of TelephoneNumber instances for each telephone
 1130 number defined for that organization. An Organization must have at least one
 1131 telephone number.

1132 **8.4 Class PostalAddress**

1133 PostalAddress is a simple reusable *Entity Class* that defines attributes of a postal
 1134 address.

1135 **8.4.1 Attribute Summary**

1136

Attribute	Data Type	Required	Default Value	Specified By	Mutable
city	ShortName	No		Client	Yes
country	ShortName	No		Client	Yes
postalCode	ShortName	No		Client	Yes

state	ShortName	No		Client	Yes
street	ShortName	No		Client	Yes
streetNumber	String32	No		Client	Yes

1137

1138 **8.4.2 Attribute city**

1139 Each PostalAddress may have a city attribute identifying the city for that address.

1140 **8.4.3 Attribute country**1141 Each PostalAddress may have a country attribute identifying the country for that
1142 address.1143 **8.4.4 Attribute postalCode**1144 Each PostalAddress may have a postalCode attribute identifying the postal code
1145 (e.g., zip code) for that address.1146 **8.4.5 Attribute state**1147 Each PostalAddress may have a state attribute identifying the state, province or
1148 region for that address.1149 **8.4.6 Attribute street**1150 Each PostalAddress may have a street attribute identifying the street name for
1151 that address.1152 **8.4.7 Attribute streetNumber**1153 Each PostalAddress may have a streetNumber attribute identifying the street
1154 number (e.g., 65) for the street address.1155 **8.4.8 Method Summary**1156 In addition to its attributes, the PostalAddress class also defines the following
1157 methods.

1158

Method Summary of ExternalLink	
Collection	getSlots() 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.

1159

1160 **8.5 Class TelephoneNumber**1161 A simple reusable *Entity Class* that defines attributes of a telephone number.

1162 **8.5.1 Attribute Summary**

1163

Attribute	Data Type	Required	Default Value	Specified By	Mutable
areaCode	String4	No		Client	Yes
countryCode	String4	No		Client	Yes
extension	String8	No		Client	Yes
number	String16	No		Client	Yes
phoneType	String32	No		Client	Yes
url	URI	No		Client	Yes

1164

1165 **8.5.2 Attribute areaCode**

1166 Each TelephoneNumber instance may have an areaCode attribute that provides
1167 the area code for that telephone number.

1168 **8.5.3 Attribute countryCode**

1169 Each TelephoneNumber instance may have an countryCode attribute that
1170 provides the country code for that telephone number.

1171 **8.5.4 Attribute extension**

1172 Each TelephoneNumber instance may have an extension attribute that provides
1173 the extension number, if any, for that telephone number.

1174 **8.5.5 Attribute number**

1175 Each TelephoneNumber instance may have a number attribute that provides the
1176 local number (without area code, country code and extension) for that telephone
1177 number.

1178 **8.5.6 Attribute phoneType**

1179 Each TelephoneNumber instance may have phoneType attribute that provides
1180 the type for the TelephoneNumber. Some examples of phoneType are “home”,
1181 “office”.

1182 **8.6 Class EmailAddress**

1183 A simple reusable *Entity Class* that defines attributes of an email address.

1184 **8.6.1 Attribute Summary**

Attribute	Data Type	Required	Default Value	Specified By	Mutable
address	ShortName	Yes		Client	Yes
type	String32	No		Client	Yes

1185 **8.6.2 Attribute address**

1186 Each EmailAddress instance must have an address attribute that provides the
1187 actual email address.

1188 **8.6.3 Attribute type**

1189 Each EmailAddress instance may have a type attribute that provides the type for
1190 that email address. This is an arbitrary value. Examples include “home”, “work”
1191 etc.

1192 **8.7 Class PersonName**

1193 A simple *Entity Class* for a person’s name.

1194 **8.7.1 Attribute Summary**

1195

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

1196 **8.7.2 Attribute firstName**

1197 Each PersonName may have a firstName attribute that is the first name of the
1198 person.

1199 **8.7.3 Attribute lastName**

1200 Each PersonName may have a lastName attribute that is the last name of the
1201 person.

1202 **8.7.4 Attribute middleName**

1203 Each PersonName may have a middleName attribute that is the middle name of the
1204 person.

1205 **8.8 Class Service**1206 **Super Classes:**

1207 [RegistryEntry](#), [RegistryObject](#)

1208

1209 Service instances provide information on services, such as web services.

1210 **8.8.1 Attribute Summary**

1211 The Service class does not define any specialized attributes other than its
1212 inherited attributes.

1213 8.8.2 Method Summary

1214 In addition to its attributes, the Service class also defines the following methods.
1215

Method Summary of Service	
Collection	getServiceBindings() Gets the collection of ServiceBinding instances defined for this Service.

1216 8.9 Class ServiceBinding

1217 Super Classes:

1218 [RegistryObject](#)

1219
1220 ServiceBinding instances are RegistryObjects that represent technical
1221 information on a specific way to access a specific interface offered by a Service
1222 instance. A Service has a Collection of ServiceBindings.
1223 The description attribute of ServiceBinding provides details about the relationship
1224 between several specification links comprising the Service Binding. This
1225 description can be useful for human understanding such that the runtime system
1226 can be appropriately configured by the human being. There is possibility of
1227 enforcing a structure on this description for enabling machine processing of the
1228 Service Binding, which is however not addressed by the current document.
1229
1230

1231 8.9.1 Attribute Summary

1232

Attribute	Data Type	Required	Default Value	Specified By	Mutable
accessURI	URI	No		Client	Yes
targetBinding	UUID	No		Client	Yes

1233

1234 8.9.2 Attribute accessURI

1235 A ServiceBinding may have an accessURI attribute that defines the URI to
1236 access that ServiceBinding. This attribute is ignored if a targetBinding attribute is
1237 specified for the ServiceBinding. If the URI is a URL then a registry must validate
1238 the URL to be resolvable at the time of submission before accepting a
1239 ServiceBinding submission to the registry.

1240 8.9.3 Attribute targetBinding

1241 A ServiceBinding may have a targetBinding attribute defined which references
1242 another ServiceBinding. A targetBinding may be specified when a service is
1243 being redirected to another service. This allows the rehosting of a service by
1244 another service provider.

1245 **8.9.4 Method Summary**

1246 In addition to its attributes, the ServiceBinding class also defines the following
1247 methods.
1248

Method Summary of ServiceBinding	
Collection	getSpecificationLinks() Get the collection of SpecificationLink instances defined for this ServiceBinding.

1249
1250
1251

1252 **8.10 Class SpecificationLink**

1253 **Super Classes:**

1254 [RegistryObject](#)

1255

1256 A SpecificationLink provides the linkage between a ServiceBinding and one of its
1257 technical specifications that describes how to use the service using the
1258 ServiceBinding. For example, a ServiceBinding may have a SpecificationLink
1259 instances that describe how to access the service using a technical specification
1260 in form of a WSDL document or a CORBA IDL document.

1261 **8.10.1 Attribute Summary**

1262

Attribute	Data Type	Required	Default Value	Specified By	Mutable
specificationObject	UUID	Yes		Client	Yes
usageDescription	InternationalString	No		Client	Yes
usageParameters	Collection of FreeFormText	No		Client	Yes

1263

1264 **8.10.2 Attribute specificationObject**

1265 A SpecificationLink instance must have a specificationObject attribute that
1266 provides a reference to a RegistryObject instance that provides a technical
1267 specification for the parent ServiceBinding. Typically, this is an ExtrinsicObject
1268 instance representing the technical specification (e.g., a WSDL document).

1269 **8.10.3 Attribute usageDescription**

1270 A SpecificationLink instance may have a usageDescription attribute that provides
1271 a textual description of how to use the optional usageParameters attribute
1272 described next. The usageDescription is of type InternationalString, thus allowing
1273 the description to be in multiple languages.

1274 8.10.4 Attribute usageParameters

1275 A SpecificationLink instance may have a usageParameters attribute that provides
1276 a collection of Strings representing the instance specific parameters needed to
1277 use the technical specification (e.g., a WSDL document) specified by this
1278 SpecificationLink object.
1279

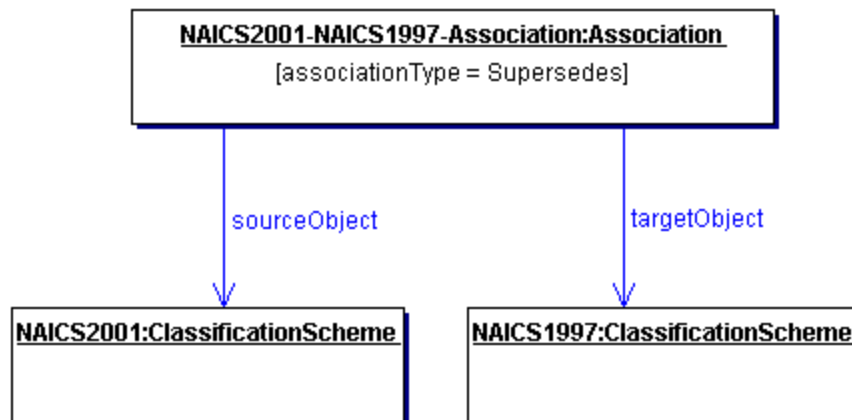
1279 9 Association of Registry Objects

1280 A RegistryObject instance may be *associated* with zero or more RegistryObject
 1281 instances. The information model defines an Association class, an instance of
 1282 which may be used to associate any two RegistryObject instances.

1283 9.1 Example of an Association

1284 One example of such an association is between two ClassificationScheme
 1285 instances, where one ClassificationScheme supersedes the other
 1286 ClassificationScheme as shown in [Figure 3](#). This may be the case when
 1287 a new version of a ClassificationScheme is submitted.

1288 In [Figure 3](#), we see how an Association is defined between a new
 1289 version of the NAICS ClassificationScheme and an older version of the NAICS
 1290 ClassificationScheme.
 1291



1292

1293

Figure 3: Example of RegistryObject Association

1294 9.2 Source and Target Objects

1295 An Association instance represents an association between a *source*
 1296 RegistryObject and a *target* RegistryObject. These are referred to as
 1297 *sourceObject* and *targetObject* for the Association instance. It is important which
 1298 object is the *sourceObject* and which is the *targetObject* as it determines the
 1299 directional semantics of an Association.

1300 In the example in [Figure 3](#), it is important to make the newer version of
 1301 NAICS ClassificationScheme be the *sourceObject* and the older version of
 1302 NAICS be the *targetObject* because the *associationType* implies that the
 1303 *sourceObject* supersedes the *targetObject* (and not the other way around).

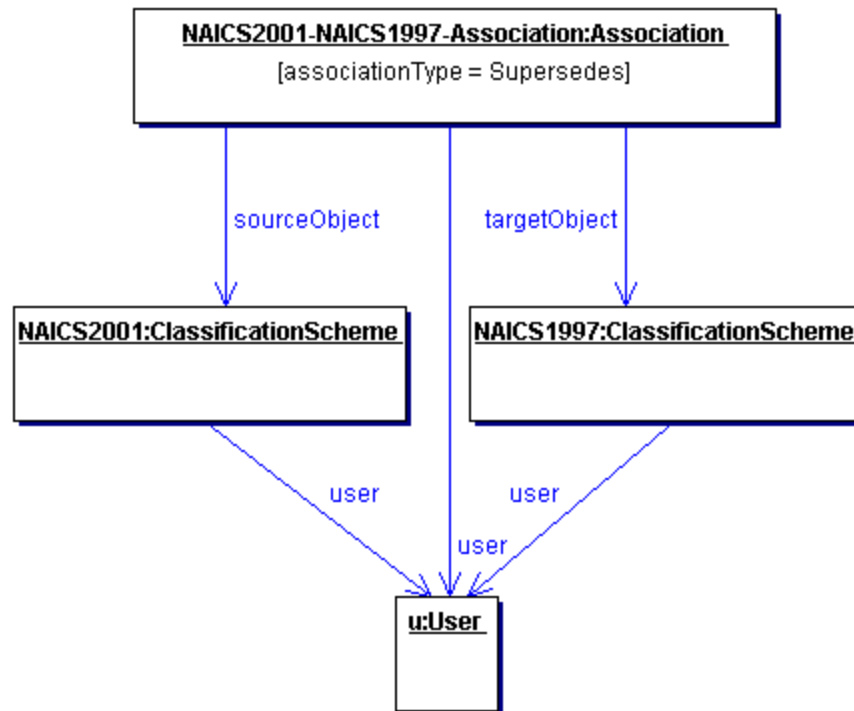
1304 9.3 Association Types

1305 Each Association must have an *associationType* attribute that identifies the type
 1306 of that association.

1307 9.4 Intramural Association

1308 A common use case for the Association class is when a User “u” creates an
 1309 Association “a” between two RegistryObjects “o1” and “o2” where association “a”
 1310 and RegistryObjects “o1” and “o2” are objects that were created by the same
 1311 User “u.” This is the simplest use case, where the association is between two
 1312 objects that are owned by the same User that is defining the Association. Such
 1313 associations are referred to as *intramural associations*.
 1314 [Figure 4](#) below, extends the previous example in [Figure 3](#) for the
 1315 intramural association case.

1316



1317

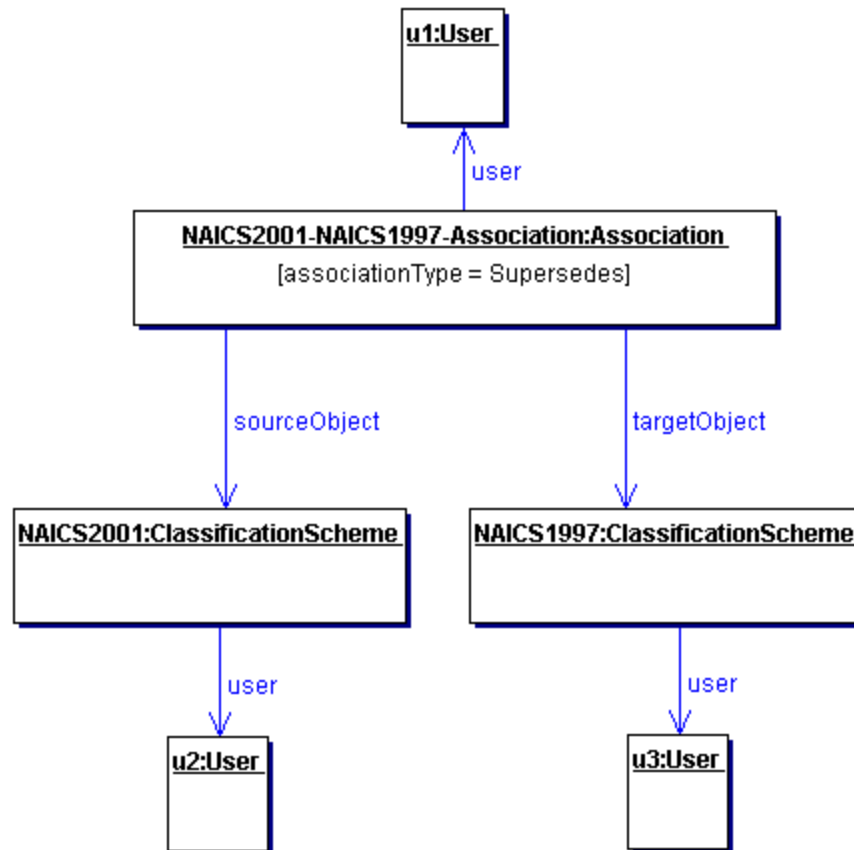
1318

Figure 4: Example of Intramural Association

1319 9.5 Extramural Association

1320 The information model also allows more sophisticated use cases. For example, a
 1321 User “u1” creates an Association “a” between two RegistryObjects “o1” and “o2”
 1322 where association “a” is owned by User “u1”, but RegistryObjects “o1” and “o2”
 1323 are owned by User “u2” and User “u3” respectively.
 1324 In this use case an Association is defined where either or both objects that are
 1325 being associated are owned by a User different from the User defining the
 1326 Association. Such associations are referred to as *extramural associations*. The
 1327 Association class provides a convenience method called `isExtramural` that
 1328 returns “true” if the Association instance is an extramural Association.

1329 [Figure 5](#) below, extends the previous example in [Figure 3](#) for the
 1330 extramural association case. Note that it is possible for an extramural association
 1331 to have two distinct Users rather than three distinct Users as shown in [Figure](#)
 1332 [5](#). In such case, one of the two users owns two of the three objects
 1333 involved (Association, sourceObject and targetObject).
 1334



1335
 1336

Figure 5: Example of Extramural Association

1337 9.6 Confirmation of an Association

1338 An association may need to be confirmed by the parties whose objects are
 1339 involved in that Association as the sourceObject or targetObject. This section
 1340 describes the semantics of confirmation of an association by the parties involved.

1341 9.6.1 Confirmation of Intramural Associations

1342 Intramural associations may be viewed as declarations of truth and do not
 1343 require any explicit steps to confirm that Association as being true. In other
 1344 words, intramural associations are implicitly considered confirmed.

1345 **9.6.2 Confirmation of Extramural Associations**

1346 Extramural associations may be thought of as a unilateral assertion that may not
1347 be viewed as truth until it has been confirmed by the other (extramural) parties
1348 involved (Users “u2” and “u3” in the example in section 9.5).
1349 To confirm an extramural association, each of the extramural parties (parties that
1350 own the source or target object but do not own the Association) must submit an
1351 identical Association (clone Association) as the Association they are intending to
1352 confirm using a SubmitObjectsRequest. The clone Association must have the
1353 same id as the original Association.

1354 **9.6.3 Deleting an Extramural Associations**

1355 An Extramural Association is deleted like any other type of RegistryObject, using
1356 the RemoveObjectsRequest as defined in [ebRS]. However, in some cases
1357 deleting an extramural Association may not actually delete it but instead only
1358 revert a confirmed association to unconfirmed state.

1359
1360 An Association must always be deleted when deleted by the owner of that
1361 Association, irrespective of its confirmation state. An extramural Association must
1362 become unconfirmed by the owner of its source/target object when deleted by
1363 the owner of its source/target object when the requestor is not the owner of the
1364 Association itself.

1365 **9.7 Visibility of Unconfirmed Associations**

1366 Extramural associations require each extramural party to confirm the assertion
1367 being made by the extramural Association before the Association is visible to
1368 third parties that are not involved in the Association. This ensures that
1369 unconfirmed Associations are not visible to third party registry clients.

1370 **9.8 Possible Confirmation States**

1371 Assume the most general case where there are three distinct User instances as
1372 shown in ~~Figure 5~~ ~~Figure 5~~ for an extramural Association. The extramural
1373 Association needs to be confirmed by both the other (extramural) parties (Users
1374 “u2” and “u3” in example) in order to be fully confirmed. The methods
1375 `isConfirmedBySourceOwner` and `isConfirmedByTargetOwner` in the
1376 Association class provide access to the confirmation state for both the
1377 sourceObject and targetObject. A third convenience method called
1378 `isConfirmed` provides a way to determine whether the Association is fully
1379 confirmed or not. So there are the following four possibilities related to the
1380 confirmation state of an extramural Association:

- 1381 ○ The Association is confirmed neither by the owner of the sourceObject nor
1382 by the owner of the targetObject.
- 1383 ○ The Association is confirmed by the owner of the sourceObject but it is not
1384 confirmed by the owner of the targetObject.
- 1385 ○ The Association is not confirmed by the owner of the sourceObject but it is
1386 confirmed by the owner of the targetObject.

- 1387 ○ The Association is confirmed by both the owner of the sourceObject and
 1388 the owner of the targetObject. This is the only state where the Association
 1389 is fully confirmed.
 1390

1391 **9.9 Class Association**

1392 **Super Classes:**

1393 [RegistryObject](#)

1394
 1395

1396 Association instances are used to define many-to-many associations among
 1397 RegistryObjects in the information model.

1398

1399 An *Instance* of the Association *Class* represents an association between two
 1400 RegistryObjects.

1401 **9.9.1 Attribute Summary**

1402

Attribute	Data Type	Required	Default Value	Specified By	Mutable
associationType	LongName	Yes		Client	No
sourceObject	UUID	Yes		Client	No
targetObject	UUID	Yes		Client	No
IsConfirmedBy-SourceOwner	boolean	No	false	Registry	No
IsConfirmedBy-TargetOwner	boolean	No	false	Registry	No

1403

1404 **9.9.2 Attribute associationType**

1405 Each Association must have an associationType attribute that identifies the type
 1406 of that association.

1407 **9.9.2.1 Pre-defined Association Types**

1408 The following table lists pre-defined association types. These pre-defined
 1409 association types are defined as a *Classification* scheme. While the scheme may
 1410 easily be extended a *Registry* MUST support the association types listed below.

1411

name	description
RelatedTo	Defines that source RegistryObject is related to target RegistryObject.
HasMember	Defines that the source RegistryPackage object has the target RegistryObject object as a member. Reserved for use in Packaging of RegistryEntries.

ExternallyLinks	Defines that the source ExternalLink object externally links the target RegistryObject object. Reserved for use in associating ExternalLinks with RegistryEntries.
Contains	Defines that source RegistryObject contains the target RegistryObject. The details of the containment relationship are specific to the usage. For example a parts catalog may define an Engine object to have a contains relationship with a Transmission object.
EquivalentTo	Defines that source RegistryObject is equivalent to the target RegistryObject.
Extends	Defines that source RegistryObject inherits from or specializes the target RegistryObject.
Implements	Defines that source RegistryObject implements the functionality defined by the target RegistryObject.
InstanceOf	Defines that source RegistryObject is an <i>Instance</i> of target RegistryObject.
Supersedes	Defines that the source RegistryObject supersedes the target RegistryObject.
Uses	Defines that the source RegistryObject uses the target RegistryObject in some manner.
Replaces	Defines that the source RegistryObject replaces the target RegistryObject in some manner.
SubmitterOf	Defines that the source Organization is the submitter of the target RegistryObject.
ResponsibleFor	Defines that the source Organization is responsible for the ongoing maintenance of the target RegistryObject.
OffersService	Defines that the source Organization object offers the target Service object as a service. Reserved for use in indicating that an Organization offers a Service.

1412

1413 **9.9.3 Attribute sourceObject**

1414 Each Association must have a sourceObject attribute that references the
1415 RegistryObject instance that is the source of that association.

1416 **9.9.4 Attribute targetObject**

1417 Each Association must have a targetObject attribute that references the
1418 RegistryObject instance that is the target of that association.

1419 **9.9.5 Attribute isConfirmedBySourceOwner**

1420 Each Association may have a n isConfirmedBySourceOwner attribute that is set
1421 by the registry to be true if the association has been confirmed by the owner of

1422 the sourceObject. For intramural Associations this attribute is always true. This
 1423 attribute must be present when the object is retrieved from the registry. This
 1424 attribute must be ignored if specified by the client when the object is submitted to
 1425 the registry.

1426 **9.9.6 Attribute isConfirmedByTargetOwner**

1427 Each Association may have an isConfirmedByTargetOwner attribute that is set
 1428 by the registry to be true if the association has been confirmed by the owner of
 1429 the targetObject. For intramural Associations this attribute is always true. This
 1430 attribute must be present when the object is retrieved from the registry. This
 1431 attribute must be ignored if specified by the client when the object is submitted to
 1432 the registry.
 1433

Method Summary of Association	
Boolean Boolean <u>lean</u>	<u>isConfirmed</u> () Returns true if isConfirmedBySourceOwner and isConfirmedByTargetOwner <u>attributes are</u> both return true. For intramural Associations always return true. An association should only be visible to third parties (not involved with the Association) if isConfirmed returns true.
Boolean Boolean <u>lean</u>	<u>isExtramural</u> () Returns true if the sourceObject and/or the targetObject are owned by a User that is different from the User that created the Association.

1434

1435 **10 Classification of RegistryObject**

1436 This section describes the how the information model supports *Classification of*
 1437 *RegistryObject*. It is a simplified version of the *OASIS* classification model [OAS].

1438

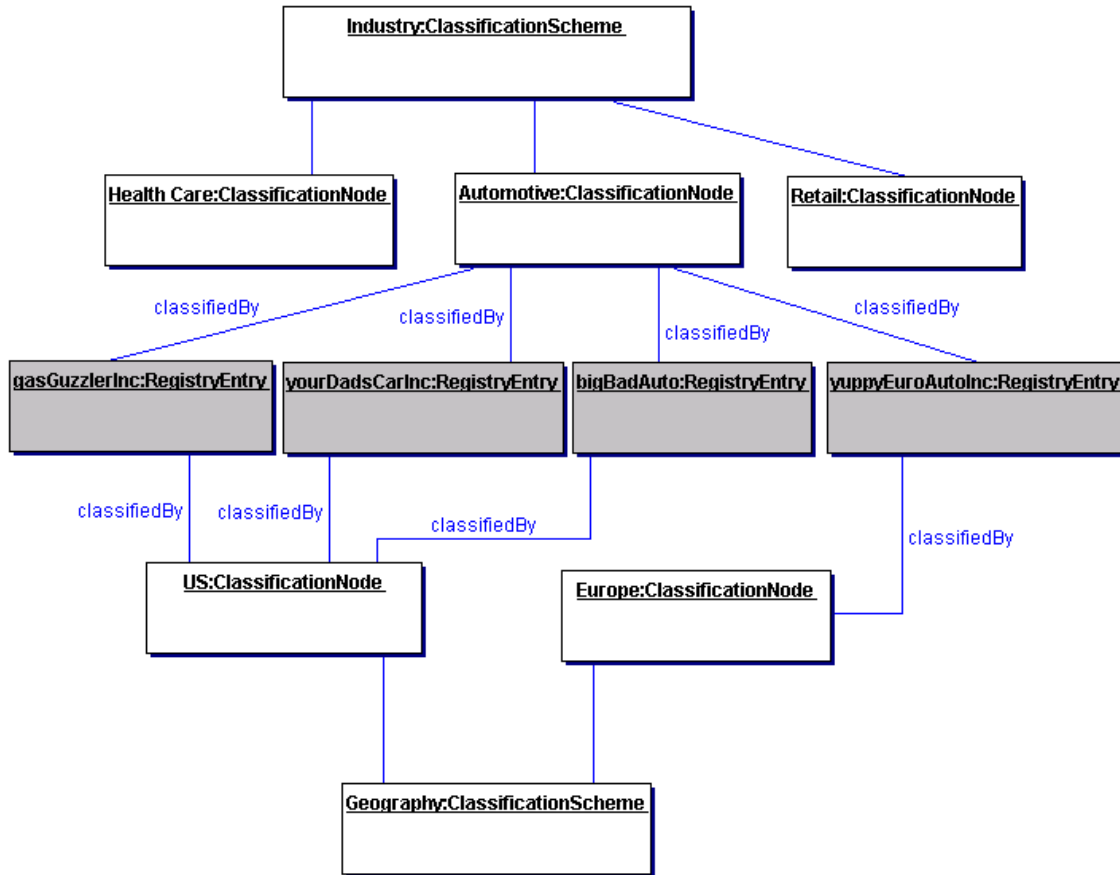
1439 A RegistryObject may be classified in many ways. For example the
 1440 RegistryObject for the same *Collaboration Protocol Profile (CPP)* may be
 1441 classified by its industry, by the products it sells and by its geographical location.

1442

1443 A general *ClassificationScheme* can be viewed as a *Classification* tree. In the
 1444 example shown in Figure 6~~Figure-6~~, RegistryObject instances representing
 1445 *Collaboration Protocol Profiles* are shown as shaded boxes. Each *Collaboration*
 1446 *Protocol Profile* represents an automobile manufacturer. Each *Collaboration*
 1447 *Protocol Profile* is classified by the ClassificationNode named "Automotive" under
 1448 the ClassificationScheme instance with name "Industry." Furthermore, the US
 1449 Automobile manufacturers are classified by the US ClassificationNode under the
 1450 ClassificationScheme with name "Geography." Similarly, a European automobile
 1451 manufacturer is classified by the "Europe" ClassificationNode under the
 1452 ClassificationScheme with name "Geography."

1453

1454 The example shows how a RegistryObject may be classified by multiple
 1455 ClassificationNode instances under multiple ClassificationScheme instances
 1456 (e.g., Industry, Geography).
 1457

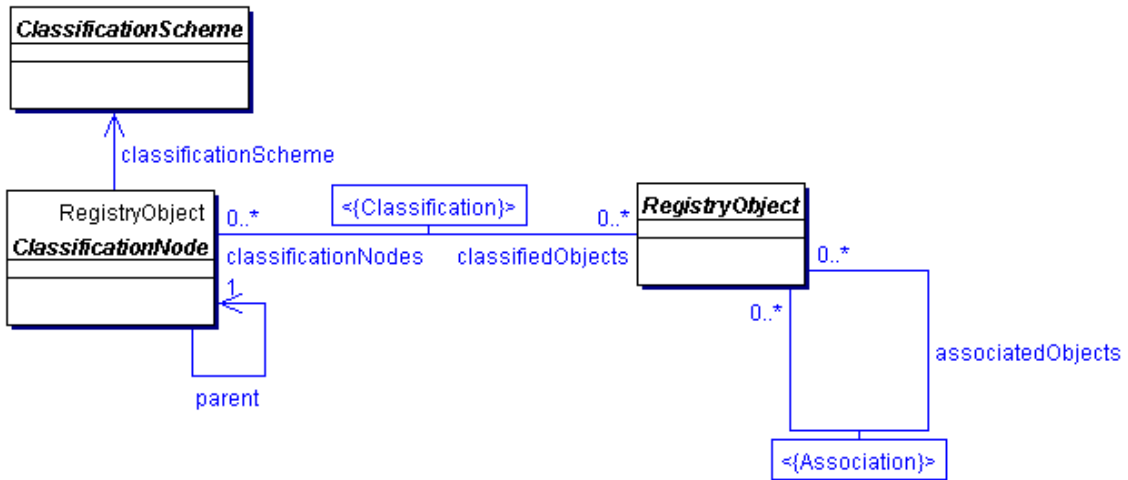


1458
 1459

Figure 6: Example showing a Classification Tree

1460 [Note]It is important to point out that the dark
 1461 nodes (gasGuzzlerInc, yourDadsCarInc etc.) are
 1462 not part of the Classification tree. The leaf
 1463 nodes of the Classification tree are Health
 1464 Care, Automotive, Retail, US and Europe. The
 1465 dark nodes are associated with the
 1466 Classification tree via a Classification
 1467 Instance that is not shown in the picture
 1468

1469 In order to support a general Classification scheme that can support single level
 1470 as well as multi-level Classifications, the information model defines the Classes
 1471 and relationships shown in [Figure 7](#).



1472

1473

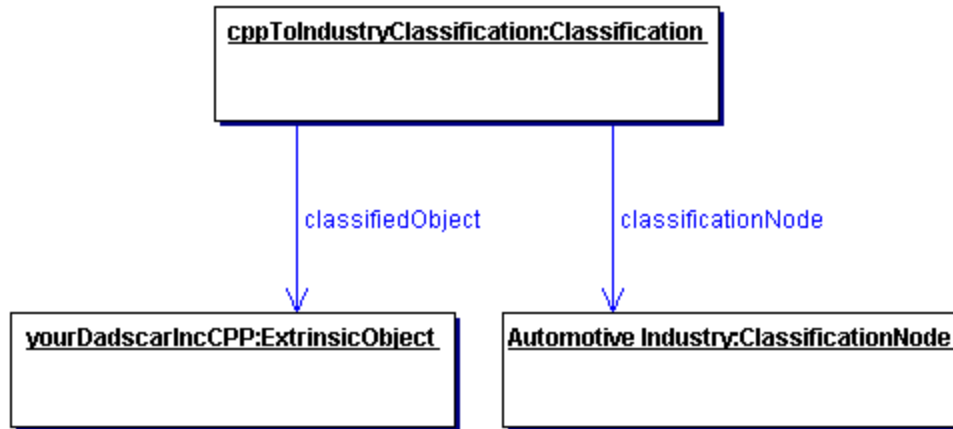
Figure 7: Information Model *Classification View*

1474

1475

1476 A Classification is somewhat like a specialized form of an Association. [Figure](#)
 1477 [8](#) shows an example of an *ExtrinsicObject Instance* for a *Collaboration*
 1478 *Protocol Profile (CPP)* object that is classified by a *ClassificationNode*
 1479 representing the Industry that it belongs to.

1480



1481

1482

Figure 8: Classification *Instance Diagram*

1483

1484

1485

1486

1487

1488

1489 **10.1 Class ClassificationScheme**

1490 **Base classes:**

1491 [RegistryEntry](#), [RegistryObject](#)

1492

1493 A ClassificationScheme instance is metadata that describes a registered
1494 taxonomy. The taxonomy hierarchy may be defined internally to the
1495 Registry by instances of ClassificationNode or it may be defined externally
1496 to the Registry, in which case the structure and values of the taxonomy
1497 elements are not known to the Registry.

1498 In the first case the classification scheme is defined to be *internal* and in
1499 the second case the classification scheme is defined to be *external*.

1500 The ClassificationScheme class inherits attributes and methods from the
1501 RegistryObject and RegistryEntry classes.

1502

1503 **10.1.1 Attribute Summary**

1504

Attribute	Data Type	Required	Default Value	Specified By	Mutable
isInternal	Boolean	Yes		Client	No
nodeType	String32	Yes		Client	No

1505 Note that attributes inherited by ClassificationScheme class from the
1506 RegistryEntry class are not shown.

1507

1508 **10.1.2 Attribute isInternal**

1509 When submitting a ClassificationScheme instance the Submitting Organization
1510 needs to declare whether the ClassificationScheme instance represents an
1511 internal or an external taxonomy. This allows the registry to validate the
1512 subsequent submissions of ClassificationNode and Classification instances in
1513 order to maintain the type of ClassificationScheme consistent throughout its
1514 lifecycle.

1515

1516 **10.1.3 Attribute nodeType**

1517 When submitting a ClassificationScheme instance the Submitting Organization
1518 needs to declare what is the structure of taxonomy nodes that this
1519 ClassificationScheme instance will represent. This attribute is an enumeration
1520 with the following values:

- 1521 - UniqueCode. This value says that each node of the taxonomy has
1522 a unique code assigned to it.
- 1523 - EmbeddedPath. This value says that a unique code assigned to
1524 each node of the taxonomy at the same time encodes its path. This
1525 is the case in the NAICS taxonomy.

1526 - NonUniqueCode. In some cases nodes are not unique, and it is
 1527 necessary to nominate the full path in order to identify the node. For
 1528 example, in a geography taxonomy Moscow could be under both
 1529 Russia and the USA, where there are five cities of that name in
 1530 different states.

1531 This enumeration might expand in the future with some new values. An example
 1532 for possible future values for this enumeration might be NamedPathElements for
 1533 support of Named-Level taxonomies such as Genus/Species.
 1534

1535 10.2 Class ClassificationNode

1536 **Base classes:**

1537 [RegistryObject](#)

1538 ClassificationNode instances are used to define tree structures where
 1539 each node in the tree is a ClassificationNode. Such *Classification* trees
 1540 are constructed with ClassificationNode instances under a
 1541 ClassificationScheme instance, and are used to define *Classification*
 1542 schemes or ontologies.
 1543
 1544

1545 10.2.1 Attribute Summary

1546

Attribute	Data Type	Required	Default Value	Specified By	Mutable
parent	UUID	No		Client	No
code	ShortName	No		Client	No
path	String	No		Registry	No

1547

1548 10.2.2 Attribute parent

1549 Each ClassificationNode may have a parent attribute. The parent attribute either
 1550 references a parent ClassificationNode or a ClassificationScheme instance in
 1551 case of first level ClassificationNode instances.
 1552

1553 10.2.3 Attribute code

1554 Each ClassificationNode may have a code attribute. The code attribute
 1555 contains a code within a standard coding scheme.

1556 **10.2.4 Attribute path**

1557 **Each ClassificationNode may have a path attribute. The path attribute must be**
 1558 **present when a ClassificationNode is retrieved from the registry. The path**
 1559 **attribute must be ignored when the path is specified by the client when the object**

1560 [is submitted to the registry. The path attribute contains the canonical path from](#)
 1561 [the ClassificationScheme of this ClassificationNode. The path syntax is defined](#)
 1562 [in 10.2.6.](#)

1563 **10.2.410.2.5 Method Summary**

1564 In addition to its attributes, the ClassificationNode class also defines the following
 1565 methods.

1566

Method Summary of ClassificationNode	
ClassificationScheme	getClassificationScheme () Get the ClassificationScheme that this ClassificationNode belongs to.
Collection	getClassifiedObjects () Get the collection of RegistryObjects classified by this ClassificationNode.
Integer	getLevelNumber () Gets the level number of this ClassificationNode in the classification scheme hierarchy. This method returns a positive integer and is defined for every node instance.

1567

1568 In [Figure 6](#)~~Figure-6~~, several instances of ClassificationNode are defined (all light
 1569 colored boxes). A ClassificationNode has zero or one parent and zero or more
 1570 ClassificationNodes for its immediate children. The parent of a
 1571 ClassificationNode may be another ClassificationNode or a ClassificationScheme
 1572 in case of first level ClassificationNodes.

1573

1574 **10.2.510.2.6 Canonical Path Syntax**

1575 The [getPath](#) ~~method~~[path attribute](#) of the ClassificationNode class **returns**
 1576 **contains** an absolute path in a canonical representation that uniquely identifies
 1577 the path leading from the ClassificationScheme to that ClassificationNode.

1578 The canonical path representation is defined by the following BNF grammar:

1579

```

1580 canonicalPath ::= '/' schemeld nodePath
1581 nodePath    ::=  '/' nodeCode
1582             |   '/' nodeCode ( nodePath )?
  
```

1583

1584 In the above grammar, schemeld is the id attribute of the ClassificationScheme
 1585 instance, and nodeCode is defined by NCName production as defined by
 1586 <http://www.w3.org/TR/REC-xml-names/#NT-NCName>.

1587

1588 [10.2.5.110.2.6.1](#) **Example of Canonical Path Representation**
 1589 The following canonical path represents what the `getPath-path-method-attribute`
 1590 would **return-contain** for the ClassificationNode with code 'United States' in the
 1591 sample Geography scheme in section [10.2.6.210.2.5.2](#).

1592
 1593 `/Geography-id/NorthAmerica/UnitedStates`

1594 [10.2.5.210.2.6.2](#) **Sample Geography Scheme**

1595 Note that in the following examples, the ID attributes have been chosen for ease
 1596 of readability and are therefore not valid URN or UUID values.

```
1597 <ClassificationScheme id='Geography-id' name="Geography"/>
1598 <ClassificationNode id="NorthAmerica-id" parent="Geography-id" code="NorthAmerica" />
1599 <ClassificationNode id="UnitedStates-id" parent="NorthAmerica-id" code="UnitedStates" />
1600 <ClassificationNode id="Asia-id" parent="Geography-id" code="Asia" />
1601 <ClassificationNode id="Japan-id" parent="Asia-id" code="Japan" />
1602 <ClassificationNode id="Tokyo-id" parent="Japan-id" code="Tokyo" />
```

1607 10.3 Class Classification

1608 Base Classes:

1609 [RegistryObject](#)

1610
 1611 A Classification instance classifies a RegistryObject instance by referencing a
 1612 node defined within a particular classification scheme. An internal classification
 1613 will always reference the node directly, by its id, while an external classification
 1614 will reference the node indirectly by specifying a representation of its value that is
 1615 unique within the external classification scheme.

1616
 1617 The attributes and methods for the Classification class are intended to allow for
 1618 representation of both internal and external classifications in order to minimize
 1619 the need for a submission or a query to distinguish between internal and external
 1620 classifications.

1621
 1622 In [Figure 6](#), Classification instances are not explicitly shown but are
 1623 implied as associations between the RegistryObject instances (shaded leaf node)
 1624 and the associated ClassificationNode.

1625 10.3.1 Attribute Summary

1626

Attribute	Data Type	Required	Default Value	Specified By	Mutable
classificationScheme	UUID	for external classifications	null	Client	No
classificationNode	UUID	for internal	null	Client	No

		classifications			
classifiedObject	UUID	Yes		Client	No
nodeRepresentation	LongName	for external classifications	null	Client	No

1627 Note that attributes inherited from the base classes of this class are not shown.
1628

1629 **10.3.2 Attribute classificationScheme**

1630 If the Classification instance represents an external classification, then the
1631 classificationScheme attribute is required. The classificationScheme value must
1632 reference a ClassificationScheme instance.
1633

1634 **10.3.3 Attribute classificationNode**

1635 If the Classification instance represents an internal classification, then the
1636 classificationNode attribute is required. The classificationNode value must
1637 reference a ClassificationNode instance.

1638 **10.3.4 Attribute classifiedObject**

1639 For both internal and external classifications, the ClassifiedObject attribute is
1640 required and it references the RegistryObject instance that is classified by this
1641 Classification.
1642

1643 **10.3.5 Attribute nodeRepresentation**

1644 If the Classification instance represents an external classification, then the
1645 nodeRepresentation attribute is required. It is a representation of a taxonomy
1646 element from a classification scheme. It is the responsibility of the registry to
1647 distinguish between different types of nodeRepresentation, like between the
1648 classification scheme node code and the classification scheme node canonical
1649 path. This allows client to transparently use different syntaxes for
1650 nodeRepresentation.

1651 **10.3.6 Context Sensitive Classification**

1652 Consider the case depicted in [Figure 9](#) where a *Collaboration Protocol Profile*
1653 for ACME Inc. is classified by the Japan ClassificationNode under the
1654 Geography *Classification* scheme. In the absence of the context for this
1655 *Classification* its meaning is ambiguous. Does it mean that ACME is located in
1656 Japan, or does it mean that ACME ships products to Japan, or does it have some
1657 other meaning? To address this ambiguity a Classification may optionally be
1658 associated with another ClassificationNode (in this example named isLocatedIn)
1659 that provides the missing context for the Classification. Another *Collaboration*
1660 *Protocol Profile* for MyParcelService may be classified by the Japan
1661 ClassificationNode where this Classification is associated with a different

1662 ClassificationNode (e.g., named shipsTo) to indicate a different context than the
 1663 one used by ACME Inc.

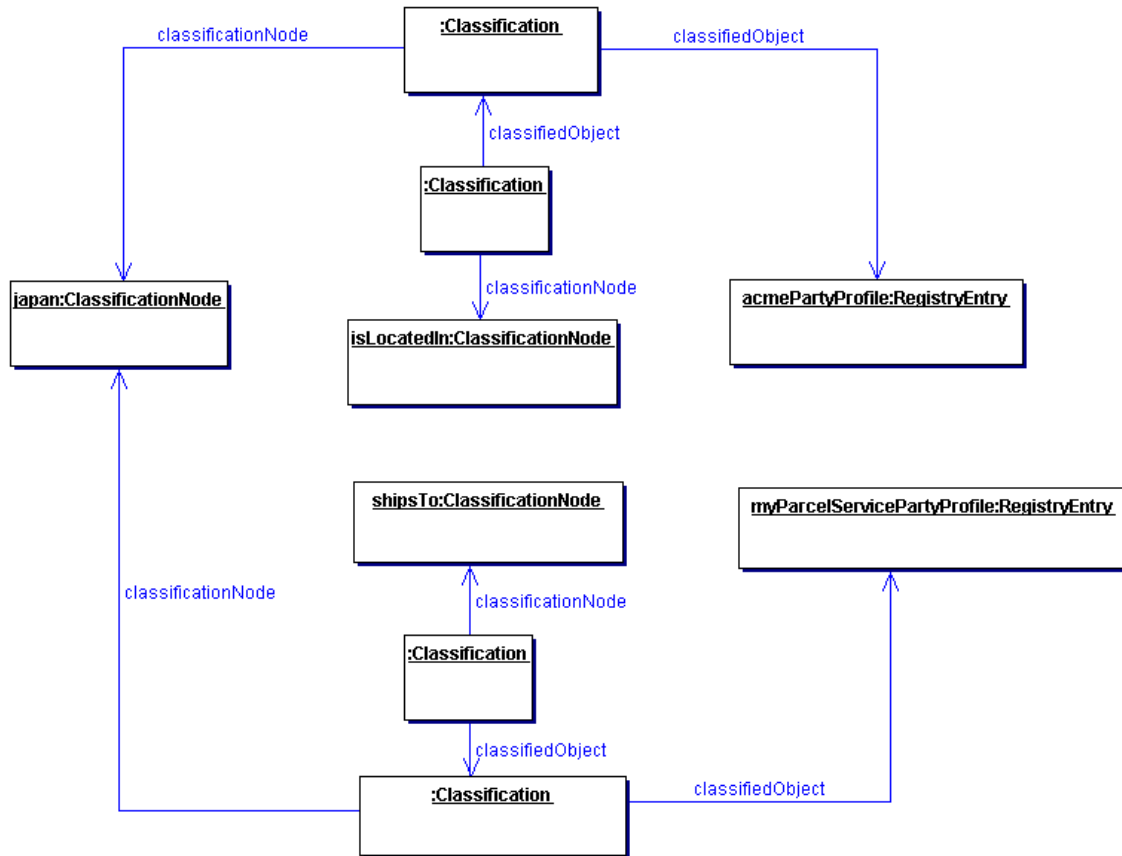


Figure 9: Context Sensitive Classification

1664
 1665

1666
 1667
 1668

1669 Thus, in order to support the possibility of Classification within multiple contexts,
 1670 a Classification is itself classified by any number of Classifications that bind the
 1671 first Classification to ClassificationNodes that provide the missing contexts.
 1672

1673 In summary, the generalized support for *Classification* schemes in the
 1674 information model allows:

- 1675 ○ A RegistryObject to be classified by defining an internal Classification that
- 1676 associates it with a ClassificationNode in a *ClassificationScheme*.
- 1677 ○ A RegistryObject to be classified by defining an external Classification that
- 1678 associates it with a value in an external *ClassificationScheme*.
- 1679 ○ A RegistryObject to be classified along multiple facets by having multiple
- 1680 *Classifications* that associate it with multiple ClassificationNodes or value
- 1681 within a *ClassificationScheme*.
- 1682 ○ A *Classification* defined for a RegistryObject to be qualified by the
- 1683 contexts in which it is being classified.

1684
1685

1686 **10.3.7 Method Summary**

1687 In addition to its attributes, the Classification class also defines the following
1688 methods:

Return Type	Method
UUID	<p>getClassificationScheme()</p> <p>For an external classification, returns the scheme identified by the classificationScheme attribute. For an internal classification, returns the scheme identified by the same method applied to the ClassificationNode instance</p>
String	<p>getPath()</p> <p>For an external classification returns a string that conforms to the string structure specified for the result of the getPath() method attribute in the ClassificationNode class. For an internal classification, returns the same value as does contained in the getPath() attribute method of applied to the ClassificationNode instance identified by the classificationNode attribute.</p>
ShortName	<p>getCode()</p> <p>For an external classification, returns a string that represents the declared value of the taxonomy element. It will not necessarily uniquely identify that node. For an internal classification, returns the value of the code attribute of the ClassificationNode instance identified by the classificationNode attribute.</p>
Organization	<p>getSubmittingOrganization()</p> <p>Gets the Organization instance of the organization that submitted the given RegistryEntry instance. This method returns a non-null result for every RegistryEntry. For privilege assignment, the organization returned by this method is regarded as the owner of the Classification instance.</p>

1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699

1700

1701 **10.4 Example of Classification Schemes**

1702 The following table lists some examples of possible *Classification* schemes
 1703 enabled by the information model. These schemes are based on a subset of
 1704 contextual concepts identified by the ebXML Business Process and Core
 1705 Components Project Teams. This list is meant to be illustrative not prescriptive.

1706

1707

Classification Scheme	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>Business</i> that sells a product or offers a service	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"	

1708

Table 1: Sample *Classification* Schemes

1709

1710

1711

1712

1713

1714

1715

1716 **11 Information Model: Security View**

1717 This section describes the aspects of the information model that relate to the
 1718 security features of the *Registry*.

1719

1720 ~~Figure 10~~Figure-10 shows the view of the objects in the *Registry* from a security
 1721 perspective. It shows object relationships as a *UML Class* diagram. It does not
 1722 show *Class* attributes or *Class* methods that will be described in subsequent
 1723 sections. It is meant to be illustrative not prescriptive.

1724

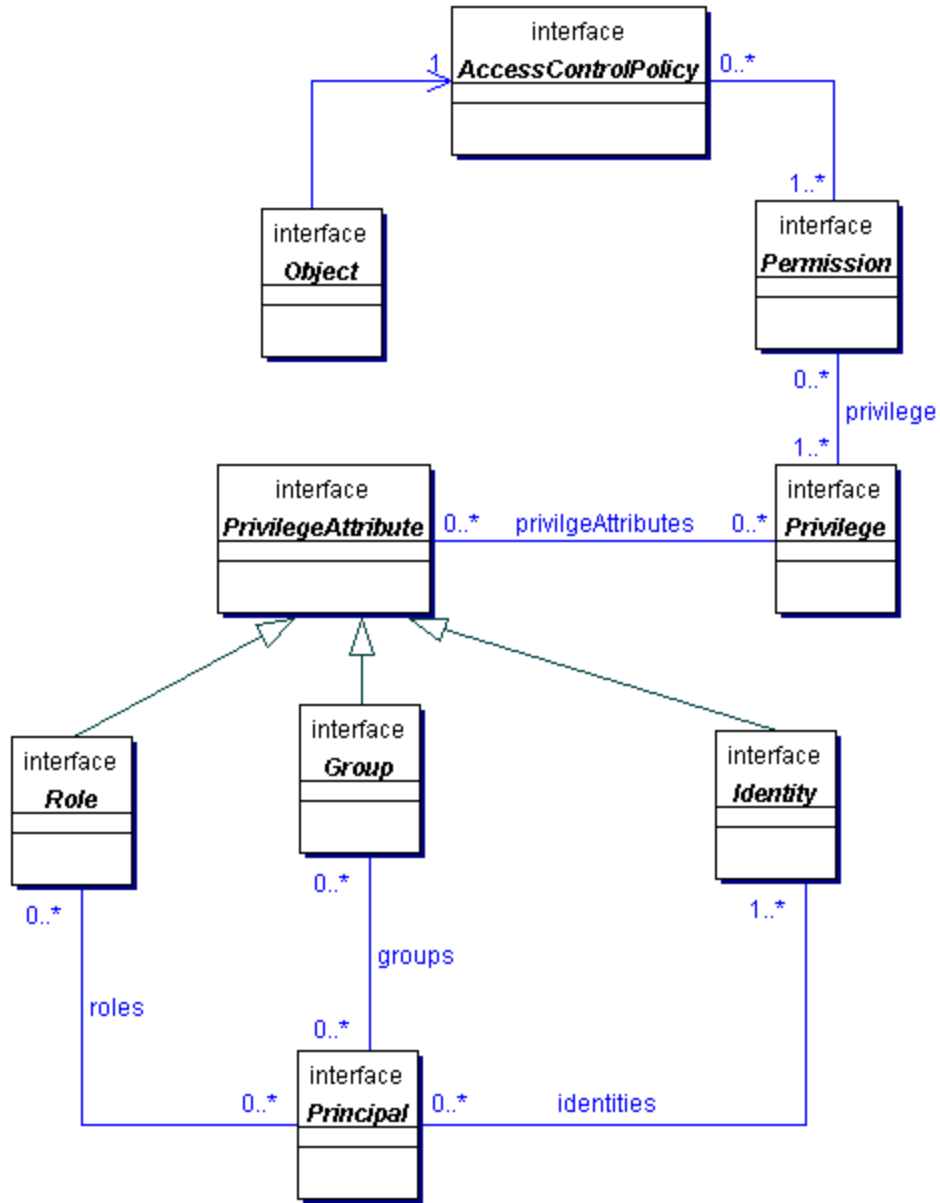


Figure 10: Information Model: Security View

1725
1726
1727

1728 **11.1 Class AccessControlPolicy**

1729 Every RegistryObject may be associated with exactly one AccessControlPolicy,
1730 which defines the policy rules that govern access to operations or methods
1731 performed on that RegistryObject. Such policy rules are defined as a collection of
1732 Permissions.

1733
1734
1735

1736

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

1737

1738 11.2 Class Permission

1739

1740 The Permission object is used for authorization and access control to
1741 RegistryObjects in the *Registry*. The Permissions for a RegistryObject are
1742 defined in an AccessControlPolicy object.

1743

1744 A Permission object authorizes access to a method in a RegistryObject if the
1745 requesting Principal has any of the Privileges defined in the Permission.

1746 **See Also:**

1747 [Privilege](#), [AccessControlPolicy](#)

1748

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> .

1749

1750 11.3 Class Privilege

1751

1752 A Privilege object contains zero or more PrivilegeAttributes. A PrivilegeAttribute
1753 can be a Group, a Role, or an Identity.

1754

1755 A requesting Principal MUST have all of the PrivilegeAttributes specified in a
1756 Privilege in order to gain access to a method in a protected RegistryObject.
1757 Permissions defined in the RegistryObject's AccessControlPolicy define the
1758 Privileges that can authorize access to specific methods.

1759

1760 This mechanism enables the flexibility to have object access control policies that
1761 are based on any combination of Roles, Identities or Groups.

1762 **See Also:**

1763 [PrivilegeAttribute](#), [Permission](#)

1764

1765

1766

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

1767

1768 11.4 Class PrivilegeAttribute

1769 **All Known Subclasses:**

1770 [Group](#), [Identity](#), [Role](#)

1771

1772 PrivilegeAttribute is a common base *Class* for all types of security attributes that
1773 are used to grant specific access control privileges to a Principal. A Principal may
1774 have several different types of PrivilegeAttributes. Specific combination of
1775 PrivilegeAttributes may be defined as a Privilege object.

1776 **See Also:**

1777 [Principal](#), [Privilege](#)

1778 11.5 Class Role

1779 **All Superclasses:**

1780 [PrivilegeAttribute](#)

1781

1782 11.5.1 A security Role PrivilegeAttribute

1783 For example a hospital may have *Roles* such as Nurse, Doctor, Administrator
1784 etc. Roles are used to grant Privileges to Principals. For example a Doctor *Role*
1785 may be allowed to write a prescription but a Nurse *Role* may not.

1786 11.6 Class Group

1787 **All Superclasses:**

1788 [PrivilegeAttribute](#)

1789

1790 11.6.1 A security Group PrivilegeAttribute

1791 A Group is an aggregation of users that may have different Roles. For example
1792 a hospital may have a Group defined for Nurses and Doctors that are
1793 participating in a specific clinical trial (e.g., AspirinTrial group). Groups are used
1794 to grant Privileges to Principals. For example the members of the AspirinTrial
1795 group may be allowed to write a prescription for Aspirin (even though Nurse Role
1796 as a rule may not be allowed to write prescriptions).

1797

1798

1799 **11.7 Class Identity**1800 **All Superclasses:**1801 [PrivilegeAttribute](#)

1802

1803 **11.7.1 A security Identity PrivilegeAttribute**

1804 This is typically used to identify a person, an organization, or software service.

1805 Identity attribute may be in the form of a digital certificate.

1806 **11.8 Class Principal**

1807

1808 Principal is a generic term used by the security community to include both people
1809 and software systems. The Principal object is an entity that has a set of1810 PrivilegeAttributes. These PrivilegeAttributes include at least one identity, and
1811 optionally a set of role memberships, group memberships or security clearances.1812 A principal is used to authenticate a requestor and to authorize the requested
1813 action based on the PrivilegeAttributes associated with the Principal.1814 **See Also:**1815 PrivilegeAttributes, [Privilege](#), [Permission](#)

1816

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> .

1817

1818

1818 **12 References**

- 1819 [ebGLOSS] ebXML Glossary,
1820 http://www.ebxml.org/documents/199909/terms_of_reference.htm
- 1821 [OAS] OASIS Information Model
1822 <http://xsun.sdct.itl.nist.gov/regrep/OasisRegrepSpec.pdf>
- 1823 [ISO] ISO 11179 Information Model
1824 <http://208.226.167.205/SC32/jtc1sc32.nsf/576871ad2f11bba785256621005419d7/b83fc7816a6064c68525690e0065f913?OpenDocument>
- 1826 [BRA97] IETF (Internet Engineering Task Force). RFC 2119: Key words for use
1827 in RFCs to Indicate Requirement Levels
1828 <http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc2119.html>
- 1829 [ebRS] ebXML Registry Services Specification
1830 [http://www.oasisopen.org/committees/regrep/documents/2.10/specs/ebRS](http://www.oasisopen.org/committees/regrep/documents/2.10/specs/ebRS.pdf)
1831 [.pdf](http://www.oasisopen.org/committees/regrep/documents/2.10/specs/ebRS.pdf)
- 1832 [ebCPP] ebXML Collaboration-Protocol Profile and Agreement Specification
1833 <http://www.ebxml.org/specrafts/>
- 1834
1835 [UUID] DCE 128 bit Universal Unique Identifier
1836 http://www.opengroup.org/onlinepubs/009629399/apdxa.htm#tagcjh_20
1837 <http://www.opengroup.org/publications/catalog/c706.htm><http://www.w3.org/TR/REC-xml>
- 1838
1839
1840 [XPath] XML Path Language (XPath) Version 1.0
1841 <http://www.w3.org/TR/xpath>
- 1842
1843 [NCName] Namespaces in XML 19990114
1844 <http://www.w3.org/TR/REC-xml-names/#NT-NCName>.

1845 **13 Disclaimer**

- 1846 The views and specification expressed in this document are those of the authors
1847 and are not necessarily those of their employers. The authors and their
1848 employers specifically disclaim responsibility for any problems arising from
1849 correct or incorrect implementation or use of this design.
1850

1850 **14 Contact Information**

1851

1852 Team Leader

1853 Name: Lisa Carnahan
1854 Company: NIST
1855 Street: 100 Bureau Drive STOP 8970
1856 City, State, Postal Code: Gaithersburg, MD 20899-8970
1857 Country: USA
1858 Phone: (301) 975-3362
1859 Email: lisa.carnahan@nist.gov

1860

1861 Editor

1862 Name: Sally Fuger
1863 Company: Automotive Industry Action Group
1864 Street: 26200 Lahser Road, Suite 200
1865 City, State, Postal Code: Southfield, MI 48034
1866 Country: USA
1867 Phone: (248) 358-9744
1868 Email: sfuger@aiag.org

1869

1870 Technical Editor

1871 Name: Farrukh S. Najmi
1872 Company: Sun Microsystems
1873 Street: 1 Network Dr., MS BUR02-302
1874 City, State, Postal Code: Burlington, MA, 01803-0902
1875 Country: USA
1876 Phone: (781) 442-0703
1877 Email: najmi@east.sun.com

1878

1879

1879 Copyright Statement

1880 OASIS takes no position regarding the validity or scope of any intellectual
1881 property or other rights that might be claimed to pertain to the implementation or
1882 use of the technology described in this document or the extent to which any
1883 license under such rights might or might not be available; neither does it
1884 represent that it has made any effort to identify any such rights. Information on
1885 OASIS's procedures with respect to rights in OASIS specifications can be found
1886 at the OASIS website. Copies of claims of rights made available for publication
1887 and any assurances of licenses to be made available, or the result of an attempt
1888 made to obtain a general license or permission for the use of such proprietary
1889 rights by implementors or users of this specification, can be obtained from the
1890 OASIS Executive Director.

1891
1892 OASIS invites any interested party to bring to its attention any copyrights, patents
1893 or patent applications, or other proprietary rights which may cover technology
1894 that may be required to implement this specification. Please address the
1895 information to the OASIS Executive Director.

1896
1897 Copyright ©The Organization for the Advancement of Structured Information
1898 Standards [OASIS] 2002. All Rights Reserved.

1899 This document and translations of it may be copied and furnished to others, and
1900 derivative works that comment on or otherwise explain it or assist in its
1901 implementation may be prepared, copied, published and distributed, in whole or
1902 in part, without restriction of any kind, provided that the above copyright notice
1903 and this paragraph are included on all such copies and derivative works.
1904 However, this document itself may not be modified in any way, such as by
1905 removing the copyright notice or references to OASIS, except as needed for the
1906 purpose of developing OASIS specifications, in which case the procedures for
1907 copyrights defined in the OASIS Intellectual Property Rights document must be
1908 followed, or as required to translate it into languages other than English.

1909 The limited permissions granted above are perpetual and will not be revoked by
1910 OASIS or its successors or assigns.

1911 This document and the information contained herein is provided on an "AS IS"
1912 basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED,
1913 INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE
1914 INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
1915 WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR
1916 PURPOSE."