



Creating A Single Global Electronic Market

1

OASIS/ebXML Registry Information Model v2.5

– Committee Approved Specification

OASIS/ebXML Registry Technical Committee

June 2003

1 *This page intentionally left blank.*

2 **1 Status of this Document**

3

4 This document is an OASIS ebXML Registry Technical Committee Approved
5 Specification - June 2003.

6 Distribution of this document is unlimited.

7 The document formatting is based on the Internet Society's Standard RFC format.

8 ***This version:***

9 <http://www.oasis-open.org/committees/regrep/documents/2.5/specs/ebrim-2.5.pdf>

10

11 ***Latest Technical Committee Approved version:***

12 <http://www.oasis-open.org/committees/regrep/documents/2.1/specs/ebRIM.pdf>

13

14 ***Latest OASIS Approved Standard:***

15 <http://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebRIM.pdf>

16 **2 OASIS/ebXML Registry Technical Committee**

17 This is an OASIS/ebXML Registry Technical Committee draft document. The following
18 persons are members of the OASIS/ebXML Registry Technical Committee:

19
20 Kathryn Breining, Boeing
21 Joseph M. Chiusano, Booz Allen Hamilton
22 Suresh Damodaran, Sterling Commerce
23 Sally Fuger, Individual Member
24 Peter Kacandes, Adobe Systems Incorporated
25 Michael Kass, NIST
26 Paul Macias, LMI
27 Matthew MacKenzie, Individual Member
28 Monica Martin, Sun Microsystems
29 Farrukh Najmi, Sun Microsystems
30 Sanjay Patil, IONA
31 Nikola Stojanovic, Individual Member
32 Uday Subarayan, Sun Microsystems

33 Contributors

34 The following persons contributed to the content of this document, but were not a voting
35 member of the OASIS/ebXML Registry Technical Committee.

36 John Bekisz, Software AG, Inc.
37 Lisa Carnahan, NIST
38 Anne Fischer, Individual
39 Len Gallagher, NIST
40 Duane Nickull, XML Global
41 John Silva, Philips Medical
42 Sekhar Vajjhala, Sun Microsystems

43

44

45

45	Table of Contents	
46		
47	1 Status of this Document.....	2
48	2 OASIS/ebXML Registry Technical Committee	3
49	3 Introduction.....	11
50	3.1 Summary of Contents of Document	11
51	3.2 General Conventions.....	11
52	3.2.1 Naming Conventions.....	11
53	3.3 Audience	12
54	3.4 Related Documents.....	12
55	4 Design Objectives	13
56	4.1 Goals	13
57	5 System Overview	14
58	5.1 Role of ebXML <i>Registry</i>	14
59	5.2 Registry Services	14
60	5.3 What the Registry Information Model Does	14
61	5.4 How the Registry Information Model Works	14
62	5.5 Where the Registry Information Model May Be Implemented	14
63	5.6 Conformance to an ebXML Registry.....	15
64	6 Registry Information Model: High Level Public View	16
65	6.1 RegistryObject.....	16
66	6.2 RepositoryItem	17
67	6.3 Slot.....	17
68	6.4 Association.....	17
69	6.5 ExternalIdentifier	17
70	6.6 ExternalLink	17
71	6.7 ClassificationScheme.....	18
72	6.8 ClassificationNode.....	18
73	6.9 Classification	18
74	6.10 RegistryPackage	18
75	6.11 AuditableEvent	18
76	6.12 User 18	
77	6.13 PostalAddress	18
78	6.14 EmailAddress	19
79	6.15 Organization	19
80	6.16 Service19	
81	6.17 ServiceBinding	19
82	6.18 SpecificationLink.....	19
83	7 Registry Information Model: Detail View.....	20
84	7.1 Attribute and Methods of Information Model Classes	21
85	7.2 Data Types.....	22
86	7.3 Object Reference Support	22

87	7.3.1	Class ObjectRef.....	22
88	7.4	Internationalization (I18N) Support	23
89	7.4.1	Class InternationalString	23
90	7.4.2	Class LocalizedString.....	24
91	7.5	Class RegistryObject.....	24
92	7.5.1	Attribute Summary.....	25
93	7.5.2	Attribute classifications	26
94	7.5.3	Attribute description.....	26
95	7.5.4	Attribute externalIdentifier	26
96	7.5.5	Attribute id.....	26
97	7.5.6	Attribute home	26
98	7.5.7	Attribute name	27
99	7.5.8	Attribute objectType	27
100	7.5.9	Attribute slots.....	27
101	7.5.10	Attribute status.....	27
102	7.5.11	Method Summary.....	28
103	7.6	Class RegistryEntry	28
104	7.6.1	Attribute Summary.....	29
105	7.6.2	Attribute expiration	29
106	7.6.3	Attribute majorVersion.....	29
107	7.6.4	Attribute minorVersion	29
108	7.6.5	Attribute stability	30
109	7.6.6	Attribute userVersion	30
110	7.7	Class Slot.....	30
111	7.7.1	Attribute Summary.....	30
112	7.7.2	Attribute name	31
113	7.7.3	Attribute slotType	31
114	7.7.4	Attribute values.....	31
115	7.8	Class ExtrinsicObject.....	31
116	7.8.1	Attribute Summary.....	31
117	7.8.2	Attribute isOpaque	32
118	7.8.3	Attribute mimeType	32
119	7.9	Class RegistryPackage	32
120	7.9.1	Attribute Summary.....	32
121	7.9.2	Method Summary.....	32
122	7.10	Class ExternalIdentifier.....	32
123	7.10.1	Attribute Summary.....	33
124	7.10.2	Attribute identificationScheme	33
125	7.10.3	Attribute registryObject.....	33
126	7.10.4	Attribute value	33
127	7.11	Class ExternalLink.....	33
128	7.11.1	Attribute Summary.....	33
129	7.11.2	Attribute externalURI.....	34
130	7.11.3	Method Summary.....	34
131	7.12	Class User.....	34

132	7.12.1 Attribute Summary.....	34
133	7.12.2 Attribute addresses.....	35
134	7.12.3 Attribute emailAddresses.....	35
135	7.12.4 Attribute personName.....	35
136	7.12.5 Attribute telephoneNumbers.....	35
137	7.12.6 Attribute <i>url</i>	35
138	7.12.7 Associating Users With Organizations.....	35
139	7.13 Class Organization.....	36
140	7.13.1 Attribute Summary.....	36
141	7.13.2 Attribute address.....	37
142	7.13.3 Attribute emailAddresses.....	37
143	7.13.4 Attribute parent.....	37
144	7.13.5 Attribute primaryContact.....	37
145	7.13.6 Attribute telephoneNumbers.....	37
146	7.13.7 Associating Organizations With RegistryObjects.....	37
147	7.14 Class PostalAddress.....	38
148	7.14.1 Attribute Summary.....	38
149	7.14.2 Attribute city.....	38
150	7.14.3 Attribute country.....	38
151	7.14.4 Attribute postalCode.....	39
152	7.14.5 Attribute state.....	39
153	7.14.6 Attribute street.....	39
154	7.14.7 Attribute streetNumber.....	39
155	7.14.8 Method Summary.....	39
156	7.15 Class TelephoneNumber.....	39
157	7.15.1 Attribute Summary.....	39
158	7.15.2 Attribute areaCode.....	40
159	7.15.3 Attribute countryCode.....	40
160	7.15.4 Attribute extension.....	40
161	7.15.5 Attribute number.....	40
162	7.15.6 Attribute phoneType.....	40
163	7.16 Class EmailAddress.....	40
164	7.16.1 Attribute Summary.....	40
165	7.16.2 Attribute address.....	40
166	7.16.3 Attribute type.....	41
167	7.17 Class PersonName.....	41
168	7.17.1 Attribute Summary.....	41
169	7.17.2 Attribute firstName.....	41
170	7.17.3 Attribute lastName.....	41
171	7.17.4 Attribute middleName.....	41
172	8 Association Information Model.....	42
173	8.1 Example of an Association.....	42
174	8.2 Source and Target Objects.....	42
175	8.3 Association Types.....	42
176	8.4 Intramural Association.....	43

177	8.5	Extramural Association.....	43
178	8.6	Confirmation of an Association.....	44
179	8.6.1	Confirmation of Intramural Associations.....	44
180	8.6.2	Confirmation of Extramural Associations.....	45
181	8.6.3	Deleting an Extramural Associations.....	45
182	8.7	Visibility of Unconfirmed Associations.....	45
183	8.8	Possible Confirmation States.....	45
184	8.9	Class Association.....	46
185	8.9.1	Attribute Summary.....	46
186	8.9.2	Attribute associationType.....	46
187	8.9.3	Attribute sourceObject.....	47
188	8.9.4	Attribute targetObject.....	47
189	8.9.5	Attribute isConfirmedBySourceOwner.....	47
190	8.9.6	Attribute isConfirmedByTargetOwner.....	47
191	9	Classification Information Model.....	48
192	9.1	Class ClassificationScheme.....	51
193	9.1.1	Attribute Summary.....	51
194	9.1.2	Attribute isInternal.....	51
195	9.1.3	Attribute nodeType.....	51
196	9.2	Class ClassificationNode.....	52
197	9.2.1	Attribute Summary.....	52
198	9.2.2	Attribute parent.....	52
199	9.2.3	Attribute code.....	52
200	9.2.4	Attribute path.....	53
201	9.2.5	Method Summary.....	53
202	9.2.6	Canonical Path Syntax.....	53
203	9.3	Class Classification.....	54
204	9.3.1	Attribute Summary.....	54
205	9.3.2	Attribute classificationScheme.....	55
206	9.3.3	Attribute classificationNode.....	55
207	9.3.4	Attribute classifiedObject.....	55
208	9.3.5	Attribute nodeRepresentation.....	55
209	9.3.6	Method Summary.....	55
210	9.3.7	Context Sensitive <i>Classification</i>	56
211	9.4	Example of Classification Schemes.....	58
212	10	Service Information Model.....	59
213	10.1	Class Service.....	59
214	10.1.1	Attribute Summary.....	59
215	10.1.2	Attribute serviceBindings.....	59
216	10.2	Class ServiceBinding.....	59
217	10.2.1	Attribute Summary.....	60
218	10.2.2	Attribute accessURI.....	60
219	10.2.3	Attribute specificationLinks.....	60
220	10.2.4	Attribute targetBinding.....	60
221	10.3	Class SpecificationLink.....	60

222	10.3.1	Attribute Summary.....	61
223	10.3.2	Attribute specificationObject.....	61
224	10.3.3	Attribute usageDescription.....	61
225	10.3.4	Attribute usageParameters.....	61
226	11	Event Information Model.....	62
227	11.1	Class AuditableEvent.....	62
228	11.1.1	Attribute Summary.....	62
229	11.1.2	Attribute eventType.....	63
230	11.1.3	Attribute affectedObjects.....	63
231	11.1.4	Attribute requestId.....	63
232	11.1.5	Attribute timestamp.....	63
233	11.1.6	Attribute user.....	64
234	11.2	Class Subscription.....	64
235	11.2.1	Attribute Summary.....	64
236	11.2.2	Attribute action.....	64
237	11.2.3	Attribute endDate.....	64
238	11.2.4	Attribute notificationInterval.....	65
239	11.2.5	Attribute selector.....	65
240	11.2.6	Attribute startDate.....	65
241	11.3	Class AdhocQuery.....	65
242	11.3.1	Method Summary.....	65
243	11.4	Class Action.....	65
244	11.5	Class NotifyAction.....	66
245	11.5.1	Attribute Summary.....	66
246	11.5.2	Attribute endPoint.....	66
247	11.5.3	Attribute notificationOption.....	66
248	12	Cooperating Registries Information Model.....	68
249	12.1.1	Class Registry.....	68
250	12.1.2	Class Federation.....	69
251	12.1.3	Federation Configuration.....	70
252	13	Access Control Information Model.....	71
253	13.1	Terminology.....	71
254	13.2	Resources.....	72
255	13.2.1	Attribute owner.....	72
256	13.2.2	Attribute selector.....	72
257	13.3	Actions.....	72
258	13.3.1	Create Action.....	72
259	13.3.2	Read Action.....	72
260	13.3.3	Update Action.....	73
261	13.3.4	Delete Action.....	73
262	13.3.5	Approve Action.....	73
263	13.3.6	Reference Action.....	73
264	13.3.7	Deprecate Action.....	73
265	13.3.8	Undeprecate Action.....	73
266	13.4	Subjects.....	73

267	13.4.1	Attribute id.....	73
268	13.4.2	Attribute group.....	74
269	13.4.3	Attribute role.....	74
270	13.5	Use Cases for Access Control Policies.....	74
271	13.5.1	Default Access Control Policy.....	74
272	13.5.2	Restrict Read Access To Specified Subjects.....	74
273	13.5.3	Grant Update and/or Delete Access To Specified Subjects.....	74
274	13.6	Abstract Access Control Model.....	75
275	13.7	Access Control Policy for a RegistryObject.....	75
276	13.7.1	Access Control Policy for a RepositoryItem.....	76
277	13.7.2	Default Access Control Policy.....	76
278	13.7.3	Root Access Control Policy.....	76
279	13.8	Access Control Model: [XACML] Binding.....	77
280	13.8.1	Resource Binding.....	78
281	13.8.2	Action Binding.....	78
282	13.8.3	Subject Binding.....	78
283	13.8.4	Constraints on [XACML] Binding.....	79
284	13.8.5	Examples of [XACML] Policies.....	79
285	13.8.6	Resolving PolicyReferences.....	87
286	13.8.7	ebXML Registry as a [XACML] Policy Store.....	87
287	13.9	Access Control Model: Custom Binding.....	87
288	Appendix A	Canonical Classification Schemes.....	88
289	A.1	ObjectType ClassificationScheme.....	88
290	A.2	AssociationType ClassificationScheme.....	88
291	A.3	PhoneType ClassificationScheme.....	88
292	A.4	EmailType ClassificationScheme.....	88
293	A.5	ContentManagementService ClassificationScheme.....	88
294	A.6	ErrorHandlingModel ClassificationScheme.....	88
295	A.7	InvocationModel ClassificationScheme.....	88
296	A.8	SubjectRole ClassificationScheme.....	88
297	A.9	SubjectGroup ClassificationScheme.....	89
298	14	References.....	90
299	15	Disclaimer.....	91
300	16	Contact Information.....	92
301	17	Copyright Statement.....	93

302 Table of Figures

303	Figure 1: Information Model High Level Public View.....	16
304	Figure 2: Information Model <i>Inheritance</i> View.....	20
305	Figure 3: User Affiliation With Organization Instance Diagram.....	36
306	Figure 4: Organization to RegistryObject Association Instance Diagram.....	38
307	Figure 5: Example of RegistryObject Association.....	42
308	Figure 6: Example of Intramural Association.....	43

309	Figure 7: Example of Extramural Association.....	44
310	Figure 8: Example showing a <i>Classification</i> Tree	49
311	Figure 9: Information Model <i>Classification</i> View	50
312	Figure 10: Classification <i>Instance</i> Diagram.....	50
313	Figure 11: Context Sensitive <i>Classification</i>	57
314	Figure 12: Service Information Model	59
315	Figure 13: Event Information Model.....	62
316	Figure 14: Federation Information Model.....	70
317	Figure 15: Instance Diagram for Abstract Access Control Information Model.....	75
318	Figure 16: Access Control Information Model: [XACML] Binding	77
319	Table of Tables	
320	Table 1: Sample Classification Schemes	58
321	Table 2: Resource Binding to [XACML]	78
322	Table 3: Action Binding to [XACML]	78
323	Table 4: Subject Binding to [XACML]	79
324		

325 **3 Introduction**

326 **3.1 Summary of Contents of Document**

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

328

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

332 **3.2 General Conventions**

333 The following conventions are used throughout this document:

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

336 The term "*repository item*" is used to refer to an object (e.g., an XML document or a
337 DTD) that resides in a repository for storage and safekeeping. Each repository item is
338 described by a RegistryObject instance. The RegistryObject catalogs the RepositoryItem
339 with metadata.

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

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

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

347 **3.2.1 Naming Conventions**

348

349 In order to enforce a consistent capitalization and naming convention in this document,
350 "Upper Camel Case" (*UCC*) and "Lower Camel Case" (*LCC*) Capitalization styles are
351 used in the following conventions:

- 352 ○ Element name is in *UCC* convention
353 (example: <UpperCamelCaseElement/>)
- 354 ○ Attribute name is in *LCC* convention
355 (example: <UpperCamelCaseElement lowerCamelCaseAttribute="whatEver"/>)
- 356 ○ *Class*, Interface names use *UCC* convention
357 (examples: ClassificationNode, Versionable)
- 358 ○ Method name uses *LCC* convention
359 (example: getName(), setName()).

360

361 **3.3 Audience**

362 The target audience for this specification is the community of software developers who
363 are:

- 364 ○ Implementers of ebXML Registry Services
- 365 ○ Implementers of ebXML Registry Clients

366 **3.4 Related Documents**

367 The following specifications provide some background and related information to the
368 reader:

- 369 a) ebXML Registry Services Specification [ebRS] - defines the actual Registry
370 Services based on this information model

371

372 **4 Design Objectives**

373 **4.1 Goals**

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

- 375 ○ Communicate what information is in the Registry and how that information is
- 376 organized
- 377 ○ Align with relevant works within other ebXML working groups
- 378 ○ Be able to evolve to support future ebXML Registry requirements
- 379 ○ Be compatible with other ebXML specifications

380

381 **5 System Overview**

382 **5.1 Role of ebXML Registry**

383

384 The registry provides a stable store where information submitted by a Submitting
385 Organization is made persistent. Such information is used to facilitate ebXML-based
386 Business to Business (B2B) partnerships and transactions. Submitted content may be
387 XML schema and documents, process descriptions, ebXML Core Components, context
388 descriptions, UML models, information about parties and even software components.

389 **5.2 Registry Services**

390 A set of Registry Services that provide access to registry content to clients of the registry
391 is defined in the ebXML Registry Services Specification [ebRS]. This document does not
392 provide details on these services but may occasionally refer to them.

393 **5.3 What the Registry Information Model Does**

394 The Registry Information Model provides a blueprint or high-level schema for the
395 ebXML Registry. Its primary value is for implementers of ebXML Registries. It provides
396 these implementers with information on the type of metadata that is stored in the registry
397 as well as the relationships among metadata classes.

- 398 ○ The Registry information model:
- 399 ○ Defines what types of objects are stored in the Registry
- 400 ○ Defines how stored objects are organized in the Registry

401

402 **5.4 How the Registry Information Model Works**

403 Implementers of the ebXML Registry MAY use the information model to determine
404 which classes to include in their registry implementation and what attributes and methods
405 these classes may have. They MAY also use it to determine what sort of database schema
406 their registry implementation may need.

407 [Note]The information model is meant to be
408 illustrative and does not prescribe any
409 specific Implementation choices.

410

411 **5.5 Where the Registry Information Model May Be Implemented**

412 The Registry Information Model MAY be implemented within an ebXML Registry in the
413 form of a relational database schema, object database schema or some other physical
414 schema. It MAY also be implemented as interfaces and classes within a registry
415 implementation.

416 **5.6 Conformance to an ebXML Registry**

417 If an implementation claims conformance to this specification then it supports all
418 required information model classes and interfaces, their attributes and their semantic
419 definitions that are visible through the ebXML Registry Services.

420 6 Registry Information Model: High Level Public View

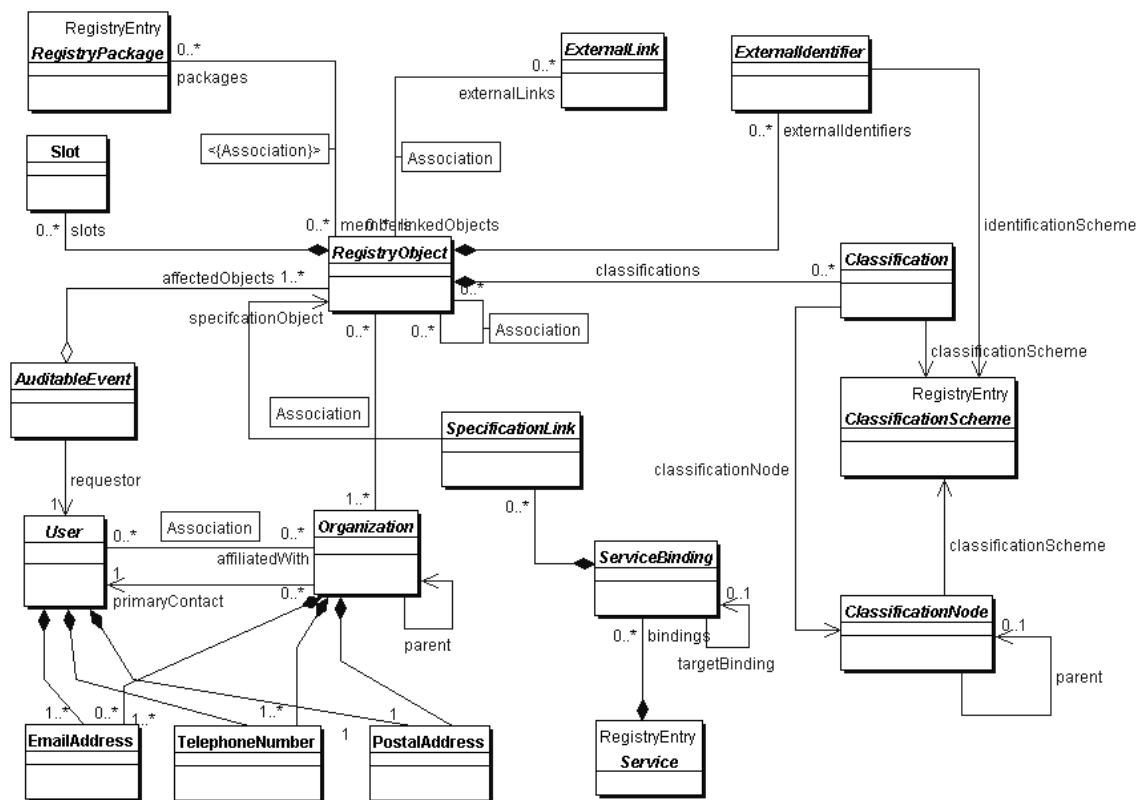
421 This section provides a high level public view of the objects in the *Registry*.

422

423 Figure 1 shows the high level public view of the objects in the *Registry* and their
 424 relationships as a *UML Class Diagram*. It does not show *Inheritance*, *Class* attributes or
 425 *Class* methods. The relationship links in the figure are either UML association or
 426 composition relationships (solid diamonds). In case of UML composition, instances of a
 427 class on the far side of the solid diamond are referred to as *composed objects* in the
 428 [ebRIM] and [ebRS] specifications.

429 The reader is again reminded that the information model is not modeling actual
 430 repository items.

431



432

433

Figure 1: Information Model High Level Public View

434 6.1 RegistryObject

435 The **RegistryObject** class is an abstract base class used by most classes in the model. It
 436 provides minimal metadata for registry objects. It also provides methods for accessing
 437 related objects that provide additional dynamic metadata for the registry object.

438 **6.2 RepositoryItem**

439 The RepositoryItem class represents an object (e.g., an XML document or a DTD) that
440 resides in a repository for storage and safekeeping. Each RepositoryItem instance is
441 associated with a RegistryObject instance. The RegistryObject catalogs the
442 RepositoryItem with metadata. The RepositoryItem class is not represented in the XML
443 schema for the registry because a repository item is a mime attachment within registry
444 requests as explained in [ebRS].

445 **6.3 Slot**

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

451 **6.4 Association**

452 Association instances are RegistryObject instances that are used to define many-to-many
453 associations between objects in the information model. Associations are described in
454 detail in section 8.

455 **6.5 ExternalIdentifier**

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

459 **6.6 ExternalLink**

460 ExternalLink instances are RegistryObject instances that contain a named URI to content
461 external to the Registry. Unlike managed content, such external content may change or be
462 deleted at any time without the knowledge of the Registry. A RegistryObject instance
463 may be associated with any number of ExternalLinks.

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

470 **6.7 ClassificationScheme**

471 ClassificationScheme instances are RegistryEntry instances that describe a structured
472 way to classify or categorize RegistryObject instances. The structure of the classification
473 scheme may be defined internal or external to the registry, resulting in a distinction
474 between internal and external classification schemes. A very common example of a
475 classification scheme in science is the “Classification of living things” where living
476 things are categorized in a tree like structure. Another example is the Dewey Decimal
477 system used in libraries to categorize books and other publications. ClassificationScheme
478 is described in detail in section 9.

479 **6.8 ClassificationNode**

480 ClassificationNode instances are RegistryObject instances that are used to define tree
481 structures under a ClassificationScheme, where each node in the tree is a
482 ClassificationNode and the root is the ClassificationScheme. Classification trees
483 constructed with ClassificationNodes are used to define the structure of Classification
484 schemes or ontologies. ClassificationNode is described in detail in section 9.

485 **6.9 Classification**

486 Classification instances are RegistryObject instances that are used to classify other
487 RegistryObject instances. A Classification instance identifies a ClassificationScheme
488 instance and taxonomy value defined within the classification scheme. Classifications can
489 be internal or external depending on whether the referenced classification scheme is
490 internal or external. Classification is described in detail in section 9.

491 **6.10 RegistryPackage**

492 RegistryPackage instances are RegistryEntry instances that group logically related
493 RegistryObject instances together.

494 **6.11 AuditableEvent**

495 AuditableEvent instances are RegistryObject instances that are used to provide an audit
496 trail for RegistryObject instances. AuditableEvent is described in detail in section 11.1.

497 **6.12 User**

498 User instances are RegistryObject instances that are used to provide information about
499 registered users within the Registry. User objects are used in audit trail for
500 RegistryObject instances. User is described in detail in section 7.12.

501 **6.13 PostalAddress**

502 PostalAddress is a simple reusable Entity Class that defines attributes of a postal address.

503 6.14 EmailAddress

504 EmailAddress is a simple reusable Entity Class that defines attributes of an email address.

505 6.15 Organization

506 Organization instances are RegistryObject instances that provide information on
507 organizations such as a Submitting Organization. Each Organization instance may have a
508 reference to a parent Organization.

509 6.16 Service

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

512 6.17 ServiceBinding

513 ServiceBinding instances are RegistryObject instances that represent technical
514 information on a specific way to access a specific interface offered by a Service instance.
515 A Service has a collection of ServiceBindings.
516

517 6.18 SpecificationLink

518 A SpecificationLink provides the linkage between a ServiceBinding and one of its
519 technical specifications that describes how to use the service with that ServiceBinding.
520 For example, a ServiceBinding may have a SpecificationLink instance that describes how
521 to access the service using a technical specification in the form of a WSDL or a CORBA
522 IDL document.
523

545 **7.1 Attribute and Methods of Information Model Classes**

546 Information model classes are defined primarily in terms of the attributes they carry.

547 These attributes provide information on the state of the instances of these classes.

548 Implementations of a registry often map class attributes to attributes in an XML store or
549 columns in a relational store.

550

551 Information model classes may also have methods defined for them. These methods
552 provide additional behavior for the class they are defined within. Methods are currently
553 used in mapping to Filter Query and the SQL Query capabilities defined in [eBRS].

554

555 Since the model supports inheritance between classes, it is usually the case that a class in
556 the model inherits attributes and methods from its base classes, in addition to defining its
557 own specialized attributes and methods.

558

558 7.2 Data Types

559 The following table lists the various data types used by the attributes within information
560 model classes:

561

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	anyURI	A URI of the form urn:uuid:<uuid> where <uuid> Must be a DCE 128 Bit Universally unique Id.	64 characters
URI	anyURI	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	

562

563 7.3 Object Reference Support

564 The information model supports the ability for an attribute in an instance of an
565 information model class to reference a RegistryObject instance using an object reference.
566 An object reference is modeled in this specification with the ObjectRef class.

567 7.3.1 Class ObjectRef

568 An instance of the ObjectRef class is used to reference a RegistryObject. A
569 RegistryObject may be referenced via an ObjectRef instance regardless of its location or
570 that of the object referring to it.

571 **7.3.1.1 Attribute Summary**

572

Attribute	Data Type	Required	Default Value	Specified By	Mutable
id	UUID	Yes		Client	Yes
home	URI	No		Client	Yes

573

574 **7.3.1.2 Attribute id**

575 Every ObjectRef instance must have an id attribute. The id attribute must contain the
576 value of the id attribute of the RegistryObject being referenced.

577 **7.3.1.3 Attribute home**

578 Every ObjectRef instance may optionally have a home attribute specified. The home
579 attribute if present must contain the base URI to the home registry for the referenced
580 RegistryObject. The base URI to a registry is described by the REST interface as defined
581 in [ebRS].

582 **7.3.1.4 Local Vs. Remote ObjectRefs**

583 When the home attribute is specified, and matches the base URI of a remote registry, then
584 ObjectRef is referred to as a remote ObjectRef.

585 If the home attribute is null then its default value is the base URI to the current registry.
586 When the home attribute is null or matches the base URI of the current registry, then the
587 ObjectRef is referred to as a local ObjectRef.

588 **7.4 Internationalization (I18N) Support**

589 Some information model classes have String attributes that are I18N capable and may be
590 localized into multiple native languages. Examples include the name and description
591 attributes of the RegistryObject class in 7.5.

592

593 The information model defines the InternationalString and the LocalizedString interfaces
594 to support I18N capable attributes within the information model classes. These classes
595 are defined below.

596 **7.4.1 Class InternationalString**

597 This class is used as a replacement for the String type whenever a String attribute needs
598 to be I18N capable. An instance of the InternationalString class composes within it
599 Collection of LocalizedString instances, where each String is specific to a particular
600 locale.

601 **7.4.1.1 Attribute Summary**

602

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

603

604 **7.4.1.2 Attribute localizedStrings**

605 Each InternationalString instance may have localizedStrings attribute that is a Collection
606 of zero or more LocalizedString instances.

607 **7.4.2 Class LocalizedString**

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

612 **7.4.2.1 Attribute Summary**

613

Attribute	Data Type	Required	Default Value	Specified By	Mutable
lang	language	No	en-US	Client	Yes
charset	String	No	UTF-8	Client	Yes
value	string	Yes		CLient	Yes

614

615 **7.4.2.2 Attribute lang**

616 Each LocalizedString instance may have a lang attribute that specifies the language used
617 by that LocalizedString.

618 **7.4.2.3 Attribute charset**

619 Each LocalizedString instance may have a charset attribute that specifies the name of the
620 character set used by that LocalizedString.

621 **7.4.2.4 Attribute value**

622 Each LocalizedString instance must have a value attribute that specifies the string value
623 used by that LocalizedString.

624 **7.5 Class RegistryObject**625 **Direct Known Subclasses:**

626 [Association](#), [AuditableEvent](#), [Classification](#), [ClassificationNode](#), [ExternalIdentifier](#),
 627 [ExternalLink](#), [Organization](#), [RegistryEntry](#), [User](#), [Service](#), [ServiceBinding](#),
 628 [SpecificationLink](#)

629

630 RegistryObject provides a common base class for almost all objects in the information
 631 model. Information model Classes whose instances have a unique identity are
 632 descendants of the RegistryObject Class.

633

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

638 7.5.1 Attribute Summary

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

641

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

642

Attribute	Data Type	Required	Default Value	Specified By	Mutable
classifications	Collection of Classification	No		Client	Yes
description	International-String	No		Client	Yes
externalIdentifiers	Collection of ExternalLink	No		Client	Yes
id	UUID	Yes		Client or registry	No
home	URI	No		Client	Yes
name	International-String	No		Client	Yes

objectType	ObjectRef	Yes		Registry	No
slots	Collection of Slot	No		Client	Yes
status	String16	Yes		Registry	Yes

643 Fix sorting on all tables??

644

645 7.5.2 Attribute classifications

646 Each RegistryObject instance may have a Collection of zero or more Classification
647 instances that are composed within the RegistryObject. These Classification instances
648 classify the RegistryObject.

649 7.5.3 Attribute description

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

652 7.5.4 Attribute externalIdentifier

653 Each RegistryObject instance may have a Collection of zero or more ExternalIdentifier
654 instances that are composed within the RegistryObject. These ExternalIdentifier instances
655 serve as alternate identifiers for the RegistryObject.

656 7.5.5 Attribute id

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

659

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

663

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

668 7.5.6 Attribute home

669 Each RegistryObject instance may have a home attribute. The home attribute if present,
670 must contain the base URI to the home registry for the RegistryObject instance. The base
671 URI to a registry is described by the REST interface as defined in [ebRS].

672 7.5.6.1 Local Replicas Vs. Remote Objects

673 When the home attribute is specified, and matches the base URI of a remote registry, then
674 RegistryObject is referred to as a local replica of a remote RegistryObject.

675 If the home attribute is null then its default value is the base URI to the current registry.
 676 When the home attribute is null or matches the base URI of the current registry, then the
 677 RegistryObject is referred to as a local RegistryObject.

678 **7.5.7 Attribute name**

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

682 **7.5.8 Attribute objectType**

683 Each RegistryObject instance has an objectType attribute. The value of the objectType
 684 attribute MUST be a reference to a ClassificationNode in the canonical ObjectType
 685 ClassificationScheme as referenced in appendix A.1. A Registry MUST support the
 686 object types as defined by the ObjectType ClassificationScheme referenced in appendix
 687 A.1. The canonical ObjectType ClassificationScheme may easily be extended by adding
 688 additional ClassificationNodes to the canonical ObjectType ClassificationScheme.

689

690 The objectType for almost all objects in the information model matches the
 691 ClassificationNode that corresponds to the name of their class. For example the
 692 objectType for a Classification is a reference to the ClassificationNode with code
 693 "Classification" in the canonical ObjectType ClassificationScheme. The only exception
 694 to this rule is that the objectType for an ExtrinsicObject or an ExternalLink instance may
 695 be defined by the submitter and indicates the type of content associated with that object.

696 **7.5.9 Attribute slots**

697 Each RegistryObject instance may have a Collection of zero or more Slot instances that
 698 are composed within the RegistryObject. These Slot instances serve as dynamically
 699 defined attributes for the RegistryObject.

700 **7.5.10 Attribute status**

701 Each RegistryObject instance must have a life cycle status indicator. The status is
 702 assigned by the registry.

703 **7.5.10.1 Pre-defined RegistryObject Status Types**

704 The following table lists pre-defined choices for RegistryObject status attribute.

705

Name	Description
Submitted	Status of a RegistryObject that catalogues content that has been submitted to the registry.
Approved	Status of a RegistryObject that catalogues content that has been

	submitted to the registry and has been subsequently approved.
Deprecated	Status of a RegistryObject that catalogues content that has been submitted to the registry and has been subsequently deprecated.
Undeprecated	Status of a RegistryObject that catalogues content that has been submitted to the registry and has been deprecated and then subsequently un-deprecated.
Withdrawn	Status of a RegistryObject that catalogues content that has been withdrawn from the registry. A repository item has been removed but its ExtrinsicObject still exists.

706

707 **7.5.11 Method Summary**

708 In addition to its attributes, the RegistryObject class also defines the following methods.

709 These methods are used to navigate relationship links from a RegistryObject instance to

710 other objects.

711

Method Summary for RegistryObject	
Collection	<u>getAuditTrail()</u> Gets the complete audit trail of all requests that effected a state change in this object as an ordered Collection of AuditableEvent objects.
Collection	<u>getExternalLinks()</u> Gets the ExternalLinks associated with this object.
Collection	<u>getRegistryPackages()</u> Gets the RegistryPackages that this object is a member of.

712

713

714 **7.6 Class RegistryEntry**715 **Super Classes:**716 RegistryObject

717

718 **Direct Known Subclasses:**719 ClassificationScheme, ExtrinsicObject, RegistryPackage, Service

720

721 RegistryEntry is a common base class for classes in the information model that require
 722 additional metadata beyond the minimal metadata provided by RegistryObject class.
 723 RegistryEntry is used as a base class for high-level coarse-grained objects in the registry.
 724 Their life cycle typically requires more management (e.g. may require approval,
 725 deprecation). They typically have relatively fewer instances but serve as a root of a
 726 composition hierarchy consisting of numerous objects that are sub-classes of
 727 RegistryObject but not RegistryEntry.
 728
 729 The additional metadata is described by the attributes of the RegistryEntry class below.

730 7.6.1 Attribute Summary

731

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
userVersion	ShortName	No		Client	Yes

732

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

735 7.6.2 Attribute expiration

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

741 7.6.3 Attribute majorVersion

742 Each RegistryEntry instance must have a major revision number for the current version
 743 of the RegistryEntry instance. This number is assigned by the registry when the object is
 744 created and it may be changed by the registry when an object is updated.

745 7.6.4 Attribute minorVersion

746 Each RegistryEntry instance must have a minor revision number for the current version
 747 of the RegistryEntry instance. This number is assigned by the registry when the object is
 748 created and it may be changed by the registry when an object is updated.

749 7.6.5 Attribute stability

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

752 7.6.5.1 Canonical RegistryEntry Stability Enumerations

753 The following table lists pre-defined choices for RegistryEntry stability attribute.

754 These pre-defined stability types are defined as a ClassificationScheme. While the
755 scheme may easily be extended, a registry MAY support the stability types listed below.

756 Following fonts need to be fixed so consistent with RS??

757

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.

758

759 7.6.6 Attribute userVersion

760 Each RegistryEntry instance may have a userVersion. The userVersion is similar to the
761 majorVersion-minorVersion tuple. They both provide an indication of the version of the
762 object. The majorVersion-minorVersion tuple is provided by the registry while
763 userVersion provides a user specified version for the object.

764 7.7 Class Slot

765 Slot instances provide a dynamic way to add arbitrary attributes to RegistryObject
766 instances. This ability to add attributes dynamically to RegistryObject instances enables
767 extensibility within the information model.

768

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

771 7.7.1 Attribute Summary

772

Attribute	Data Type	Required	Default Value	Specified By	Mutable
name	LongName	Yes		Client	No

slotType	LongName	No		Client	No
values	Collection of LongName	Yes		Client	No

773

774 **7.7.2 Attribute name**

775 Each Slot instance must have a name. The name is the primary means for identifying a
776 Slot instance within a RegistryObject. Consequently, the name of a Slot instance must be
777 locally unique within the RegistryObject instance.

778 **7.7.3 Attribute slotType**

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

780 **7.7.4 Attribute values**

781 A Slot instance must have a Collection of values. The collection of values may be empty.
782 Since a Slot represent an extensible attribute whose value may be a collection, therefore a
783 Slot is allowed to have a collection of values rather than a single value.

784 **7.8 Class ExtrinsicObject**785 **Super Classes:**

786 RegistryEntry, RegistryObject

787

788

789 ExtrinsicObjects provide metadata that describes submitted content whose type is not
790 intrinsically known to the registry and therefore MUST be described by means of
791 additional attributes (e.g., mime type).

792

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

795 **7.8.1 Attribute Summary**

796

Attribute	Data Type	Required	Default Value	Specified By	Mutable
isOpaque	Boolean	No	false	Client	No
mimeType	LongName	No	application/ octet-stream	Client	No

797

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

800 7.8.2 Attribute isOpaque

801 Each ExtrinsicObject instance may have an isOpaque attribute defined. This attribute
 802 determines whether the content catalogued by this ExtrinsicObject is opaque to (not
 803 readable by) the registry. In some situations, a Submitting Organization may submit
 804 content that is encrypted and not even readable by the registry.

805 7.8.3 Attribute mimeType

806 Each ExtrinsicObject instance may have a mimeType attribute defined. The mimeType
 807 provides information on the type of repository item catalogued by the ExtrinsicObject
 808 instance.

809

810 7.9 Class RegistryPackage

811 Super Classes:

812 RegistryEntry, RegistryObject

813

814 RegistryPackage instances allow for grouping of logically related RegistryObject
 815 instances even if individual member objects belong to different Submitting
 816 Organizations.

817 7.9.1 Attribute Summary

818

819 The RegistryPackage class defines no new attributes other than those that are inherited
 820 from RegistryEntry and RegistryObject base classes. The inherited attributes are not
 821 shown here.

822 7.9.2 Method Summary

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

824

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

825

826 7.10 Class ExternalIdentifier

827 Super Classes:

828 RegistryObject

829

830 ExternalIdentifier instances provide the additional identifier information to
 831 RegistryObject such as DUNS number, Social Security Number, or an alias name of the
 832 organization. The attribute *identificationScheme* is used to reference the identification
 833 scheme (e.g., “DUNS”, “Social Security #”), and the attribute *value* contains the actual
 834 information (e.g., the DUNS number, the social security number). Each RegistryObject
 835 may contain 0 or more ExternalIdentifier instances.

836 7.10.1 Attribute Summary

837

Attribute	Data Type	Required	Default Value	Specified By	Mutable
identificationScheme	ObjectRef	Yes		Client	Yes
registryObject	ObjectRef	Yes		Client	No
value	LongName	Yes		Client	Yes

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

839 7.10.2 Attribute identificationScheme

840 Each ExternalIdentifier instance must have an identificationScheme attribute that
 841 references a ClassificationScheme. This ClassificationScheme defines the namespace
 842 within which an identifier is defined using the value attribute for the RegistryObject
 843 referenced by the RegistryObject attribute.

844 7.10.3 Attribute registryObject

845 Each ExternalIdentifier instance must have a *registryObject* attribute that references the
 846 parent RegistryObject for which this is an ExternalIdentifier.

847 7.10.4 Attribute value

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

850 7.11 Class ExternalLink

851 Super Classes:

852 RegistryObject

853

854 ExternalLinks use URIs to associate content in the registry with content that may reside
 855 outside the registry. For example, an organization submitting a DTD could use an
 856 ExternalLink to associate the DTD with the organization's home page.

857 7.11.1 Attribute Summary

858

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

859

860 **7.11.2 Attribute externalURI**

861 Each ExternalLink instance must have an externalURI attribute defined. The externalURI
 862 attribute provides a URI to the external resource pointed to by this ExternalLink instance.
 863 If the URI is a URL then a registry must validate the URL to be resolvable at the time of
 864 submission before accepting an ExternalLink submission to the registry.

865 **7.11.3 Method Summary**

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

867

Method Summary of ExternalLink	
Collection	getLinkedObjects () Gets the collection of RegistryObjects that are linked by this ExternalLink to content outside the registry.

868

869

870 **7.12 Class User**871 **Super Classes:**872 RegistryObject

873

874 User instances represent users that have registered with a registry. User instances are also
 875 used in an AuditableEvent to keep track of the identity of the requestor that sent the
 876 request that generated the AuditableEvent.

877 **7.12.1 Attribute Summary**

878

Attribute	Data Type	Required	Default Value	Specified By	Mutable
addresses	Collection of PostalAddress	Yes		Client	Yes
emailAddresses	Collection of EmailAddress	Yes		Client	Yes
personName	PersonName	Yes		Client	No
telephoneNumbers	Collection of	Yes		Client	Yes

	TelephoneNumber				
url	URI	No		Client	Yes

879

880 **7.12.2 Attribute addresses**

881 Each User instance has an attribute addresses that is a Collection of PostalAddress
 882 instances. Each PostalAddress provides an postal address for that user. A User must have
 883 at least one postal address.

884 **7.12.3 Attribute emailAddresses**

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

888 **7.12.4 Attribute personName**

889 Each User instance must have a *personName* attribute that provides the name for that
 890 user.

891 **7.12.5 Attribute telephoneNumbers**

892 Each User instance must have a *telephoneNumbers* attribute that contains the Collection
 893 of TelephoneNumber instances defined for that user. A User must have at least one
 894 TelephoneNumber.

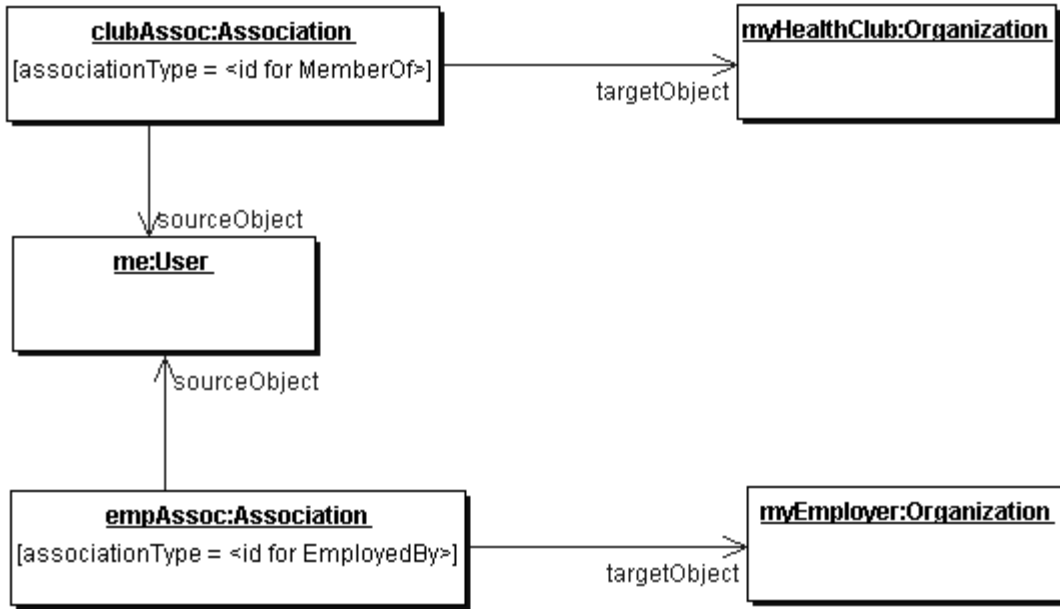
895 **7.12.6 Attribute url**

896 *Italicise all attributes in headings and para as done in url above??*

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

899 **7.12.7 Associating Users With Organizations**

900 A user may be affiliated with zero or more organizations. Each such affiliation is
 901 modeled in ebRIM using an Association instance between a User instance and an
 902 Organization instance. The associationType in such cases should be either the canonical
 903 “AffiliatedWith” associationType or a ClassificationNode that is a descendant of the
 904 ClassificationNode representing the canonical “AffiliatedWith” associationType.



905
906

Figure 3: User Affiliation With Organization Instance Diagram

907 **7.13 Class Organization**

908 **Super Classes:**

909 RegistryObject

910

911 Organization instances provide information on organizations such as a Submitting
912 Organization. Each Organization instance may have a reference to a parent Organization.

913 **7.13.1 Attribute Summary**

914

Attribute	Data Type	Required	Default Value	Specified By	Mutable
Address	PostalAddress	Yes		Client	Yes
emailAddresses	Collection of EmailAddress	No		Client	Yes
parent	ObjectRef	No		Client	Yes
primaryContact	ObjectRef	Yes		Client	No
telephoneNumbers	Collection of TelephoneNumber	Yes		Client	Yes

915

916 7.13.2 Attribute address

917 Each Organization instance must have an *address* attribute that provides the postal
918 address for that organization.

919 7.13.3 Attribute emailAddresses

920 Each Organization instance may have an attribute *emailAddresses* that is a Collection of
921 EmailAddress instances. Each EmailAddress provides an email address for that
922 Organization.

923 7.13.4 Attribute parent

924 Each Organization instance may have a *parent* attribute that references the parent
925 Organization instance, if any, for that organization.

926 7.13.5 Attribute primaryContact

927 Each Organization instance must have a *primaryContact* attribute that references the User
928 instance for the user that is the primary contact for that organization.

929 7.13.6 Attribute telephoneNumbers

930 Each Organization instance must have a *telephoneNumbers* attribute that contains the
931 Collection of TelephoneNumber instances defined for that organization. An Organization
932 must have at least one telephone number.

933 7.13.7 Associating Organizations With RegistryObjects

934 An organization may be associated with zero or more RegistryObject instances. Each
935 such association is modeled in ebRIM using an Association instance between an
936 Organization instance and a RegistryObject instance. The *associationType* in such cases
937 may be (but not restricted to) either the canonical “SubmitterOf” *associationType* or the
938 canonical “ResponsibleFor” *associationType*. The “SubmitterOf” *associationType*
939 indicates the organization that submitted the RegistryObject (via a User). The
940 “ResponsibleFor” *associationType* indicates the organization that is designated as the
941 organization responsible for the ongoing maintenance of the RegistryObject.

942 Association between Organizations and RegistryObjects do not entitle any special
943 privileges for the organizations with respect to the RegistryObject. Such privileges are
944 defined by the Access Control Policies defined for the RegistryObject as described in
945 chapter 13.

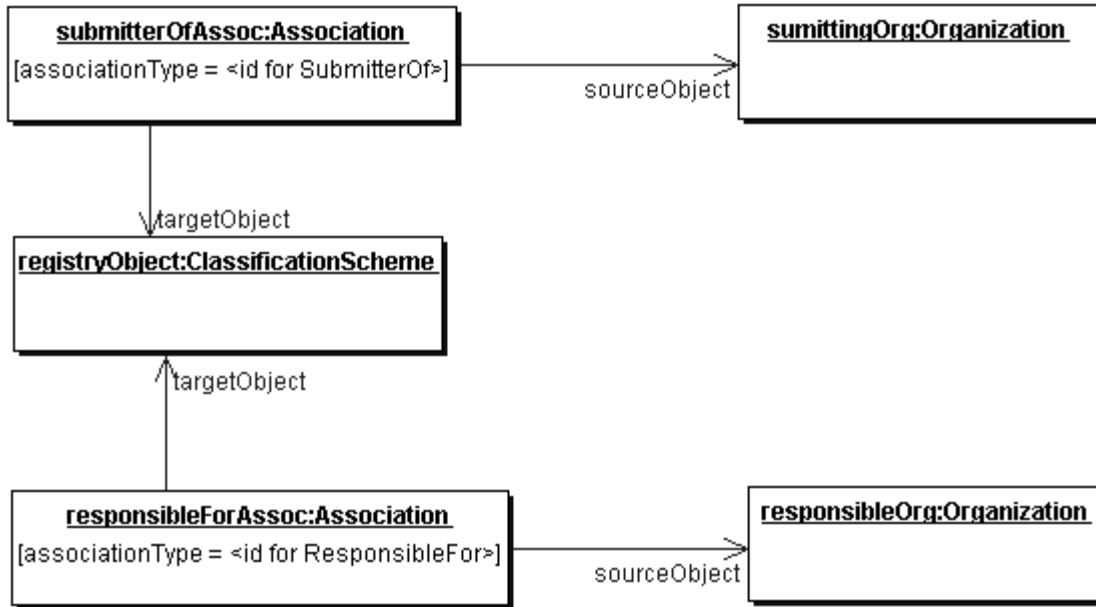


Figure 4: Organization to RegistryObject Association Instance Diagram

946
947

948

7.14 Class PostalAddress

949

950 PostalAddress is a simple reusable Entity Class that defines attributes of a postal address.

7.14.1 Attribute Summary

951

952

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

953

7.14.2 Attribute city

954

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

7.14.3 Attribute country

956

957 Each PostalAddress may have a *country* attribute identifying the country for that address.

958 7.14.4 Attribute postalCode

959 Each PostalAddress may have a *postalCode* attribute identifying the postal code (e.g., zip
960 code) for that address.

961 7.14.5 Attribute state

962 Each PostalAddress may have a *state* attribute identifying the state, province or region for
963 that address.

964 7.14.6 Attribute street

965 Each PostalAddress may have a *street* attribute identifying the street name for that
966 address.

967 7.14.7 Attribute streetNumber

968 Each PostalAddress may have a *streetNumber* attribute identifying the street number
969 (e.g., 65) for the street address.

970 7.14.8 Method Summary

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

972

Method Summary of ExternalLink	
Collection	<p>getSlots()</p> <p>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.</p>

973

974 7.15 Class TelephoneNumber

975 A simple reusable Entity Class that defines attributes of a telephone number.

976 7.15.1 Attribute Summary

977

Attribute	Data Type	Required	Default Value	Specified By	Mutable
areaCode	String8	No		Client	Yes
countryCode	String8	No		Client	Yes
extension	String8	No		Client	Yes
number	String16	No		Client	Yes

phoneType	ObjectRef	No		Client	Yes
url	URI	No		Client	Yes

978

979 **7.15.2 Attribute areaCode**

980 Each TelephoneNumber instance may have an *areaCode* attribute that provides the area
981 code for that telephone number.

982 **7.15.3 Attribute countryCode**

983 Each TelephoneNumber instance may have a *countryCode* attribute that provides the
984 country code for that telephone number.

985 **7.15.4 Attribute extension**

986 Each TelephoneNumber instance may have an *extension* attribute that provides the
987 extension number, if any, for that telephone number.

988 **7.15.5 Attribute number**

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

991 **7.15.6 Attribute phoneType**

992 Each TelephoneNumber instance may have *phoneType* attribute that provides the type for
993 the TelephoneNumber. The value of the phoneType attribute MUST be a reference to a
994 ClassificationNode in the canonical PhoneType ClassificationScheme as referenced in
995 appendix A.3.

996 **7.16 Class EmailAddress**

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

998 **7.16.1 Attribute Summary**

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

999 **7.16.2 Attribute address**

1000 Each EmailAddress instance must have an *address* attribute that provides the actual email
1001 address.

1002 **7.16.3 Attribute type**

1003 Each EmailAddress instance may have a *type* attribute that provides the type for that
 1004 email address. The value of the type attribute MUST be a reference to a
 1005 ClassificationNode in the canonical EmailType ClassificationScheme as referenced in
 1006 appendix A.4.

1007 **7.17 Class PersonName**

1008 A simple Entity Class for a person's name.

1009 **7.17.1 Attribute Summary**

1010

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

1011 **7.17.2 Attribute firstName**

1012 Each PersonName may have a *firstName* attribute that is the first name of the person.

1013 **7.17.3 Attribute lastName**

1014 Each PersonName may have a *lastName* attribute that is the last name of the person.

1015 **7.17.4 Attribute middleName**

1016 Each PersonName may have a *middleName* attribute that is the middle name of the
 1017 person.

1018 8 Association Information Model

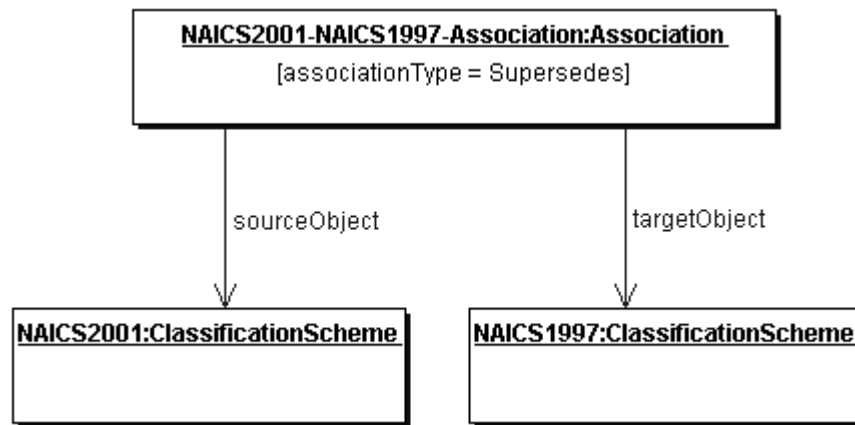
1019 A RegistryObject instance may be associated with zero or more RegistryObject instances.
 1020 The information model defines an Association class, an instance of which may be used to
 1021 associate any two RegistryObject instances.

1022 8.1 Example of an Association

1023 One example of such an association is between two ClassificationScheme instances,
 1024 where one ClassificationScheme supersedes the other ClassificationScheme as shown in
 1025 Figure 5. This may be the case when a new version of a ClassificationScheme is
 1026 submitted.

1027 In Figure 5, we see how an Association is defined between a new version of the NAICS
 1028 ClassificationScheme and an older version of the NAICS ClassificationScheme.

1029



1030

1031

Figure 5: Example of RegistryObject Association

1032 8.2 Source and Target Objects

1033 An Association instance represents an association between a source RegistryObject and a
 1034 target RegistryObject. These are referred to as *sourceObject* and *targetObject* for the
 1035 Association instance. It is important which object is the *sourceObject* and which is the
 1036 *targetObject* as it determines the directional semantics of an Association.

1037 In the example in Figure 5, it is important to make the newer version of NAICS
 1038 ClassificationScheme be the *sourceObject* and the older version of NAICS be the
 1039 *targetObject* because the *associationType* implies that the *sourceObject* supersedes the
 1040 *targetObject* (and not the other way around).

1041 8.3 Association Types

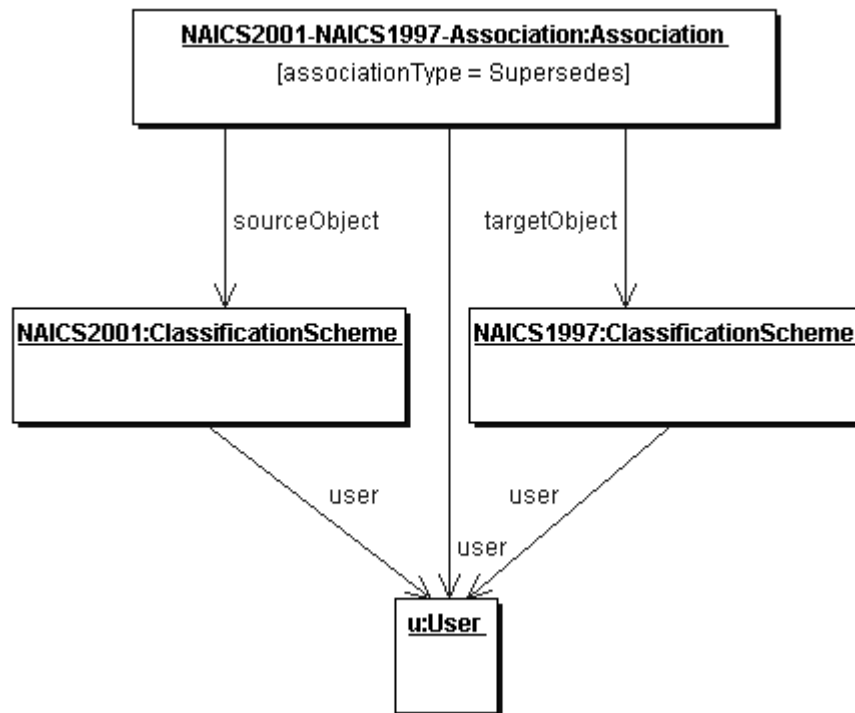
1042 Each Association must have an *associationType* attribute that identifies the type of that
 1043 association.

1044 8.4 Intramural Association

1045 A common use case for the Association class is when a User “u” creates an Association
 1046 “a” between two RegistryObjects “o1” and “o2” where Association “a” and
 1047 RegistryObjects “o1” and “o2” are objects that were created by the same User “u.” This
 1048 is the simplest use case, where the Association is between two objects that are owned by
 1049 the same User that is defining the Association. Such Associations are referred to as
 1050 intramural Associations.

1051 Figure 6 below, extends the previous example in Figure 5 for the intramural Association
 1052 case.

1053



1054

1055

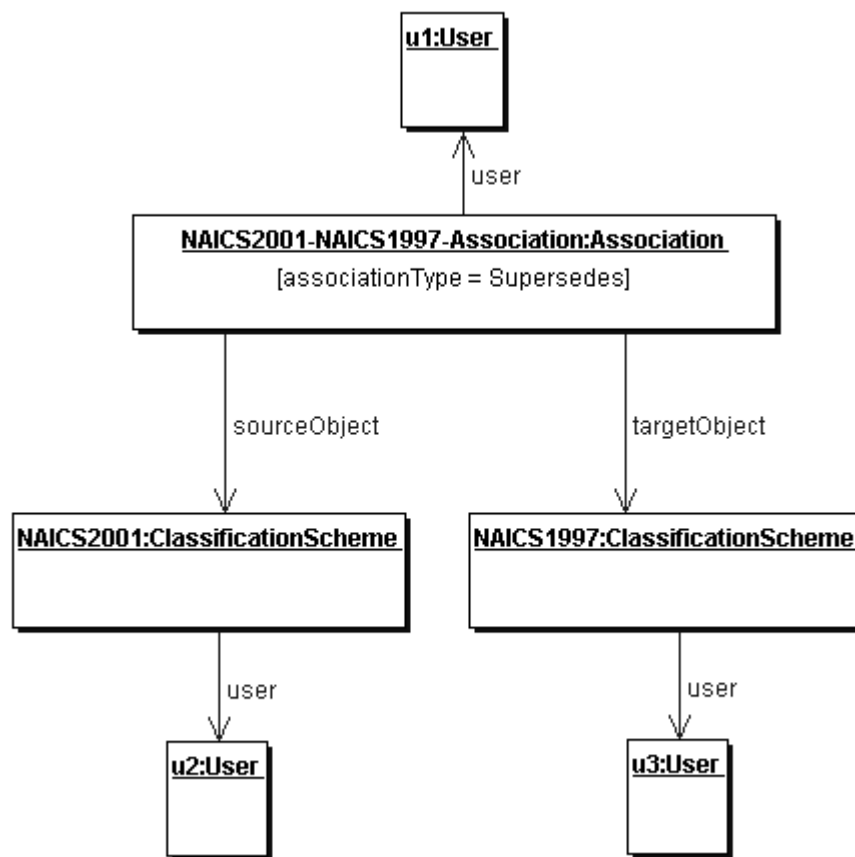
Figure 6: Example of Intramural Association

1056 8.5 Extramural Association

1057 The information model also allows more sophisticated use cases. For example, a User
 1058 “u1” creates an Association “a” between two RegistryObjects “o1” and “o2” where
 1059 association “a” is owned by User “u1”, but RegistryObjects “o1” and “o2” are owned by
 1060 User “u2” and User “u3” respectively.

1061 In this use case an Association is defined where either or both objects that are being
 1062 associated are owned by a User different from the User defining the Association. Such
 1063 Associations are referred to as extramural Associations. The Association class provides a
 1064 convenience method called *isExtramural* that returns “true” if the Association instance is
 1065 an extramural Association.

1066 Figure 7 below, extends the previous example in Figure 6 for the extramural Association
 1067 case. Note that it is possible for an extramural Association to have two distinct Users
 1068 rather than three distinct Users as shown in Figure 7. In such case, one of the two users
 1069 owns two of the three objects involved (Association, sourceObject and targetObject).
 1070



1071
 1072

Figure 7: Example of Extramural Association

1073 8.6 Confirmation of an Association

1074 An Association may need to be confirmed by the parties whose objects are involved in
 1075 that Association as the sourceObject or targetObject. This section describes the semantics
 1076 of confirmation of an association by the parties involved.

1077 8.6.1 Confirmation of Intramural Associations

1078 Intramural associations may be viewed as declarations of truth and do not require any
 1079 explicit steps to confirm that Association as being true. In other words, intramural
 1080 associations are implicitly considered confirmed.

1081 **8.6.2 Confirmation of Extramural Associations**

1082 An extramural association may be thought of as a unilateral assertion that may not be
1083 viewed as truth until it has been confirmed by the other (extramural) parties involved
1084 (Users “u2” and “u3” in the example in section 8.5).

1085 To confirm an extramural association, each of the extramural parties (parties that own the
1086 source or target object but do not own the Association) must submit an identical
1087 Association (clone Association) as the Association they are intending to confirm using a
1088 SubmitObjectsRequest. The clone Association must use the same id as the original
1089 Association.

1090 **8.6.3 Deleting an Extramural Associations**

1091 An Extramural Association is deleted like any other type of RegistryObject, using the
1092 RemoveObjectsRequest as defined in [ebRS]. However, in some cases deleting an
1093 extramural Association may not actually delete it but instead only revert a confirmed
1094 association to unconfirmed state.

1095

1096 When deleted by its owner, an extramural Association must always be deleted
1097 irrespective of its confirmation state.

1098 When deleted by the owner of its source/target object who is not the owner of the
1099 Association itself, an extramural Association must become unconfirmed.

1100 **8.7 Visibility of Unconfirmed Associations**

1101 Unconfirmed extramural Associations are visible to third party registry clients. Third
1102 party registry clients can determine the confirmation state of an extramural Association
1103 and decide whether to trust that Association or not.

1104 **8.8 Possible Confirmation States**

1105 Assume the most general case where there are three distinct User instances as shown in
1106 Figure 7 for an extramural Association. The extramural Association needs to be
1107 confirmed by both extramural parties (Users “u2” and “u3” in example) in order to be
1108 fully confirmed. The attributes *isConfirmedBySourceOwner* and
1109 *isConfirmedByTargetOwner* in the Association class provide access to the confirmation
1110 state for both the sourceObject and targetObject. A convenience method called
1111 *isConfirmed* provides a way to determine whether the Association is fully confirmed or
1112 not. So there are the following four possibilities related to the confirmation state of an
1113 extramural Association:

- 1114 ○ The Association is confirmed neither by the owner of the *sourceObject* nor by the
1115 owner of the *targetObject*.
- 1116 ○ The Association is confirmed by the owner of the *sourceObject* but it is not
1117 confirmed by the owner of the *targetObject*.
- 1118 ○ The Association is not confirmed by the owner of the *sourceObject* but it is
1119 confirmed by the owner of the *targetObject*.

- 1120 ○ The Association is confirmed by both the owner of the *sourceObject* and the
1121 owner of the *targetObject*. This is the only state where the Association is fully
1122 confirmed.

1123

1124 8.9 Class Association

1125 Super Classes:

1126 RegistryObject

1127

1128

1129 Association instances are used to define many-to-many associations among
1130 RegistryObjects in the information model.

1131

1132 An instance of the Association class represents an association between two
1133 RegistryObjects.

1134 8.9.1 Attribute Summary

1135

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

1136

1137 8.9.2 Attribute associationType

1138 Each Association must have an *associationType* attribute that identifies the type of that
1139 association. The value of the *associationType* attribute MUST be a reference to a
1140 ClassificationNode within the canonical AssociationType ClassificationScheme as
1141 described in appendix A.2.

1142 8.9.2.1 Canonical Association Types

1143 The following table lists canonical association types. These canonical association types
1144 are defined as a *ClassificationScheme* called AssociationType. While the ObjectType
1145 scheme may easily be extended, a Registry MUST support the association types as
1146 defined by the AssociationType scheme referenced in appendix A.2.

1147 **8.9.3 Attribute sourceObject**

1148 Each Association must have a *sourceObject* attribute that references the RegistryObject
1149 instance that is the source of that Association.

1150 **8.9.4 Attribute targetObject**

1151 Each Association must have a *targetObject* attribute that references the RegistryObject
1152 instance that is the target of that Association.

1153 **8.9.5 Attribute isConfirmedBySourceOwner**

1154 Each Association may have an *isConfirmedBySourceOwner* attribute that is set by the
1155 registry to be true if the Association has been confirmed by the owner of the
1156 sourceObject. For intramural Associations this attribute is always *true*. This attribute
1157 must be present when the object is retrieved from the registry. This attribute must be
1158 ignored if specified by the client when the object is submitted to the registry.

1159 **8.9.6 Attribute isConfirmedByTargetOwner**

1160 Each Association may have an *isConfirmedByTargetOwner* attribute that is set by the
1161 registry to be true if the association has been confirmed by the owner of the *targetObject*.
1162 For intramural Associations this attribute is always *true*. This attribute must be present
1163 when the object is retrieved from the registry. This attribute must be ignored if specified
1164 by the client when the object is submitted to the registry.
1165

Method Summary of Association	
Boolean	<p><u>isConfirmed()</u> Returns true if <i>isConfirmedBySourceOwner</i> and <i>isConfirmedByTargetOwner</i> attributes are both true. For intramural Associations always returns true.</p>
Boolean	<p><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.</p>

1166

1167 **9 Classification Information Model**

1168 This section describes how the information model supports Classification of
1169 RegistryObject.

1170

1171 A RegistryObject may be classified in many ways. For example the RegistryObject for
1172 the same Collaboration Protocol Profile (CPP) may be classified by its industry, by the
1173 products it sells and by its geographical location.

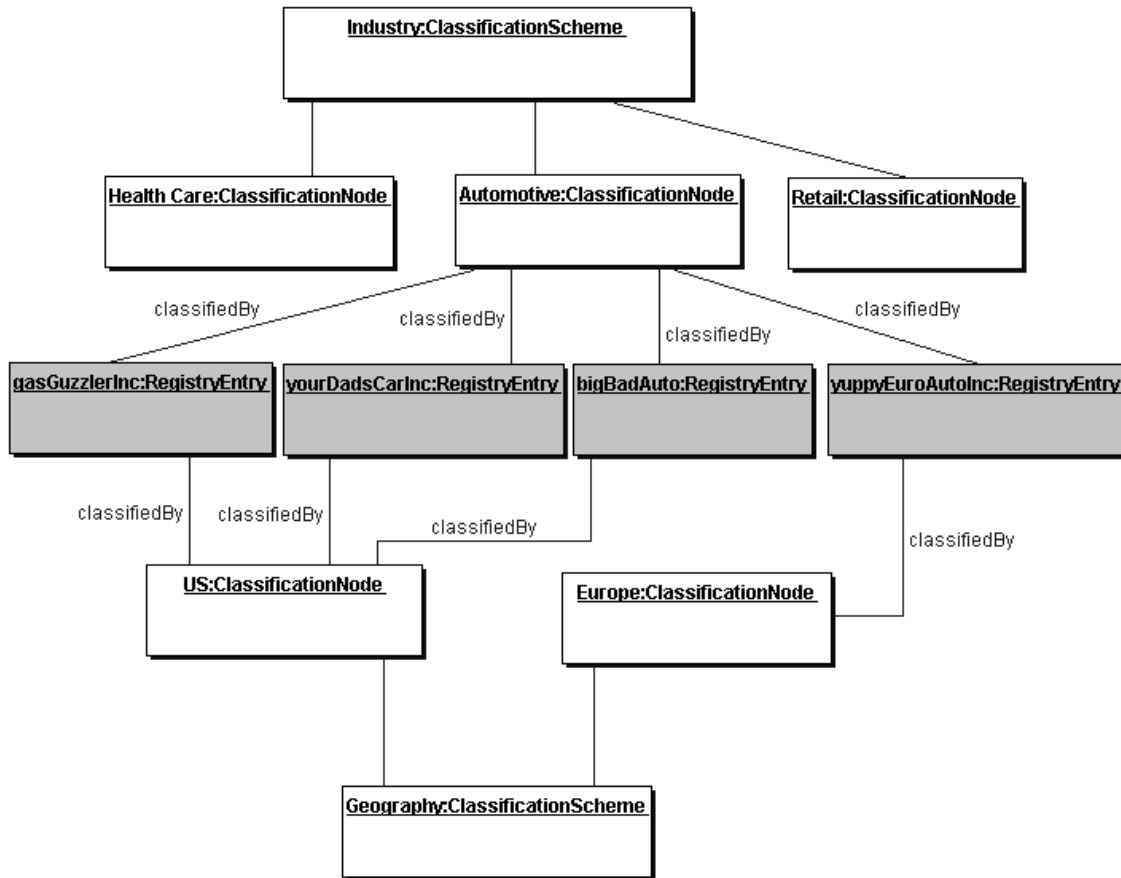
1174

1175 A general ClassificationScheme can be viewed as a Classification tree. In the example
1176 shown in Figure 8, RegistryObject instances representing Collaboration Protocol Profiles
1177 are shown as shaded boxes. Each Collaboration Protocol Profile represents an automobile
1178 manufacturer. Each Collaboration Protocol Profile is classified by the ClassificationNode
1179 named “Automotive” under the ClassificationScheme instance with name “Industry.”
1180 Furthermore, the US Automobile manufacturers are classified by the “US”
1181 ClassificationNode under the ClassificationScheme with name “Geography.” Similarly, a
1182 European automobile manufacturer is classified by the “Europe” ClassificationNode
1183 under the ClassificationScheme with name “Geography.”

1184

1185 The example shows how a RegistryObject may be classified by multiple
1186 ClassificationNode instances under multiple ClassificationScheme instances (e.g.,
1187 Industry, Geography).

1188

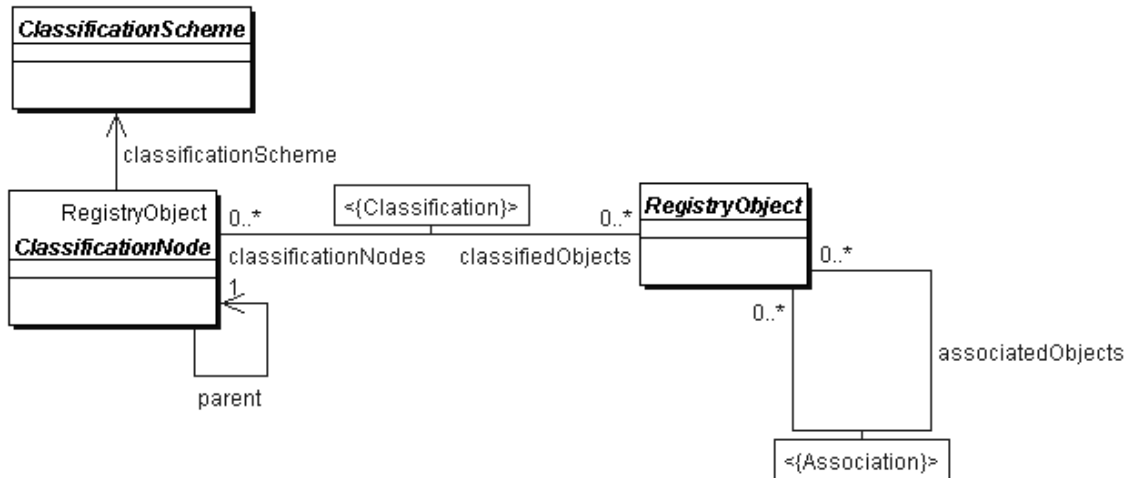


1189
1190

Figure 8: Example showing a *Classification Tree*

1191 It is important to point out that the shaded nodes (gasGuzzlerInc, yourDadsCarInc etc.)
1192 are not part of the Classification tree. The leaf nodes of the Classification tree are Health
1193 Care, Automotive, Retail, US and Europe. The shaded nodes are associated with the
1194 Classification tree via a Classification Instance that is not shown in the picture.
1195

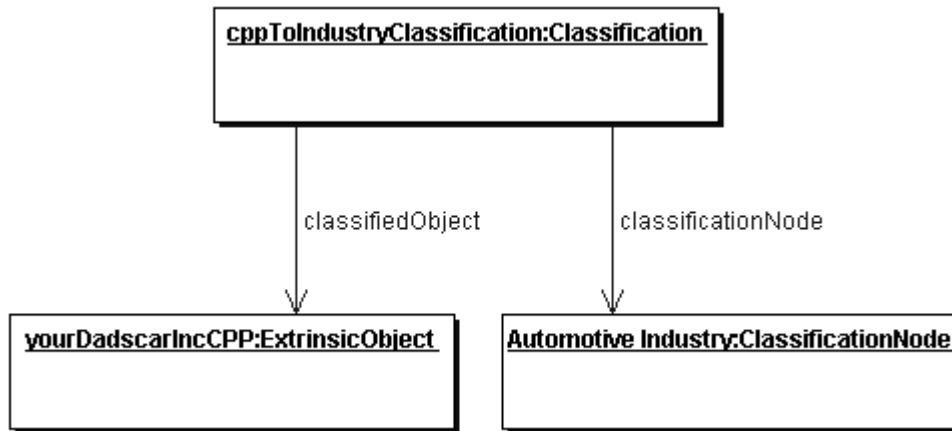
1196 In order to support a general ClassificationScheme that can support single level as well as
1197 multi-level Classifications, the information model defines the classes and relationships
1198 shown in Figure 9.



1199
 1200
 1201
 1202
 1203
 1204
 1205
 1206
 1207

Figure 9: Information Model Classification View

A Classification is somewhat like a specialized form of an Association. Figure 10 shows an example of an ExtrinsicObject Instance for a Collaboration Protocol Profile (CPP) object that is classified by a ClassificationNode representing the Industry that it belongs to.



1208
 1209
 1210
 1211
 1212
 1213
 1214
 1215

Figure 10: Classification Instance Diagram

1216 9.1 Class ClassificationScheme

1217 Base classes:

1218 RegistryEntry, RegistryObject

1219

1220 A ClassificationScheme instance describes a registered taxonomy. The taxonomy
1221 hierarchy may be defined internally to the registry by instances of ClassificationNode or
1222 it may be defined externally to the Registry, in which case the structure and values of the
1223 taxonomy elements are not known to the Registry.

1224 In the first case the classification scheme is defined to be *internal* and in the second case
1225 the classification scheme is defined to be *external*.

1226 The ClassificationScheme class inherits attributes and methods from the RegistryObject
1227 and RegistryEntry classes.

1228 9.1.1 Attribute Summary

1229

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

1230 Note that attributes inherited by ClassificationScheme class from the RegistryEntry class
1231 are not shown.

1232 9.1.2 Attribute isInternal

1233 When submitting a ClassificationScheme instance the Submitting Organization must
1234 declare whether the ClassificationScheme instance represents an internal or an external
1235 taxonomy. This allows the registry to validate the subsequent submissions of
1236 ClassificationNode and Classification instances in order to maintain the type of
1237 ClassificationScheme consistent throughout its lifecycle.

1238 9.1.3 Attribute nodeType

1239 When submitting a ClassificationScheme instance the Submitting Organization must
1240 declare the structure of taxonomy nodes within the ClassificationScheme. This attribute is
1241 an enumeration with the following values:

- 1242 ○ *UniqueCode*. This value indicates that each node of the taxonomy has a unique
1243 code assigned to it.
- 1244 ○ *EmbeddedPath*. This value indicates that the unique code assigned to each node
1245 of the taxonomy also encodes its path. This is the case in the NAICS taxonomy.

- 1246 ○ *NonUniqueCode*. In some cases nodes are not unique, and it is necessary to use
 1247 the full path (from ClassificationScheme to the node of interest) in order to
 1248 identify the node. For example, in a geography taxonomy Moscow could be under
 1249 both Russia and the USA, where there are five cities of that name in different
 1250 states.
 1251

1252 9.2 Class ClassificationNode

1253 Base classes:

1254 RegistryObject

1255
 1256 ClassificationNode instances are used to define tree structures where each node in the
 1257 tree is a ClassificationNode. Such Classification trees are constructed with
 1258 ClassificationNode instances under a ClassificationScheme instance, and are used to
 1259 define Classification schemes or ontologies.

1260 9.2.1 Attribute Summary

1261

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

1262

1263 9.2.2 Attribute parent

1264 Each ClassificationNode may have a *parent* attribute. The parent attribute either
 1265 references a parent ClassificationNode or a ClassificationScheme instance in case of first
 1266 level ClassificationNode instances.

1267 9.2.3 Attribute code

1268 Each ClassificationNode may have a *code* attribute. The code attribute contains a code
 1269 within a standard coding scheme. The code attribute of a ClassificationNode must be
 1270 unique with respect to all sibling ClassificationNodes that are immediate children of the
 1271 same parent ClassificationNode or ClassificationScheme.

1272 9.2.4 Attribute path

1273 Each ClassificationNode may have a *path* attribute. The path attribute must be present
 1274 when a ClassificationNode is retrieved from the registry. The path attribute must be
 1275 ignored when the path is specified by the client when the object is submitted to the
 1276 registry. The path attribute contains the canonical path from the ClassificationScheme of
 1277 this ClassificationNode. The path attribute of a ClassificationNode must be unique within
 1278 a registry. The path syntax is defined in 9.2.6.

1279 9.2.5 Method Summary

1280 In addition to its attributes, the ClassificationNode class also defines the following
 1281 methods.

1282

Method Summary of ClassificationNode	
ClassificationScheme	<u>getClassificationScheme</u> () Get the ClassificationScheme that this ClassificationNode belongs to.
Collection	<u>getClassifiedObjects</u> () Get the collection of RegistryObjects classified by this ClassificationNode.
Integer	<u>getLevelNumber</u> () 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.

1283

1284 In Figure 8, several instances of ClassificationNode are defined (all unshaded boxes). A
 1285 ClassificationNode has zero or one parent and zero or more ClassificationNodes for its
 1286 immediate children. The parent of a ClassificationNode may be another
 1287 ClassificationNode or a ClassificationScheme in case of first level ClassificationNodes.
 1288

1289 9.2.6 Canonical Path Syntax

1290 The path attribute of the ClassificationNode class contains an absolute path in a canonical
 1291 representation that uniquely identifies the path leading from the ClassificationScheme to
 1292 that ClassificationNode.

1293 The canonical path representation is defined by the following BNF grammar:

1294

```

1295 canonicalPath ::= '/' schemeId nodePath
1296 nodePath      ::= '/' nodeCode
1297               | '/' nodeCode ( nodePath )?
1298

```

1299 In the above grammar, schemeId is the id attribute of the ClassificationScheme instance,
 1300 and nodeCode is defined by NCName production as defined by <http://www.w3.org/TR/REC-xml-names/#NT-NCName>.
 1301

1302

1303 9.2.6.1 Example of Canonical Path Representation

1304 The following canonical path represents what the *path* attribute would contain for the
1305 ClassificationNode with code "United States" in the sample Geography scheme in section
1306 9.2.6.2.

1307
1308

```
/Geography-id/NorthAmerica/UnitedStates
```

1309 9.2.6.2 Sample Geography Scheme

1310 Note that in the following examples, the *id* attributes have been chosen for ease of
1311 readability and are therefore not valid URN or UUID values.

1312

```
1313 <ClassificationScheme id='Geography-id' name="Geography" />
```

1314

```
1315 <ClassificationNode id="NorthAmerica-id" parent="Geography-id"  
1316 code="NorthAmerica" />
```

```
1317 <ClassificationNode id="UnitedStates-id" parent="NorthAmerica-id"  
1318 code="UnitedStates" />
```

1319

```
1320 <ClassificationNode id="Asia-id" parent="Geography-id" code="Asia" />
```

```
1321 <ClassificationNode id="Japan-id" parent="Asia-id" code="Japan" />
```

```
1322 <ClassificationNode id="Tokyo-id" parent="Japan-id" code="Tokyo" />
```

1323

1324 9.3 Class Classification**1325 Base Classes:**

1326 RegistryObject

1327

1328 A Classification instance classifies a RegistryObject instance by referencing a node
1329 defined within a particular ClassificationScheme. An internal Classification will always
1330 reference the node directly, by its id, while an external Classification will reference the
1331 node indirectly by specifying a representation of its value that is unique within the
1332 external classification scheme.

1333

1334 The attributes and methods for the Classification class are intended to allow for
1335 representation of both internal and external classifications in order to minimize the need
1336 for a submission or a query to distinguish between internal and external classifications.

1337

1338 In Figure 8, Classification instances are not explicitly shown but are implied as
1339 associations between the RegistryObject instances (shaded leaf node) and the associated
1340 ClassificationNode.

1341 9.3.1 Attribute Summary

1342

Attribute	Data Type	Required	Default Value	Specified By	Mutable
classificationScheme	ObjectRef	for external classifications	null	Client	No
classificationNode	ObjectRef	for internal classifications	null	Client	No
classifiedObject	ObjectRef	Yes		Client	No
nodeRepresentation	LongName	for external classifications	null	Client	No

1343 Note that attributes inherited from the base classes of this class are not shown.

1344 9.3.2 Attribute classificationScheme

1345 If the Classification instance represents an external classification, then the
 1346 *classificationScheme* attribute is required. The *classificationScheme* value must reference
 1347 a ClassificationScheme instance.

1348 9.3.3 Attribute classificationNode

1349 If the Classification instance represents an internal classification, then the
 1350 *classificationNode* attribute is required. The *classificationNode* value must reference a
 1351 ClassificationNode instance.

1352 9.3.4 Attribute classifiedObject

1353 For both internal and external classifications, the *classifiedObject* attribute is required and
 1354 it references the RegistryObject instance that is classified by this Classification.

1355 9.3.5 Attribute nodeRepresentation

1356 If the Classification instance represents an external classification, then the
 1357 *nodeRepresentation* attribute is required. It is a representation of a taxonomy element
 1358 from a classification scheme. It is the responsibility of the registry to distinguish between
 1359 different types of *nodeRepresentation*, like between the classification scheme node code
 1360 and the classification scheme node canonical path. This allows the client to transparently
 1361 use different syntaxes for *nodeRepresentation*.

1362 9.3.6 Method Summary

1363 In addition to its attributes, the Classification class also defines the following methods:
 1364 Fix indentation in first para of method descriptions inside tables of methods??

Return Type	Method
String	<u>getPath()</u>

	<p>For an external classification returns a string that conforms to the canonical path syntax as specified in 9.2.6.</p> <p>For an internal classification, returns the value contained in the path attribute of the ClassificationNode instance identified by the classificationNode attribute.</p>
LongName	<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.</p> <p>For an internal classification, returns the value of the code attribute of the ClassificationNode instance identified by the classificationNode attribute.</p>

1365

1366 **9.3.7 Context Sensitive Classification**

1367 Consider the case depicted in Figure 11 where a Collaboration Protocol Profile for
 1368 ACME Inc. is classified by the “Japan” ClassificationNode under the “Geography”
 1369 Classification scheme. In the absence of the context for this Classification its meaning is
 1370 ambiguous. Does it mean that ACME is located in Japan, or does it mean that ACME
 1371 ships products to Japan, or does it have some other meaning? To address this ambiguity a
 1372 Classification may optionally be associated with another ClassificationNode (in this
 1373 example named isLocatedIn) that provides the missing context for the Classification.
 1374 Another Collaboration Protocol Profile for MyParcelService may be classified by the
 1375 “Japan” ClassificationNode where this Classification is associated with a different
 1376 ClassificationNode (e.g., named shipsTo) to indicate a different context than the one used
 1377 by ACME Inc.

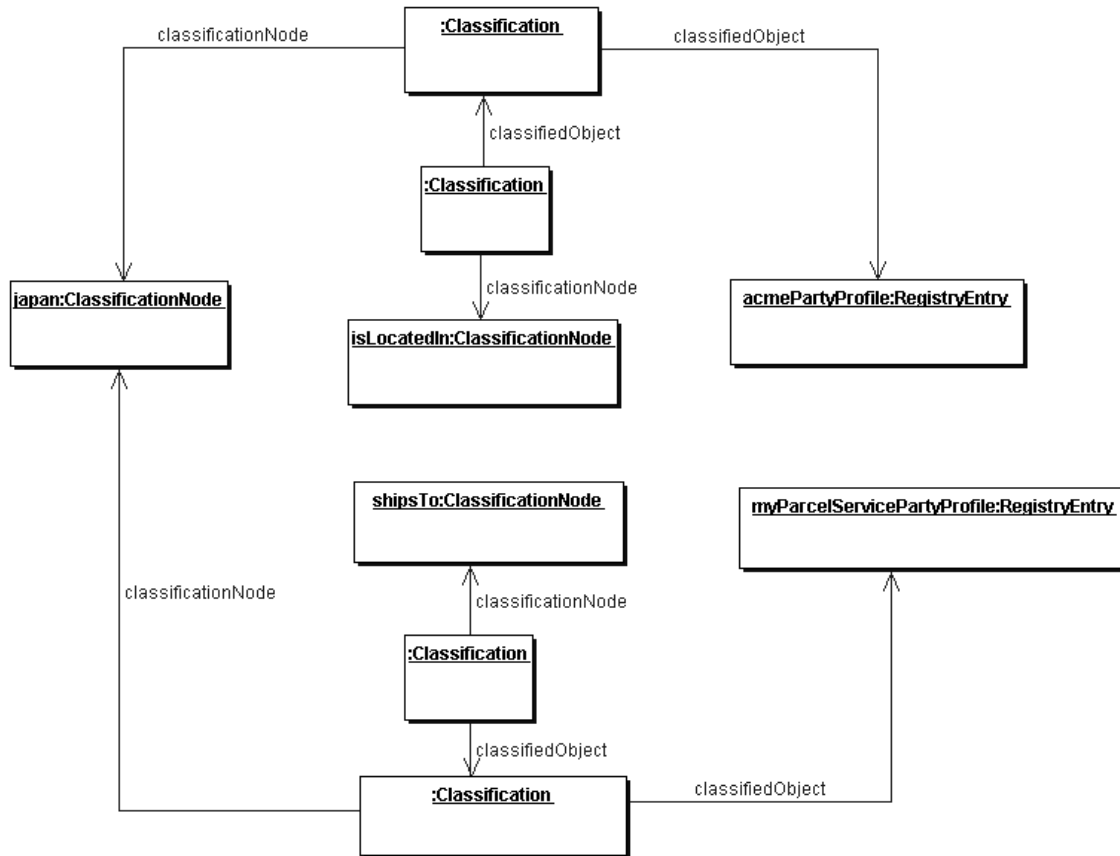


Figure 11: Context Sensitive Classification

1378
1379

1380 Thus, in order to support the possibility of Classification within multiple contexts, a
1381 Classification is itself classified by any number of Classifications that bind the first
1382 Classification to ClassificationNodes that provide the missing contexts.

1383

1384 In summary, the generalized support for *Classification* schemes in the information model
1385 allows:

- 1386 ○ A RegistryObject to be classified by defining an internal Classification that
- 1387 associates it with a ClassificationNode in a ClassificationScheme.
- 1388 ○ A RegistryObject to be classified by defining an external Classification that
- 1389 associates it with a value in an external ClassificationScheme.
- 1390 ○ A RegistryObject to be classified along multiple facets by having multiple
- 1391 Classifications that associate it with multiple ClassificationNodes or value within
- 1392 a ClassificationScheme.
- 1393 ○ A Classification defined for a RegistryObject to be qualified by the contexts in
- 1394 which it is being classified.

1395
1396

1397 9.4 Example of Classification Schemes

1398 The following table lists some examples of possible ClassificationSchemes enabled by
 1399 the information model. These schemes are based on a subset of contextual concepts
 1400 identified by the ebXML Business Process and Core Components Project Teams. This
 1401 list is meant to be illustrative not prescriptive.

1402

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 Business 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 Role of "Seller"	

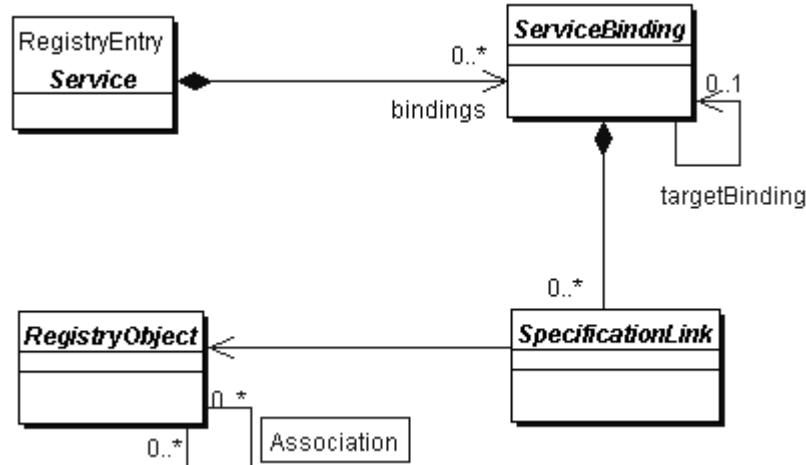
1403

Table 1: Sample Classification Schemes

1404

1405 **10 Service Information Model**

1406 This chapter describes the classes in the information model that support the registration
 1407 of services. The service registration information model is flexible and supports the
 1408 registration of web services as well as other types of services.



1409
 1410

Figure 12: Service Information Model

1411 **10.1 Class Service**

1412 **Super Classes:**

1413 RegistryEntry, RegistryObject

1414

1415 Service instances provide information on services, such as web services.

1416 **10.1.1 Attribute Summary**

1417

Attribute	Data Type	Required	Default Value	Specified By	Mutable
serviceBindings	Collection of ServiceBinding	Yes		Client	Yes

1418

1419 **10.1.2 Attribute serviceBindings**

1420 A Service must have a *serviceBindings* attribute that defines the service bindings that
 1421 provide access to that Service. Each ServiceBinding instance represents technical
 1422 information on a specific way to access a specific interface offered by a Service instance.

1423 **10.2 Class ServiceBinding**

1424 **Super Classes:**

1425 RegistryObject

1426

1427 ServiceBinding instances are RegistryObjects that represent technical information on a
1428 specific way to access a specific interface offered by a Service instance. A Service has a
1429 Collection of ServiceBindings.

1430 The *description* attribute of ServiceBinding provides details about the relationship
1431 between several specification links comprising the Service Binding.

1432 10.2.1 Attribute Summary

1433

Attribute	Data Type	Required	Default Value	Specified By	Mutable
accessURI	URI	No		Client	Yes
specificationLinks	Collection of SpecificationLink	Yes		Client	Yes
targetBinding	ObjectRef	No		Client	Yes

1434 10.2.2 Attribute accessURI

1435 A ServiceBinding may have an *accessURI* attribute that defines the URI to access that
1436 ServiceBinding. This attribute is ignored if a *targetBinding* attribute is specified for the
1437 ServiceBinding. If the URI is a URL then a registry must validate the URL to be
1438 resolvable at the time of submission before accepting a ServiceBinding submission to the
1439 registry.

1440 10.2.3 Attribute specificationLinks

1441 A ServiceBinding must have a *specificationLinks* attribute defined that is a collection of
1442 references to SpecificationLink instances. Each SpecificationLink instance links the
1443 ServiceBinding to a particular technical specification that may be used to access the
1444 Service for the ServiceBinding.

1445 10.2.4 Attribute targetBinding

1446 A ServiceBinding may have a *targetBinding* attribute defined that references another
1447 ServiceBinding. A *targetBinding* may be specified when a service is being redirected to
1448 another service. This allows the rehosting of a service by another service provider.

1449 10.3 Class SpecificationLink

1450 Super Classes:

1451 RegistryObject

1452

1453 A SpecificationLink provides the linkage between a ServiceBinding and one of its
 1454 technical specifications that describes how to use the service using the ServiceBinding.
 1455 For example, a ServiceBinding may have SpecificationLink instances that describe how
 1456 to access the service using a technical specification such as a WSDL document or a
 1457 CORBA IDL document.

1458 10.3.1 Attribute Summary

1459

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

1460

1461 10.3.2 Attribute specificationObject

1462 A SpecificationLink instance must have a *specificationObject* attribute that provides a
 1463 reference to a RegistryObject instance that provides a technical specification for the
 1464 parent ServiceBinding. Typically, this is an ExtrinsicObject instance representing the
 1465 technical specification (e.g., a WSDL document). It may also be an ExternalLink object
 1466 in case the technical specification is a resource that is external to the registry.

1467 10.3.3 Attribute usageDescription

1468 A SpecificationLink instance may have a *usageDescription* attribute that provides a
 1469 textual description of how to use the optional usageParameters attribute described next.
 1470 The usageDescription is of type InternationalString, thus allowing the description to be in
 1471 multiple languages.

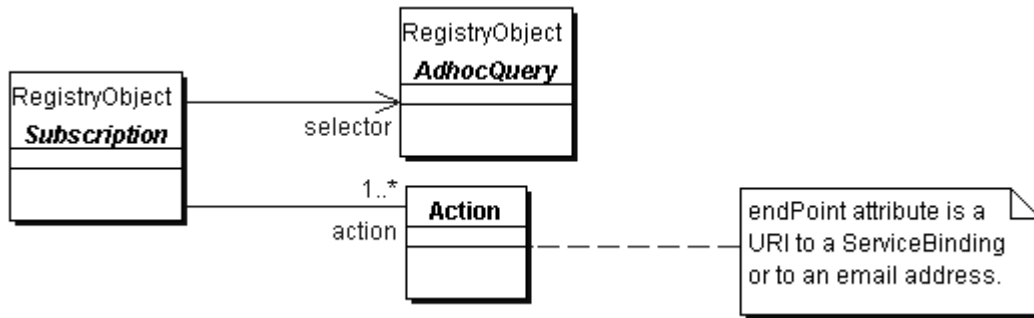
1472 10.3.4 Attribute usageParameters

1473 A SpecificationLink instance may have a *usageParameters* attribute that provides a
 1474 collection of Strings representing the instance specific parameters needed to use the
 1475 technical specification (e.g., a WSDL document) specified by this SpecificationLink
 1476 object.

1477

1478 **11 Event Information Model**

1479 This chapter defines the information model classes that support the registry Event
 1480 Notification feature. These classes include AuditableEvent, Subscription, Selector and
 1481 Action. They constitute the foundation of the Event Notification information model.
 1482 Figure 13 shows how a Subscription may be defined that uses a pre-configured
 1483 AdhocQuery instance as a selector to select the AuditableEvents of interest to the
 1484 subscriber and an Action to deliver the selected events to the subscriber. The Action may
 1485 deliver the events by using its endPoint attribute to invoke a registered ServiceBinding to
 1486 a registered Service or by sending the vents to an email address.



1487
 1488

Figure 13: Event Information Model

1489 **11.1 Class AuditableEvent**

1490 **Super Classes:**

1491 RegistryObject

1492
 1493 AuditableEvent instances provide a long-term record of events that effected a change in a
 1494 RegistryObject. A RegistryObject is associated with an ordered Collection of
 1495 AuditableEvent instances that provide a complete audit trail for that RegistryObject.
 1496 AuditableEvents are usually a result of a client-initiated request. AuditableEvent
 1497 instances are generated by the Registry Service to log such Events.
 1498 Often such events effect a change in the life cycle of a RegistryObject. For example a
 1499 client request could Create, Update, Deprecate or Delete a RegistryObject. An
 1500 AuditableEvent is created if and only if a request creates or alters the content or
 1501 ownership of a RegistryObject. Read-only requests do not generate an AuditableEvent.

1502 **11.1.1 Attribute Summary**

1503

Attribute	Data Type	Required	Default Value	Specified By	Mutable
eventType	LongName	Yes		Registry	No
affectedObjects	Collection of	Yes		Registry	No

	ObjectRef				
requestId	URI	Yes		Registry	No
timestamp	dateTime	Yes		Registry	No
user	ObjectRef	Yes		Registry	No

1504

1505 11.1.2 Attribute eventType

1506 Each AuditableEvent must have an eventType attribute which identifies the type of event
1507 recorded by the AuditableEvent.

1508 11.1.2.1 Pre-defined Auditable Event Types

1509 The following table lists pre-defined auditable event types. A *Registry* MUST support the
1510 event types listed below.

1511

Name	Description
Approved	An Event that marks the approval of a RegistryObject.
Created	An Event that marks the creation of a RegistryObject.
Deleted	An Event that marks the deletion of a RegistryObject.
Deprecated	An Event that marks the deprecation of a RegistryObject.
Downloaded	An Event that marks the downloading of a RegistryObject.
Relocated	An Event that marks the relocation of a RegistryObject.
Updated	An Event that that marks the updating of a RegistryObject.
Versioned	An Event that marks the versioning of a RegistryObject.

1512 11.1.3 Attribute affectedObjects

1513 Each AuditableEvent must have an *affectedObjects* attribute that identifies the collection
1514 of RegistryObjects instances that were affected by this event.

1515 11.1.4 Attribute requestId

1516 Each AuditableEvent must have a *requestId* attribute that identifies the client request
1517 instance that affected this event.

1518 11.1.5 Attribute timestamp

1519 Each AuditableEvent must have a *timestamp* attribute that records the date and time that
1520 this event occurred.

1521 **11.1.6 Attribute user**

1522 Each AuditableEvent must have a *user* attribute that identifies the User that sent the
1523 request that generated this event affecting the RegistryObject instance.

1524 **11.2 Class Subscription**1525 **Super Classes:**

1526 RegistryObject

1527

1528 Subscription instances are RegistryObjects that define a User's interest in certain types of
1529 AuditableEvents. A User may create a subscription with a registry if she wishes to
1530 receive notification for a specific type of event.

1531 **11.2.1 Attribute Summary**

1532

Attribute	Data Type	Required	Default Value	Specified By	Mutable
actions	Collection of Action	Yes, may be empty		Client	Yes
endDate	dateTime	No		Client	Yes
notificationInterval	duration	No	P1D (1 day)	Client	No
selector	AdhocQuery	Yes		Client	No
startDate	dateTime	No	Current time	Client	Yes

1533

1534 **11.2.2 Attribute action**

1535 A Subscription instance must have an *actions* attribute that is a Collection of zero or
1536 more Action instances. An Action instance describes what action the registry must take
1537 when an event matching the Subscription transpires. The Action class is described in
1538 section 11.4.

1539 **11.2.3 Attribute endDate**

1540 This attribute denotes the time after which the subscription expires and is no longer
1541 active. If this attribute is missing the subscription never expires.

1542 **11.2.4 Attribute notificationInterval**

1543 This attribute denotes the duration that a registry must wait between delivering successive
 1544 notifications to the client. The client specifies this attribute in order to control the
 1545 frequency of notification communication between registry and client. If this attribute is
 1546 missing, sending of notifications should happen as soon as relevant events occur.

1547 **11.2.5 Attribute selector**

1548 This attribute defines the selection criteria that determines which events match this
 1549 Subscription and are of interest to the User. The selector attribute references a pre-
 1550 defined query that is stored in the registry as an instance of the AdhocQuery class. This
 1551 AdhocQuery instance specifies or “selects” events that are of interest to the subscriber.
 1552 The AdhocQueryClass is described in section **Error! Reference source not found.**

1553 **11.2.6 Attribute startDate**

1554 This attribute denotes the time at which the subscription becomes active. If this attribute
 1555 is missing subscription starts immediately.

1556 **11.3 Class AdhocQuery**

1557 This abstract class is a sub-class of RegistryObject class and is the base class for all
 1558 AdhocQuery classes supported by the registry. Instances of this class represent an ad hoc
 1559 query that may be used for discover RegistryObjects within the registry. Instances of
 1560 AdhocQuery may be stored in registry like other RegistryObjects. Such stored
 1561 AdhocQuery instances are similar in purpose to the concept of stored procedures in
 1562 relational databases.

1563 **11.3.1 Method Summary**

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

Method Summary for RegistryObject	
String	getQuery ()
	Gets the query String for this AdhocQuery instances. This may be an SQL or Filter query string as described by [ebRS].

1566

1567

1568

1569 **11.4 Class Action**

1570 The Action class is an abstract base class that specifies what the registry must do when an
 1571 event matching the action’s Subscription transpires. A registry uses Actions within a
 1572 Subscription to asynchronously deliver event Notifications to the subscriber.

1573 If no Actions are defined within the Subscription that implies that the user does not wish
 1574 to be notified asynchronously by the registry and instead intends to periodically poll the
 1575 registry and pull the pending Notifications.
 1576 This class does not currently define any attributes or method.

1577 **11.5 Class NotifyAction**

1578 The NotifyAction class is a sub-class of Action class. An instance of NotifyAction
 1579 represents an Action that the registry must perform in order to notify the subscriber of a
 1580 Subscription of the events of interest to that subscriber.

1581 **11.5.1 Attribute Summary**

1582

Attribute	Data Type	Required	Default Value	Specified By	Mutable
endPoint	URI	YES		Client	
notificationOption	Enumeration	No	“ObjectRefs”	Client	Yes

1583

1584 **11.5.2 Attribute endPoint**

1585 This attribute specifies a URI that identifies an service end point that may be used by the
 1586 registry to deliver notifications. Currently this attribute can either be a “mailto” URI (e.g.
 1587 mailto:someone@acme.com) or an “urn:uuid” URI. If it is a “mailto” URI then the
 1588 registry must use the specified email address to deliver the notification via email. If it is a
 1589 “urn:uuid” URI then it must be a reference to a ServiceBinding object to a Service that
 1590 implements the RegistryClient interface as defined by [ebRS]. In this case the registry
 1591 must deliver the notification by invoking the onResponse method of the RegistryClient
 1592 interface.

1593 **11.5.3 Attribute notificationOption**

1594 This attribute controls the specific type of event notification content desired by the
 1595 subscriber. It is used by the subscriber to control the granularity of event notification
 1596 content communicated by the registry to the subscriber.

1597 **11.5.3.1 Pre-defined notificationOption Values**

1598 The following table lists pre-defined notificationOption values.

1599

Name	Description
ObjectRefs	Indicates that the subscriber wants to receive only references to RegistryObjects that match the Subscription within a notification.
Objects	Indicates that the subscriber wants to receive actual

	RegistryObjects that match the Subscription within a notification.
--	--

1600

1601

1602 12 Cooperating Registries Information Model

1603 This chapter describes the classes in the information model that support the cooperating
1604 registries capability.

1605 12.1.1 Class Registry

1606 Super Classes:

1607 RegistryEntry

1608

1609 Registry instances are used to represent a single physical OASIS ebXML registry.

1610 12.1.1.1 Attribute Summary

1611

Attribute	Data Type	Required	Default Value	Specified By	Mutable
catalogingLatency	duration	No	P1D (1 day)	Registry	Yes
eventNotificationSupported	boolean	No	false	Registry	Yes
objectRelocationSupported	boolean	No	false	Registry	Yes
objectReplicationSupported	boolean	No	false	Registry	Yes
operator	ObjectRef	Yes		Registry	Yes
replicationSyncLatency	duration	No	P1D (1 day)	Registry	Yes
specificationVersion	Sring8	Yes		Registry	Yes
sqlQuerySupported	boolean	No	false	Registry	Yes

1612

1613 12.1.1.2 Attribute catalogingLatency

1614 Each Registry instance may have an attribute named *catalogingLatency* that specifies the
1615 maximum latency between the time a submission is made to the registry and the time it
1616 gets cataloged by any cataloging services defined for the objects within the submission.

1617 12.1.1.3 Attribute eventNotificationSupported

1618 Each Registry instance may have an attribute named *eventNotificationSupported* that
1619 decalres whether the registry supports the optional Event Notification feature.

1620 12.1.1.4 Attribute objectRelocationSupported

1621 Each Registry instance may have an attribute named *objectRelocationSupported* that
1622 decalres whether the registry supports the optional Object Relocation feature.

1623 **12.1.1.5 Attribute objectReplicationSupported**

1624 Each Registry instance may have an attribute named *objectReplicationSupported* that
1625 decalres whether the registry supports the optional Object Replication feature.

1626 **12.1.1.6 Attribute operator**

1627 Each Registry instance must have an attribute named *operator* that is a reference to the
1628 Organization instance representing the organization for the registry's operator. Since the
1629 same Organization may operate multiple registries, it is possible that the home registry
1630 for the Organization referenced by operator may not be the local registry.

1631

1632 **12.1.1.7 Attribute replicationSyncLatency**

1633 Each Registry instance may have an attribute named *replicationSyncLatency* that
1634 specifies the maximum latency between the time when an original object changes and the
1635 time when its replica object within the registry gets updated to synchronize with the new
1636 state of the original object.

1637

1638 **12.1.1.8 Attribute specificationVersion**

1639 Each Registry instance must have an attribute named *specificationVersion* that is the
1640 version of the ebXML Registry Services Specification [ebRS].

1641

1642 **12.1.1.9 Attribute sqlQuerySupported**

1643 Each Registry instance may have an attribute named *sqlQuerySupported* that decalres
1644 whether the registry supports the optional SQL Query feature.

1645 **12.1.2 Class Federation**

1646 **Super Classes:**

1647 RegistryEntry

1648

1649 Federation instances are used to represent a registry federation.

1650 **12.1.2.1 Attribute Summary**

1651

Attribute	Data Type	Required	Default Value	Specified By	Mutable
replicationSync Latency	duration	No	P1D (1 day)	Client	Yes

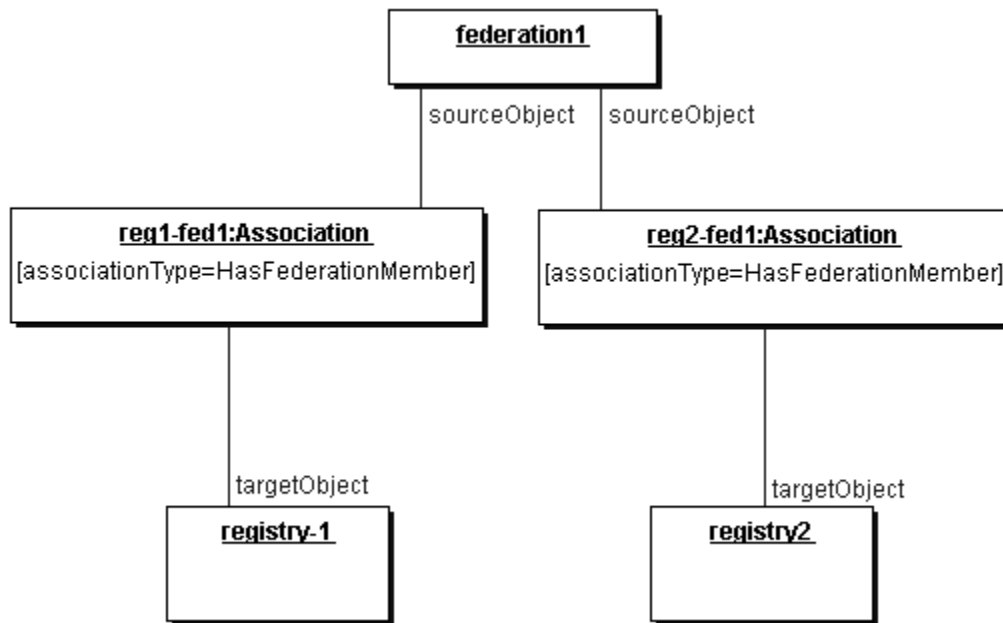
1652

1653 **12.1.2.2 Attribute replicationSyncLatency**

1654 Each Federation instance may specify a *replicationSyncLatency* attribute that describes
 1655 the time duration that is the amount of time within which a member of this Federation
 1656 must synchronize itself with the current state of the Federation. Members of the
 1657 Federation may use this parameter to periodically synchronize the federation metadata
 1658 they must cache locally about the state of the Federation and its members. Such
 1659 synchronization may be based upon the registry event notification capability.

1660 **12.1.3 Federation Configuration**

1661 A federation is created by the creation of a Federation instance. Membership of a registry
 1662 within a federation is established by creating an Association between the Registry
 1663 instances for the registry seeking membership with the Federation instance. The
 1664 Association must have its associationType be “HasFederationMember”, the federation
 1665 instance as its sourceObject and the Registry instance as its targetObject as shown in
 1666 Figure 14.



1667
 1668 Figure 14: Federation Information Model
 1669
 1670
 1671
 1672

1673 **13 Access Control Information Model**

1674 This chapter defines the Access Control Information Model used by the registry to
1675 control access to RegistryObjects and RepositoryItems managed by it.

1676 This specification first defines an abstract Access Control Model that enables access
1677 control policies to be defined and associated with RegistryObjects.

1678 Next, it defines a normative and required binding of that abstract model to [XACML].

1679 Finally, it defines how a registry may support additional bindings to custom access
1680 control technologies.

1681 **13.1 Terminology**

1682 The Access Control Model attempts to reuse terms defined by [XACML] wherever
1683 possible. The definition of some key terms are duplicated here from [XACML] for
1684 convenience of the reader:

1685 **Access** - Performing an **action**. An example is a user performing a *delete action* on a
1686 RegistryObject.

1687 **Access control** - Controlling **access** in accordance with a **policy**. An example is preventing a
1688 user from performing a *delete action* on a RegistryObject that is not owned by that user.

1689 **Action** - An operation on a **resource**. An example is the *delete action* on a RegistryObject.

1690 **Attribute** - Characteristic of a **subject**, **resource**, **action**. Some examples are:

- 1691 • *id attribute* of a subject.
- 1692 • *role attribute* of a subject.
- 1693 • *group attribute* of a subject.
- 1694 • *id attribute* of a RegistryObject resource.

1695

1696 **Policy** - A set of **rules**. May be a component of a **policy set**

1697 **Policy set** - A set of **policies**, other **policy sets**. May be a component of another **policy set**

1698 **Resource** - Data, service or system component. Examples are:

- 1699 • *A RegistryObject resource*
- 1700 • *A RepositoryItem resource*

1701

1702 **Subject** - An actor whose **attributes** may be referenced by within a Policy definition. Examples
1703 are:

- 1704 • A User instance within the registry

1705

1706 **13.2 Resources**

1707 A registry must control access to the following types of resources:

1708

- 1709 • *RegistryObject resource* is any instance of RegistryObject class or its sub-classes. Each RegistryObject resource references an Access Control Policy that controls all access to that object.
- 1710
- 1711
- 1712 • *RegistryEntry resource* is any instance of RepositoryItem class. By default, access control to a RepositoryItem is managed by the same Access Control Policy as its ExtrinsicObject.
- 1713
- 1714

1715

1716 A Registry must support each and every attribute of the RegistryObject class and all of its sub-classes within its Access Control Policies. In addition a registry must support the following additional resource attributes.

1717

1719 **13.2.1 Attribute owner**

1720 The *owner* attribute of a Resource carries the value of id attribute of the User instance within the registry that represents the owner of the resource.

1721

1722 **13.2.2 Attribute selector**

1723 The *selector* attribute of a Resource carries a string representing a query as define by a sub-type of AdhocQueryType in [ebRS].

1724

1725 **13.3 Actions**

1726 A registry must support the following actions as operations on RegistryObject and RepositoryItem resources managed by the registry.

1727

1728 **13.3.1 Create Action**

1729 The *create action* creates a RegistryObject or a RepositoryItem. A submitObjects operation performed on the LifeCycleManager interface of the registry result in a *create action*.

1730

1732 **13.3.2 Read Action**

1733 The *read action* reads a RegistryObject or a RepositoryItem without having any impact on its state. An operation performed on the QueryManager interface of the registry result in a *read action*.

1734

1735

1736 13.3.3 Update Action

1737 The *update action* updates or modifies the state of a RegistryObject or a RepositoryItem.
1738 An updateObjects operation performed on the LifeCycleManager interface of the registry
1739 result in a *update action*.

1740 13.3.4 Delete Action

1741 The *delete action* deletes a RegistryObject or a RepositoryItem. A removeObjects
1742 operation performed on the LifeCycleManager interface of the registry result in a *delete*
1743 *action*.

1744 13.3.5 Approve Action

1745 The *approve action* approves a RegistryObject. An approveObjects operation performed
1746 on the LifeCycleManager interface of the registry result in a *approve action*.

1747 13.3.6 Reference Action

1748 The *reference action* creates a reference to a RegistryObject. A submitObjects or
1749 updateObjects operation performed on the LifeCycleManager interface of the registry
1750 may result in a *reference action*. An example of a reference action is when an Association
1751 is created that references a RegistryObject resource as its source or target object.

1752 13.3.7 Deprecate Action

1753 The *deprecate action* deprecates a RegistryObject. A deprecateObjects operation
1754 performed on the LifeCycleManager interface of the registry result in a *deprecate action*.

1755 13.3.8 Undeprecate Action

1756 The *undeprecate action* undeprecates a previously deprecated RegistryObject. An
1757 undeprecateObjects operation performed on the LifeCycleManager interface of the
1758 registry result in a *undeprecate action*.

1759 13.4 Subjects

1760 A registry must support the following Subject attributes within its Access Control
1761 Policies. In addition a registry may support additional subject attributes.

1762 13.4.1 Attribute id

1763 The *identity* attribute of a Subject carries the value of id attribute of a User instance
1764 within the registry.

1765 **13.4.2 Attribute group**

1766 The *group* attribute of a Subject carries the value of the code attribute of a
1767 ClassificationNode within the canonical SubjectGroup ClassificationScheme (see
1768 appendix A.9) within the registry.

1769 **13.4.2.1 Assigning To Users to Groups**

1770 Arbitrary groups may be defined by extending the canonical SubjectGroup
1771 ClassificationScheme. Groups may be assigned to registered users by classifying their
1772 User instance with a ClassificationNode within the canonical SubjectGroup
1773 ClassificationScheme.
1774

1775 **13.4.3 Attribute role**

1776 The *role* attribute of a Subject carries the value of the code attribute of a
1777 ClassificationNode within the canonical SubjectRole ClassificationScheme (see appendix
1778 A.8) within the registry.
1779

1780 **13.4.3.1 Assigning Roles To Users**

1781 Arbitrary roles may be defined by extending the canonical SubjectRole
1782 ClassificationScheme. Roles may be assigned to registered users by classifying their User
1783 instance with a ClassificationNode within the canonical SubjectRole
1784 ClassificationScheme.

1785 **13.5 Use Cases for Access Control Policies**

1786 The following are some common use cases for access control policy:

1787 **13.5.1 Default Access Control Policy**

1788 Define a default access control policy that gives *read access* to any one and access to all
1789 actions to ContentOwner and Registry Administrator. This access control policy
1790 implicitly applies to any resource that does not explicitly have a custom Access Control
1791 Policy defined for it.

1792 **13.5.2 Restrict Read Access To Specified Subjects**

1793 Define a custom access control policy to restrict *read access* to a resource to specified
1794 user(s), group(s) and/or role(s).

1795 **13.5.3 Grant Update and/or Delete Access To Specified Subjects**

1796 Define a custom access control policy to grant *update* and/or *delete access* to a resource
1797 to specified user(s), group(s) and/or role(s).

1798 **13.6 Abstract Access Control Model**

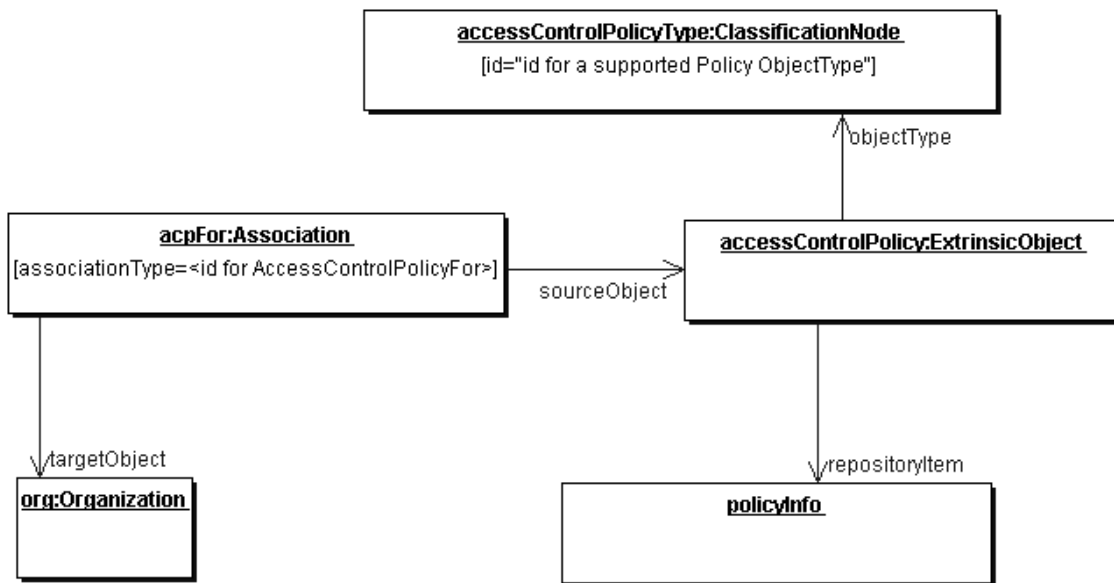
1799 Every RegistryObject is associated with exactly one Access Control Policy that governs
 1800 “who” is authorized to perform “what” action on that RegistryObject. The abstract
 1801 Access Control Model allows the Access Control Policy to be defined in any arbitrary
 1802 format as long as it is represented in the registry as a repositoryItem and its
 1803 corresponding ExtrinsicObject. The objectType attribute of this ExtrinsicObject must
 1804 reference the AccessControlPolicy node in the canonical ObjectType
 1805 ClassificationScheme or one of its descendents. This distinguishes Access Control Policy
 1806 objects from other ExtrinsicObject instances.

1807 **13.7 Access Control Policy for a RegistryObject**

1808 A RegistryObject may be associated with an Access Control Policy by a special
 1809 Association with the canonical associationType of “AccessControlPolicyFor”. This
 1810 association has the reference to the ExtrinsicObject representing the Access Control
 1811 Policy as the value of its sourceObject and has the reference to the RegistryObject as the
 1812 value of its targetObject attribute.

1813 If a RegistryObject does not have an Access Control Policy explicitly associated with it,
 1814 then it is implicitly associated with the default Access Control Policy defined for the
 1815 registry.

1816
 1817



1818
 1819

Figure 15: Instance Diagram for Abstract Access Control Information Model

1820 Figure 15 shows an instance diagram where an Organization instance *org* references an
1821 ExtrinsicObject instance *accessControlPolicy* as its Access Control Policy object. The
1822 *accessControlPolicy* object has its objectType attribute referencing a node in the
1823 canonical ObjectType ClassificationScheme that represents a supported Access Control
1824 Policy format. The *accessControlPolicy* ExtrinsicObject has a repositoryItem defining its
1825 access control policy information in a specific format.

1826 **13.7.1 Access Control Policy for a RepositoryItem**

1827 **By default, access control to a RepositoryItem is managed by the Access Control Policy**
1828 **associated with its ExtrinsicObject that provides metadata for the RepositoryItem. A RepositoryItem**
1829 **may have an Access Control Policy separate from its ExtrinsicObject. In such case, the Access**
1830 **Control Policy for the RepositoryItem is referenced via a Special Slot on its ExtrinsicObject. This**
1831 **special Slot has “repositoryItemACP” as its name and the id of the ExtrinsicObject representing the**
1832 **Access Control Policy for the RepositoryItem as its value.**

1833 **13.7.2 Default Access Control Policy**

1834 A registry must support the default Access Control Policy.

1835 The default access control policy applies to any RegistryObject that does not explicitly
1836 reference a specific access control policy via its accessControlPolicy attribute. This is the
1837 case when a RegistryObject has a null value for its accessControlPolicy attribute.

1838 The following list summarizes the default AccessControlPolicy semantic that a registry
1839 should implement:

- 1840 • Only a Registered User is granted access to create actions.
- 1841 • An unauthenticated Registry Client is granted access to read actions. The Registry
1842 must assign the default GuestReader role to such Registry Clients.
- 1843 • A Registered User has access to all actions on Registry Objects submitted by the
1844 Registered User. Such Registered Users have the role of ContentOwner for the
1845 RegistryObject.
- 1846 • The RegistryAdministrator and Registry Authority have access to all actions on all
1847 Registry Objects.

1848

1849 A registry may have a default access control policy that differs from the above semantics.

1850 **13.7.3 Root Access Control Policy**

1851 A registry must have a root Access Control Policy that bootstraps the Access Control
1852 Model by controlling access to Access Control Policies.

1853 As described in Figure 15, an access control policy is an ExtrinsicObject which contains
1854 a pointer to a repository item. The access control policies themselves are created,
1855 updated, and deleted.

1856 To define who may create access control policies pertaining to specified resources, it is
 1857 necessary to have one or more administrative Access Control Policies. Such policies
 1858 restrict Registry Users from creating access control policies to unauthorized resources.
 1859 This version of the Registry specifications defines a single Root Access Control Policy
 1860 that allows all actions on Access Control Policies for a resource under the following
 1861 conditions:

- 1862 • Subject has a role of ContentOwner for the resource
- 1863 • Subject has a role of RegistryAdministrator

1864

1865 **13.8 Access Control Model: [XACML] Binding**

1866 A registry may support custom access control policies based upon a normative though
 1867 optional binding of the Access Control Model to [XACML].

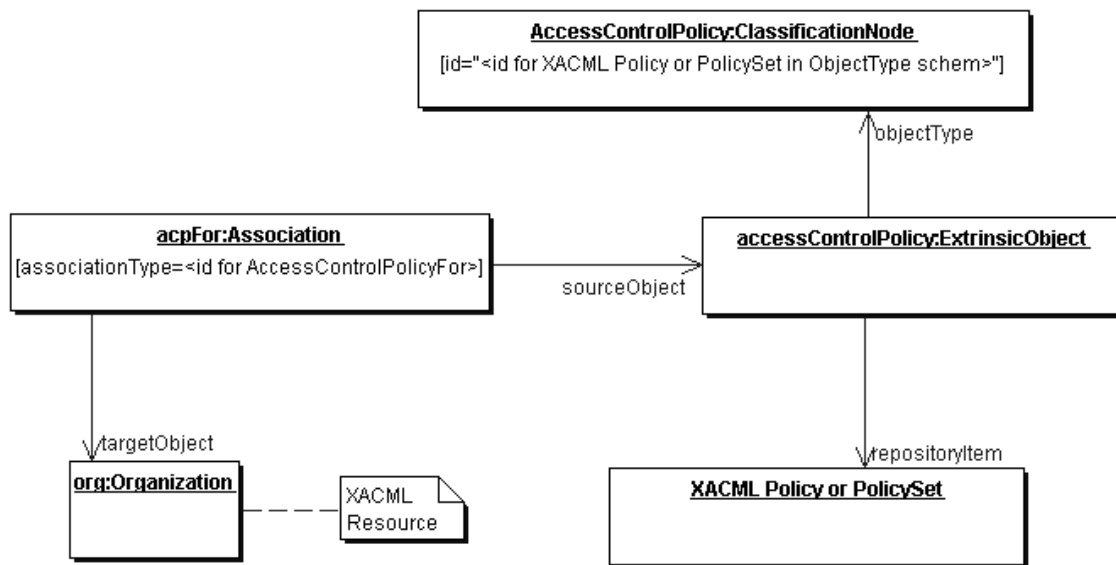
1868 This section defines the normative though optional binding of the abstract Access Control
 1869 Model to [XACML]. This section assumes the reader is familiar with [XACML].

1870 This binding to [XACML] enables a flexible access control mechanism that supports
 1871 access control policy definition from the simples to the most sophisticated use cases.

1872 In this binding the policyInfo repositoryItem in the abstract Access Control Model must
 1873 be one of the following:

- 1874 • A PolicySet as defined by [XACML]
- 1875 • A Policy as defined by [XACML]

1876



1877
 1878

Figure 16: Access Control Information Model: [XACML] Binding

1879 **13.8.1 Resource Binding**

1880 [XACML] defines an element called ResourceAttributeDesignator that identifies the type
1881 of resource attribute being specified in a ResourceMatch or Apply element.

1882 The resource attributes defined by the abstract Access Control Model map to the
1883 following ResourceAttributeDesignator definitions:

1884

Resource Attribute	ResourceAttributeDesignator	Data Type
owner	urn:oasis:names:tc:ebxml-regrep:2.5:rim:acp:resource:owner	http://www.w3.org/2001/XMLSchema#anyURI
selector	urn:oasis:names:tc:ebxml-regrep:2.5:rim:acp:resource:selector	http://www.w3.org/2001/XMLSchema#string
<attribute>	urn:oasis:names:tc:ebxml-regrep:2.5:rim:acp:resource:<attribute>	As defined by attribute definition

1885

Table 2: Resource Binding to [XACML]

1886 The resource attribute <attribute> in last row in the table represents any attribute defined
1887 by the RegistryObject type or one of its sub-types.

1888 **13.8.2 Action Binding**

1889 [XACML] defines an element called ActionAttributeDesignator that identifies the type of
1890 action being specified within in an ActionMatch or Apply element.

1891 The actions defined by the abstract Access Control Model map to the following
1892 AttributeId and AttributeValue in the ActionMatch definitions:

1893

Registry Action	ActionMatch.ActionAttributeDesignator.AttributeId	AttributeValue
Create	urn:oasis:names:tc:xacml:1.0:action:action-id	create
Read	urn:oasis:names:tc:xacml:1.0:action:action-id	read
Update	urn:oasis:names:tc:xacml:1.0:action:action-id	update
Delete	urn:oasis:names:tc:xacml:1.0:action:action-id	delete
Approve	urn:oasis:names:tc:xacml:1.0:action:action-id	approve
Deprecate	urn:oasis:names:tc:xacml:1.0:action:action-id	deprecate
Undeprecate	urn:oasis:names:tc:xacml:1.0:action:action-id	undeprecate

1894

Table 3: Action Binding to [XACML]1895 **13.8.3 Subject Binding**

1896 [XACML] defines an element called SubjectAttributeDesignator that identifies the type
1897 of subject attribute being specified in a SubjectMatch or Apply element.

1898

1899 The subjects defined by the abstract Access Control Model map to the following
1900 SubjectAttributeDesignator definitions:

1901

Subject Attribute	SubjectAttributeDesignator	Data Type
id	urn:oasis:names:tc:xacml:1.0:subject:subject-id	http://www.w3.org/2001/XMLSchema#anyURI
role	urn:oasis:names:tc:ebxml-regrep:2.5:rim:acp:subject:role	http://www.w3.org/2001/XMLSchema#string
group	urn:oasis:names:tc:ebxml-regrep:2.5:rim:acp:subject:group	http://www.w3.org/2001/XMLSchema#string

1902

Table 4: Subject Binding to [XACML]

1903 13.8.4 Constraints on [XACML] Binding

1904 This specification normatively defines the following constraints on the binding of the
 1905 Access Control Model to [XACML]. These constraints may be relaxed in future versions
 1906 of this specification.

- 1907 • All Policy and PolicySet definitions must reside with an ebXML registry as
 1908 RepositoryItems.

1909

1910 13.8.5 Examples of [XACML] Policies

1911 The following examples illustrate how [XACML] Policies may be used to address some
 1912 common use cases that have been identified.

1913 13.8.5.1 Default Access Control Policy

1914 The following Policy defines the default access control policy. This Policy must
 1915 implicitly apply to any resource that does not have an explicit Access Control Policy
 1916 defined.

1917 It consists of 3 rules, which in plain English are described as follows:

- 1918 1. Any subject can perform read action on any resource
- 1919 2. A subject may perform any action on a resource for which they have the role of
 1920 ContentOwner.
- 1921 3. A subject with role of RegistryAdministrator may perform any action on any
 1922 resource.

1923 The listing of the suggested default Access Control Policy follows:

1924

```
1925 <?xml version="1.0" encoding="UTF-8"?>
```

```
1926 <PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy"
1927 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1928 xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy
1929 C:\tmp\xacml\cs-xacml-schema-policy-01.xsd"
```

```
1930 PolicySetId="urn:oasis:names:tc:ebxml-regrep:2.5:rim:acp:policy:default-access-
1931 control-policy" PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
1932 combining-algorithm:permit-overrides">
```

```
1933 <Description>
```



```
1934     This PolicySet defines the default Access Control Policy for all registry
1935 resources.
1936     </Description>
1937     <Target>
1938         <Subjects>
1939             <AnySubject/>
1940         </Subjects>
1941         <Resources>
1942             <AnyResource/>
1943         </Resources>
1944         <Actions>
1945             <AnyAction/>
1946         </Actions>
1947     </Target>
1948     <Policy PolicyId="urn:oasis:names:tc:ebxml-
1949 regrep:2.5:rim:acp:policy:policyid:permit-anyone-to-read"
1950 RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1951 algorithm:permit-overrides">
1952     <Target>
1953         <Subjects>
1954             <AnySubject/>
1955         </Subjects>
1956         <Resources>
1957             <AnyResource/>
1958         </Resources>
1959         <Actions>
1960             <AnyAction/>
1961         </Actions>
1962     </Target>
1963     <Rule RuleId="urn:oasis:names:tc:ebxml-
1964 regrep:2.5:rim:acp:rule:ruleid:permit-anyone-to-read" Effect="Permit">
1965     <Description>
1966     Any Subject can perform read action on any resource.
1967     </Description>
1968     <Target>
1969         <Subjects>
1970             <AnySubject/>
1971         </Subjects>
1972         <Resources>
1973             <AnyResource/>
1974         </Resources>
1975         <Actions>
1976             <Action>
1977                 <ActionMatch
1978 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
```

```

1979         <AttributeValue
1980   DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
1981         <ActionAttributeDesignator
1982   AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1983   DataType="http://www.w3.org/2001/XMLSchema#string"/>
1984         </ActionMatch>
1985       </Action>
1986     </Actions>
1987   </Target>
1988 </Rule>
1989 </Policy>
1990   <Policy PolicyId="urn:oasis:names:tc:ebxml-
1991   regrep:2.5:rim:acp:policy:policyid:permit-owner-all"
1992   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1993   algorithm:permit-overrides">
1994     <Target>
1995       <Subjects>
1996         <AnySubject/>
1997       </Subjects>
1998       <Resources>
1999         <AnyResource/>
2000       </Resources>
2001       <Actions>
2002         <AnyAction/>
2003       </Actions>
2004     </Target>
2005     <Rule RuleId="urn:oasis:names:tc:ebxml-
2006   regrep:2.5:rim:acp:rule:ruleid:permit-owner-all" Effect="Permit">
2007       <Description>
2008         A Subject with role of ContenOwner can perform any action on
2009   resources owned by them.
2010       </Description>
2011       <Target>
2012         <Subjects>
2013           <AnySubject/>
2014         </Subjects>
2015         <Resources>
2016           <AnyResource/>
2017         </Resources>
2018         <Actions>
2019           <AnyAction/>
2020         </Actions>
2021       </Target>
2022       <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-
2023   equal">

```

```
2024         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-
2025 one-and-only">
2026             <SubjectAttributeDesignator
2027 AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
2028 DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
2029             </Apply>
2030         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-
2031 one-and-only">
2032             <ResourceAttributeDesignator
2033 AttributeId="urn:oasis:names:tc:ebxml-regrep:2.5:rim:acp:resource:owner"
2034 DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
2035             </Apply>
2036         </Condition>
2037     </Rule>
2038 </Policy>
2039 <Policy PolicyId="urn:oasis:names:tc:ebxml-
2040 regrep:2.5:rim:acp:policy:policyid:permit-registryadministrator-all"
2041 RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
2042 algorithm:permit-overrides">
2043     <Target>
2044         <Subjects>
2045             <AnySubject/>
2046         </Subjects>
2047         <Resources>
2048             <AnyResource/>
2049         </Resources>
2050         <Actions>
2051             <AnyAction/>
2052         </Actions>
2053     </Target>
2054     <Rule RuleId="urn:oasis:names:tc:ebxml-
2055 regrep:2.5:rim:acp:rule:ruleid:permit-registryadministrator-all" Effect="Permit">
2056         <Description>
2057             A Subject with role of RegistryAdministrator can perform any action on
2058             any resource.
2059         </Description>
2060         <Target>
2061             <Subjects>
2062                 <Subject>
2063                     <SubjectMatch
2064 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
2065                         <AttributeValue
2066 DataType="http://www.w3.org/2001/XMLSchema#string">RegistryAdministrator<
2067 /AttributeValue>
```

```

2068         <SubjectAttributeDesignator
2069 AttributeId="urn:oasis:names:tc:ebxml-regrep:2.5:rim:acp:subject:role"
2070 DataType="http://www.w3.org/2001/XMLSchema#string"/>
2071         </SubjectMatch>
2072     </Subject>
2073 </Subjects>
2074 <Resources>
2075     <AnyResource/>
2076 </Resources>
2077 <Actions>
2078     <AnyAction/>
2079 </Actions>
2080 </Target>
2081 </Rule>
2082 </Policy>
2083 </PolicySet>
2084

```

2085 13.8.5.2 Custom Access Control Policy

2086 The following Policy defines a custom access control policy to restrict *read access* to a
 2087 resource to specified user or role. It also grants update access to specified role.
 2088 It consists of 3 rules, which in plain English are described as follows:

- 2089
- 2090 1. A subject may perform any action on a resource for which they have the role of
 2091 ContentOwner. This reuses a Policy by reference from the default Access Control
 2092 PolicySet.
 - 2093 2. A subject with role of RegistryAdministrator may perform any action on any
 2094 resource. This reuses a Policy by reference from the default Access Control
 2095 PolicySet.
 - 2096 3. A subject with specified id may perform read actions on the resource. This
 2097 restricts read access to the specified subject.
 - 2098 4. A subject with role of Manager may perform update actions on the resource. This
 2099 relaxes update access restrictions to the specified subject.

2100
 2101 The listing of the custom Access Control Policy follows:

```

2102
2103 <?xml version="1.0" encoding="UTF-8"?>
2104 <PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy"
2105 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2106 xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy
2107 C:\tmp\xacml\cs-xacml-schema-policy-01.xsd"
2108 PolicySetId="urn:oasis:names:tc:ebxml-regrep:2.5:rim:acp:policy:restricted-
2109 access-control-policysset"
2110 PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-
2111 algorithm:permit-overrides">

```

```
2112     <Description>
2113     This PolicySet restricts the default Access Control Policy to limit read access
2114 to specified subjects.
2115     </Description>
2116     <Target>
2117         <Subjects>
2118             <AnySubject/>
2119         </Subjects>
2120         <Resources>
2121             <AnyResource/>
2122         </Resources>
2123         <Actions>
2124             <AnyAction/>
2125         </Actions>
2126     </Target>
2127     <PolicyIdReference>urn:oasis:names:tc:ebxml-
2128 regrep:2.5:rim:acp:policy:policyid:permit-owner-all</PolicyIdReference>
2129     <PolicyIdReference>urn:oasis:names:tc:ebxml-
2130 regrep:2.5:rim:acp:policy:policyid:permit-registryadministrator-
2131 all</PolicyIdReference>
2132     <Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy"
2133 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2134 xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy
2135 C:\tmp\xacml\cs-xacml-schema-policy-01.xsd"
2136 PolicyId="urn:oasis:names:tc:ebxml-regrep:2.5:rim:acp:policy:permit-delete-
2137 access-control-policy" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-
2138 combining-algorithm:permit-overrides">
2139     <Description>
2140     Allow Subject with specified id to perform delete action on any resource.
2141     </Description>
2142     <Target>
2143         <Subjects>
2144             <AnySubject/>
2145         </Subjects>
2146         <Resources>
2147             <AnyResource/>
2148         </Resources>
2149         <Actions>
2150             <AnyAction/>
2151         </Actions>
2152     </Target>
2153     <Rule RuleId="urn:oasis:names:tc:ebxml-
2154 regrep:2.5:rim:acp:rule:ruleid:permit-delete-rule" Effect="Permit">
2155     <Description>
2156     Allow Subject with specified id to perform delete action on any resource.
```

```
2157     </Description>
2158     <Target>
2159       <Subjects>
2160         <Subject>
2161           <SubjectMatch
2162             MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
2163             <AttributeValue
2164               DataType="http://www.w3.org/2001/XMLSchema#anyURI">urn:uuid:977d9380-
2165               00e2-4ce8-9cdc-d8bf6a4157be</AttributeValue>
2166             <SubjectAttributeDesignator
2167               AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
2168               DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
2169             </SubjectMatch>
2170           </Subject>
2171         </Subjects>
2172       <Resources>
2173         <AnyResource/>
2174       </Resources>
2175     <Actions>
2176       <Action>
2177         <ActionMatch
2178           MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
2179           <AttributeValue
2180             DataType="http://www.w3.org/2001/XMLSchema#string">delete</AttributeValue
2181           >
2182           <ActionAttributeDesignator
2183             AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
2184             DataType="http://www.w3.org/2001/XMLSchema#string"/>
2185           </ActionMatch>
2186         </Action>
2187       </Actions>
2188     </Target>
2189   </Rule>
2190 </Policy>
2191 <Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy"
2192 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2193 xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy
2194 C:\tmp\xacml\cs-xacml-schema-policy-01.xsd"
2195 PolicyId="urn:oasis:names:tc:ebxml-regrep:2.5:rim:acp:policy:permit-update-
2196 access-control-policy" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-
2197 combining-algorithm:permit-overrides">
2198   <Description>
2199     Allow Subjects with Manager role to perform update action on any
2200     resource.
2201   </Description>
```

```

2202     <Target>
2203         <Subjects>
2204             <AnySubject/>
2205         </Subjects>
2206         <Resources>
2207             <AnyResource/>
2208         </Resources>
2209         <Actions>
2210             <AnyAction/>
2211         </Actions>
2212     </Target>
2213     <Rule RuleId="urn:oasis:names:tc:ebxml-
2214 regrep:2.5:rim:acp:rule:ruleid:permit-update-rule" Effect="Permit">
2215         <Description>
2216             Allow Subjects with Manager role to perform read action on any
2217 resource.
2218         </Description>
2219         <Target>
2220             <Subjects>
2221                 <Subject>
2222                     <SubjectMatch
2223 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
2224                     <AttributeValue
2225 DataType="http://www.w3.org/2001/XMLSchema#string">Manager</AttributeVal
2226 ue>
2227                     <SubjectAttributeDesignator
2228 AttributeId="urn:oasis:names:tc:ebxml-regrep:2.5:rim:acp:subject:role"
2229 DataType="http://www.w3.org/2001/XMLSchema#string"/>
2230                     </SubjectMatch>
2231                 </Subject>
2232             </Subjects>
2233         </Resources>
2234             <AnyResource/>
2235         </Resources>
2236         <Actions>
2237             <Action>
2238                 <ActionMatch
2239 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
2240                 <AttributeValue
2241 DataType="http://www.w3.org/2001/XMLSchema#string">update</AttributeValue
2242 >
2243                 <ActionAttributeDesignator
2244 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
2245 DataType="http://www.w3.org/2001/XMLSchema#string"/>
2246                 </ActionMatch>

```

```
2247         </Action>
2248     </Actions>
2249 </Target>
2250 </Rule>
2251 </Policy>
2252 </PolicySet>
2253
```

2254 **13.8.6 Resolving PolicyReferences**

2255 An XACML PolicySet may reference XACML Policy objects defined outside the
2256 repository item containing the XACML PolicySet. A registry implementation must be
2257 able to resolve such references. To resolve such references efficiently a registry must be
2258 able to find the repository item containing the referenced Policy without having to load
2259 and search all Access Control Policies in the repository. This section describes the
2260 normative behaviour that enables a registry to resolve policy references efficiently.

2261 A registry must define a Content Cataloging Service for the canonical XACML PolicySet
2262 objectType. The PolicySet cataloging service must automatically catalog every PolicySet
2263 upon submission to contain a special Slot with name "ComposedPolicies". The value of
2264 this Slot must be a Collection where each element in the Collection is the id for a Policy
2265 object that is composed within the PolicySet.

2266 Thus a registry is able to use an ad hoc query to find the repositoryItem representing an
2267 XACML PolicySet that contains the Policy that is being referenced by another PolicySet.

2268 **13.8.7 ebXML Registry as a [XACML] Policy Store**

2269 So far we have defined how ebXML registries may use [XACML] to define Access
2270 Control Policies to control access to RegistryObject and RepositoryItem resources.

2271 An important side effect of the normative binding of the Access Control Model to
2272 [XACML] is that enterprises may also use ebXML Registry as a [XACML] Policy store
2273 to manage Policies for protecting resources outside the registry.

2274 In this use case, enterprises may submit [XACML] Policies and PolicySets as
2275 ExtrinsicObject-RepositoryItem pairs. These Policies may be accessed or referenced by
2276 their URL as defined by the HTTP binding of the ebXML Registry Services interface in
2277 [ebRS].

2278 **13.9 Access Control Model: Custom Binding**

2279 A registry may support bindings to policies describes in formats other than [XACML].
2280 The use of such policies sacrifices interoperability and is therefore discouraged. In such
2281 cases the RepositoryItem for the policy information may be in any format supported by
2282 the registry in an implementation specific manner.

2283 **Appendix A Canonical Classification Schemes**

2284 This section lists the canonical ClassificationSchemes that are required to be present in
2285 all ebXML Registries. These Canonical ClassificationSchemes may be extended by
2286 adding additional ClassificationNodes. However, no ClassificationNode defined
2287 normatively in the links below may be modified within a registry. In particular they must
2288 preserve their canonical id attributes in all registries.

2289 **A.1 ObjectType ClassificationScheme**

2290 [http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/
2291 SubmitObjectsRequest_ObjectTypeScheme.xml](http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/SubmitObjectsRequest_ObjectTypeScheme.xml)

2292 **A.2 AssociationType ClassificationScheme**

2293 [http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/
2294 SubmitObjectsRequest_AssociationTypeScheme.xml](http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/SubmitObjectsRequest_AssociationTypeScheme.xml)

2295 **A.3 PhoneType ClassificationScheme**

2296 [http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/
2297 SubmitObjectsRequest_PhoneTypeScheme.xml](http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/SubmitObjectsRequest_PhoneTypeScheme.xml)

2298 **A.4 EmailType ClassificationScheme**

2299 [http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/
2300 SubmitObjectsRequest_EmailTypeScheme.xml](http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/SubmitObjectsRequest_EmailTypeScheme.xml)

2301 **A.5 ContentManagementService ClassificationScheme**

2302 [http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/
2303 SubmitObjectsRequest_CMSScheme.xml](http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/SubmitObjectsRequest_CMSScheme.xml)

2304 **A.6 ErrorHandlingModel ClassificationScheme**

2305 [http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/
2306 SubmitObjectsRequest_ErrorHandlingModelScheme.xml](http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/SubmitObjectsRequest_ErrorHandlingModelScheme.xml)

2307 **A.7 InvocationModel ClassificationScheme**

2308 [http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/
2309 SubmitObjectsRequest_InvocationModelScheme.xml](http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/SubmitObjectsRequest_InvocationModelScheme.xml)

2310 **A.8 SubjectRole ClassificationScheme**

2311 [http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/
2312 SubmitObjectsRequest_SubjectRoleScheme.xml](http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/SubmitObjectsRequest_SubjectRoleScheme.xml)

- 2313 **A.9 SubjectGroup ClassificationScheme**
- 2314 [http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/
SubmitObjectsRequest_SubjectGroupScheme.xml](http://www.oasis-open.org/committees/regrep/documents/2.5/canonical/SubmitObjectsRequest_SubjectGroupScheme.xml)
- 2316
- 2317

- 2318 **14 References**
- 2319 [ebGLOSS] ebXML Glossary,
2320 http://www.ebxml.org/documents/199909/terms_of_reference.htm
- 2321 [OAS] OASIS Information Model
2322 <http://xsun.sdct.itl.nist.gov/regrep/OasisRegrepSpec.pdf>
- 2323 [ISO] ISO 11179 Information Model
2324 <http://208.226.167.205/SC32/jtc1sc32.nsf/576871ad2f11bba785256621005419d7/b83fc7816a6064c68525690e0065f913?OpenDocument>
2325
- 2326 [BRA97] IETF (Internet Engineering Task Force). RFC 2119: Key words for use in
2327 RFCs to Indicate Requirement Levels
2328 <http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc2119.html>
- 2329 [ebRS] ebXML Registry Services Specification
2330 <http://www.oasisopen.org/committees/regrep/documents/2.5/specs/ebRS.pdf>
- 2331 [ebCPP] ebXML Collaboration-Protocol Profile and Agreement Specification
2332 <http://www.ebxml.org/specrafts/>
2333
- 2334 [UUID] DCE 128 bit Universal Unique Identifier
2335 http://www.opengroup.org/onlinepubs/009629399/apdx.htm#tagcjh_20
2336 <http://www.opengroup.org/publications/catalog/c706.htm><http://www.w3.org/TR/REC-xml>
2337
- 2338 [XPATH] XML Path Language (XPath) Version 1.0
2339 <http://www.w3.org/TR/xpath>
2340
- 2341 [XACML] OASIS eXtensible Access Control Markup Language (XACML) Version 1.0
2342 <http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-01.pdf>
2343
- 2344 [NCName] Namespaces in XML 19990114
2345 <http://www.w3.org/TR/REC-xml-names/#NT-NCName>.

2346 15 Disclaimer

2347 The views and specification expressed in this document are those of the authors and are
2348 not necessarily those of their employers. The authors and their employers specifically
2349 disclaim responsibility for any problems arising from correct or incorrect implementation
2350 or use of this design.

2351 **16 Contact Information**

2352

2353 Team Leader

2354 Name: Kathryn R. Breininger
2355 Company: The Boeing Company
2356 Street: P.O. Box 3707 MC 62-LC
2357 City, State, Postal Code: Seattle, WA 98124-2207
2358 Country: USA
2359 Phone: 425-965-0182
2360 Email: kathryn.r.breininger@boeing.com

2361

2362 Editor

2363 Name: Sally Fuger
2364 Company: Automotive Industry Action Group
2365 Street: 26200 Lahser Road, Suite 200
2366 City, State, Postal Code: Southfield, MI 48034
2367 Country: USA
2368 Phone: (248) 358-9744
2369 Email: sfuger@aiag.org

2370

2371 Technical Editor

2372 Name: Farrukh S. Najmi
2373 Company: Sun Microsystems
2374 Street: 1 Network Dr., MS BUR02-302
2375 City, State, Postal Code: Burlington, MA, 01803-0902
2376 Country: USA
2377 Phone: (781) 442-9017
2378 Email: farrukh.najmi@sun.com

2379

2380 Technical Editor

2381 Name: Nikola Stojanovic
2382 Company: Metaspaces Consulting Street:
2383 101 Pineview Terrace
2384 City, State, Postal Code: Ithaca, NY, 14850
2385 Country: USA
2386 Phone: (607) 273-2224
2387 Email: nikola.stojanovic@acm.org

2388

2389

2390 **17 Copyright Statement**

2391 OASIS takes no position regarding the validity or scope of any intellectual property or
2392 other rights that might be claimed to pertain to the implementation or use of the
2393 technology described in this document or the extent to which any license under such
2394 rights might or might not be available; neither does it represent that it has made any effort
2395 to identify any such rights. Information on OASIS's procedures with respect to rights in
2396 OASIS specifications can be found at the OASIS website. Copies of claims of rights
2397 made available for publication and any assurances of licenses to be made available, or the
2398 result of an attempt made to obtain a general license or permission for the use of such
2399 proprietary rights by implementors or users of this specification, can be obtained from the
2400 OASIS Executive Director.

2401
2402 OASIS invites any interested party to bring to its attention any copyrights, patents or
2403 patent applications, or other proprietary rights which may cover technology that may be
2404 required to implement this specification. Please address the information to the OASIS
2405 Executive Director.

2406
2407 Copyright ©The Organization for the Advancement of Structured Information Standards
2408 [OASIS] 2002. All Rights Reserved.

2409 This document and translations of it may be copied and furnished to others, and
2410 derivative works that comment on or otherwise explain it or assist in its implementation
2411 may be prepared, copied, published and distributed, in whole or in part, without
2412 restriction of any kind, provided that the above copyright notice and this paragraph are
2413 included on all such copies and derivative works. However, this document itself may not
2414 be modified in any way, such as by removing the copyright notice or references to
2415 OASIS, except as needed for the purpose of developing OASIS specifications, in which
2416 case the procedures for copyrights defined in the OASIS Intellectual Property Rights
2417 document must be followed, or as required to translate it into languages other than
2418 English.

2419 The limited permissions granted above are perpetual and will not be revoked by OASIS
2420 or its successors or assigns.

2421 This document and the information contained herein is provided on an "AS IS" basis and
2422 OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING
2423 BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE
2424 INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
2425 WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR
2426 PURPOSE."