

ebXML Registry Information Model Version 3.0

OASIS Standard, 2 May, 2005

Document identifier:
regrep-rim-3.0-os

Location:
<http://docs.oasis-open.org/regrep-rim/v3.0/>

Editors:

Name	Affiliation
Sally Fuger	Individual
Farrukh Najmi	Sun Microsystems
Nikola Stojanovic	RosettaNet

Contributors:

Name	Affiliation
Diego Ballve	Individual
Ivan Bedini	France Telecom
Kathryn Breininger	The Boeing Company
Joseph Chiusano	Booz Allen Hamilton
Peter Kacandes	Adobe Systems
Paul Macias	LMI Government Consulting
Carl Mattocks	CHECKMi
Matthew MacKenzie	Adobe Systems
Monica Martin	Sun Microsystems
Richard Martell	Galdos Systems Inc
Duane Nickull	Adobe Systems

12

13 **Abstract:**

14 This document defines the types of metadata and content that can be stored in an ebXML
15 Registry.

16 A separate document, ebXML Registry: Service and Protocols [ebRS], defines the services and
17 protocols for an ebXML Registry.

18

19 **Status:**

20 This document is an OASIS ebXML Registry Technical Committee Approved Draft
21 Specification.

22 Committee members should send comments on this specification to the [regrep@lists.oasis-](mailto:regrep@lists.oasis-open.org)
23 [open.org](mailto:regrep@lists.oasis-open.org) list. Others should subscribe to and send comments to the [regrep-](mailto:regrep-comment@lists.oasis-open.org)
24 [comment@lists.oasis-open.org](mailto:regrep-comment@lists.oasis-open.org) list. To subscribe, send an email message to [regrep-comment-](mailto:regrep-comment-request@lists.oasis-open.org)
25 [request@lists.oasis-open.org](mailto:regrep-comment-request@lists.oasis-open.org) with the word "subscribe" as the body of the message.

26 For information on whether any patents have been disclosed that may be essential to
27 implementing this specification, and any offers of patent licensing terms, please refer to the
28 Intellectual Property Rights section of the OASIS ebXML Registry TC web page
29 (<http://www.oasis-open.org/committees/regrep/>).

Table of Contents

30		
31	1 Introduction.....	10
32	1.1 Audience.....	10
33	1.2 Terminology.....	10
34	1.3 Notational Conventions.....	10
35	1.3.1 UML Diagrams.....	10
36	1.3.2 Identifier Placeholders.....	10
37	1.3.3 Constants.....	10
38	1.3.4 Bold Text.....	11
39	1.3.5 Example Values.....	11
40	1.4 XML Schema Conventions.....	11
41	1.4.1 Schemas Defined by ebXML Registry.....	11
42	1.4.2 Schemas Used By ebXML Registry.....	12
43	1.5 RepositoryItems and RegistryObjects.....	13
44	1.6 Canonical ClassificationSchemes.....	13
45	1.7 Registry Information Model: Overview.....	15
46	1.7.1 Class Relationships View.....	15
47	1.7.2 Class Inheritance View.....	15
48	1.7.2.1 Class Identifiable.....	16
49	2 Core Information Model.....	18
50	2.1 Attributes of Information Model Classes.....	18
51	2.2 Data Types.....	18
52	2.3 Internationalization (I18N) Support.....	19
53	2.3.1 Class InternationalString.....	19
54	2.3.1.1 Attribute Summary.....	19
55	2.3.1.2 Attribute localizedStrings.....	19
56	2.3.2 Class LocalizedString.....	19
57	2.3.2.1 Attribute Summary.....	20
58	2.3.2.2 Attribute lang.....	20
59	2.3.2.3 Attribute charset.....	20
60	2.3.2.4 Attribute value.....	20
61	2.4 Class Identifiable.....	20
62	2.4.1 Attribute Summary.....	20
63	2.4.2 Attribute id.....	20
64	2.4.3 Attribute home.....	20
65	2.4.4 Attribute slots.....	21
66	2.5 Class RegistryObject.....	21
67	2.5.1 Attribute Summary.....	21
68	2.5.2 Composed Object.....	21
69	2.5.3 Attribute classifications.....	22
70	2.5.4 Attribute description.....	22
71	2.5.5 Attribute externalIdentifier.....	22
72	2.5.6 Attribute lid.....	22
73	2.5.7 Attribute name.....	23
74	2.5.8 Attribute objectType.....	23

75	2.5.9 Attribute status.....	23
76	2.5.9.1 Pre-defined RegistryObject Status Types.....	23
77	2.5.10 Attribute versionInfo.....	24
78	2.6 Class VersionInfo	24
79	2.6.1 Attribute Summary.....	24
80	2.6.2 Attribute versionName.....	24
81	2.6.3 Attribute comment.....	24
82	2.7 Class ObjectRef.....	24
83	2.7.1 Attribute Summary.....	24
84	2.7.2 Attribute id.....	25
85	2.7.3 Attribute home.....	25
86	2.7.3.1 Local Vs. Remote ObjectRefs.....	25
87	2.7.4 Attribute createReplica.....	25
88	2.8 Class Slot	25
89	2.8.1 Attribute Summary	25
90	2.8.2 Attribute name.....	25
91	2.8.3 Attribute slotType.....	25
92	2.8.4 Attribute values.....	26
93	2.9 Class ExtrinsicObject	26
94	2.9.1 Attribute Summary.....	26
95	2.9.2 Attribute contentVersionInfo.....	26
96	2.9.3 Attribute isOpaque.....	26
97	2.9.4 Attribute mimeType.....	26
98	2.10 Class RegistryPackage.....	26
99	2.10.1 Attribute Summary.....	26
100	2.11 Class ExternalIdentifier.....	27
101	2.11.1 Attribute Summary.....	27
102	2.11.2 Attribute identificationScheme.....	27
103	2.11.3 Attribute registryObject.....	27
104	2.11.4 Attribute value.....	27
105	2.12 Class ExternalLink	27
106	2.12.1 Attribute Summary.....	27
107	2.12.2 Attribute externalURI.....	27
108	3 Association Information Model.....	28
109	3.1 Example of an Association.....	28
110	3.2 Source and Target Objects.....	28
111	3.3 Association Types.....	28
112	3.4 Intramural Association.....	28
113	3.5 Extramural Association.....	29
114	3.5.1 Controlling Extramural Associations.....	30
115	3.6 Class Association	30
116	3.6.1 Attribute Summary.....	30
117	3.6.2 Attribute associationType.....	31
118	3.6.3 Attribute sourceObject.....	31
119	3.6.4 Attribute targetObject.....	31

120	4	Classification Information Model.....	32
121	4.1	Class ClassificationScheme.....	33
122	4.1.1	Attribute Summary.....	33
123	4.1.2	Attribute isInternal.....	34
124	4.1.3	Attribute nodeType.....	34
125	4.2	Class ClassificationNode	34
126	4.2.1	Attribute Summary.....	34
127	4.2.2	Attribute parent.....	34
128	4.2.3	Attribute code.....	35
129	4.2.4	Attribute path.....	35
130	4.2.5	Canonical Path Syntax.....	35
131	4.2.5.1	Example of Canonical Path Representation.....	35
132	4.2.5.2	Sample Geography Scheme.....	35
133	4.3	Class Classification	36
134	4.3.1	Attribute Summary.....	36
135	4.3.2	Attribute classificationScheme.....	36
136	4.3.3	Attribute classificationNode.....	36
137	4.3.4	Attribute classifiedObject.....	36
138	4.3.5	Attribute nodeRepresentation.....	36
139	4.3.6	Context Sensitive Classification.....	37
140	4.4	Example of Classification Schemes.....	38
141	5	Provenance Information Model.....	39
142	5.1	Class Person.....	39
143	5.1.1	Attribute Summary.....	39
144	5.1.2	Attribute addresses.....	39
145	5.1.3	Attribute emailAddresses.....	39
146	5.1.4	Attribute personName.....	39
147	5.1.5	Attribute telephoneNumbers.....	39
148	5.2	Class User.....	40
149	5.2.1	Associating Users With Organizations.....	40
150	5.3	Class Organization.....	40
151	5.3.1	Attribute Summary.....	40
152	5.3.2	Attribute addresses.....	41
153	5.3.3	Attribute emailAddresses.....	41
154	5.3.4	Attribute parent.....	41
155	5.3.5	Attribute primaryContact.....	41
156	5.3.6	Attribute telephoneNumbers.....	41
157	5.4	Associating Organizations With RegistryObjects.....	41
158	5.5	Class PostalAddress.....	42
159	5.5.1	Attribute Summary.....	42
160	5.5.2	Attribute city.....	42
161	5.5.3	Attribute country.....	42
162	5.5.4	Attribute postalCode.....	42
163	5.5.5	Attribute stateOrProvince.....	43
164	5.5.6	Attribute street.....	43

165	5.5.7 Attribute streetNumber.....	43
166	5.6 Class TelephoneNumber.....	43
167	5.6.1 Attribute Summary.....	43
168	5.6.2 Attribute areaCode.....	43
169	5.6.3 Attribute countryCode.....	43
170	5.6.4 Attribute extension.....	43
171	5.6.5 Attribute number.....	43
172	5.6.6 Attribute phoneType.....	43
173	5.7 Class EmailAddress.....	44
174	5.7.1 Attribute Summary.....	44
175	5.7.2 Attribute address.....	44
176	5.7.3 Attribute type.....	44
177	5.8 Class PersonName.....	44
178	5.8.1 Attribute Summary.....	44
179	5.8.2 Attribute firstName.....	44
180	5.8.3 Attribute lastName.....	44
181	5.8.4 Attribute middleName.....	44
182	6 Service Information Model.....	45
183	6.1 Class Service.....	45
184	6.1.1 Attribute Summary.....	45
185	6.1.2 Attribute serviceBindings.....	45
186	6.2 Class ServiceBinding.....	45
187	6.2.1 Attribute Summary.....	46
188	6.2.2 Attribute accessURI.....	46
189	6.2.3 Attribute service.....	46
190	6.2.4 Attribute specificationLinks.....	46
191	6.2.5 Attribute targetBinding.....	46
192	6.3 Class SpecificationLink.....	46
193	6.3.1 Attribute Summary.....	46
194	6.3.2 Attribute serviceBinding.....	47
195	6.3.3 Attribute specificationObject.....	47
196	6.3.4 Attribute usageDescription.....	47
197	6.3.5 Attribute usageParameters.....	47
198	7 Event Information Model.....	48
199	7.1 Class AuditableEvent.....	48
200	7.1.1 Attribute Summary.....	48
201	7.1.2 Attribute eventType.....	49
202	7.1.2.1 Pre-defined Auditable Event Types.....	49
203	7.1.3 Attribute affectedObjects.....	49
204	7.1.4 Attribute requestId.....	49
205	7.1.5 Attribute timestamp.....	49
206	7.1.6 Attribute user.....	49
207	7.2 Class Subscription.....	49
208	7.2.1 Attribute Summary.....	50
209	7.2.2 Attribute actions.....	50

210	7.2.3 Attribute endTime.....	50
211	7.2.4 Attribute notificationInterval.....	50
212	7.2.5 Attribute selector.....	50
213	7.2.5.1 Specifying Selector Query Parameters.....	50
214	7.2.6 Attribute startTime.....	50
215	7.3 Class AdhocQuery.....	50
216	7.3.1 Attribute Summary.....	51
217	7.3.2 Attribute queryExpression.....	51
218	7.4 Class QueryExpression.....	51
219	7.4.1 Attribute Summary.....	51
220	7.4.2 Attribute queryLanguage.....	51
221	7.4.3 Attribute <any>.....	51
222	7.5 Class Action.....	51
223	7.6 Class NotifyAction.....	52
224	7.6.1 Attribute Summary.....	52
225	7.6.2 Attribute endPoint.....	52
226	7.6.3 Attribute notificationOption.....	52
227	7.6.3.1 Pre-defined notificationOption Values.....	52
228	7.7 Class Notification.....	53
229	7.7.1 Attribute Summary.....	53
230	7.7.2 Attribute subscription.....	53
231	7.7.3 Attribute registryObjectList.....	53
232	8 Cooperating Registries Information Model.....	54
233	8.1 Class Registry.....	54
234	8.1.1.1 Attribute Summary.....	54
235	8.1.2 Attribute catalogingLatency.....	54
236	8.1.3 Attribute conformanceProfile.....	54
237	8.1.4 Attribute operator.....	54
238	8.1.5 Attribute replicationSyncLatency.....	54
239	8.1.6 Attribute specificationVersion.....	54
240	8.2 Class Federation.....	55
241	8.2.1.1 Attribute Summary.....	55
242	8.2.2 Attribute replicationSyncLatency.....	55
243	8.2.3 Federation Configuration.....	55
244	9 Access Control Information Model.....	57
245	9.1 Terminology.....	57
246	9.2 Use Cases for Access Control Policies.....	57
247	9.2.1 Default Access Control Policy.....	58
248	9.2.2 Restrict Read Access To Specified Subjects.....	58
249	9.2.3 Grant Update and/or Delete Access To Specified Subjects.....	58
250	9.2.4 Reference Access Control.....	58
251	9.3 Resources.....	58
252	9.3.1 Resource Attribute: owner.....	58
253	9.3.2 Resource Attribute: selector.....	58
254	9.3.3 Resource Attribute: <attribute>.....	58
255	9.4 Actions.....	58

256	9.4.1 Create Action.....	59
257	9.4.2 Read Action.....	59
258	9.4.3 Update Action.....	59
259	9.4.4 Delete Action.....	59
260	9.4.5 Approve Action.....	59
261	9.4.6 Reference Action.....	59
262	9.4.7 Deprecate Action.....	59
263	9.4.8 Undeprecate Action.....	59
264	9.4.9 Action Attribute: action-id.....	59
265	9.4.10 Action Attribute: reference-source.....	59
266	9.4.11 Action Attribute: reference-source-attribute.....	60
267	9.5 Subjects.....	60
268	9.5.1 Attribute id.....	60
269	9.5.2 Attribute group.....	60
270	9.5.2.1 Assigning Groups To Users	60
271	9.5.3 Attribute role.....	60
272	9.5.3.1 Assigning Roles To Users.....	60
273	9.6 Abstract Access Control Model.....	60
274	9.6.1 Access Control Policy for a RegistryObject.....	61
275	9.6.2 Access Control Policy for a RepositoryItem.....	61
276	9.6.3 Default Access Control Policy.....	61
277	9.6.4 Root Access Control Policy.....	62
278	9.6.5 Performance Implications.....	62
279	9.7 Access Control Model: XACML Binding.....	62
280	9.7.1 Resource Binding.....	63
281	9.7.2 Action Binding.....	63
282	9.7.3 Subject Binding.....	64
283	9.7.4 Function classification-node-compare.....	64
284	9.7.5 Constraints on XACML Binding.....	65
285	9.7.6 Example: Default Access Control Policy.....	65
286	9.7.7 Example: Custom Access Control Policy.....	69
287	9.7.8 Example: Package Membership Access Control.....	72
288	9.7.9 Resolving Policy References.....	74
289	9.7.10 ebXML Registry as a XACML Policy Store.....	74
290	9.8 Access Control Model: Custom Binding.....	75
291	10 References.....	76
292	10.1 Normative References.....	76
293	10.2 Informative References.....	76
294		

Illustration Index

Figure 1: Information Model Relationships View.....	15
Figure 2: Information Model Inheritance View.....	16
Figure 3: Example of RegistryObject Association	28
Figure 4: Example of Intramural Association.....	29
Figure 5: Example of Extramural Association.....	30
Figure 6: Example showing a Classification Tree.....	32
Figure 7: Information Model Classification View.....	33
Figure 8: Classification Instance Diagram.....	33
Figure 9: Context Sensitive Classification.....	37
Figure 10: User Affiliation With Organization Instance Diagram.....	40
Figure 11: Organization to RegistryObject Association Instance Diagram.....	42
Figure 12: Service Information Model.....	45
Figure 13: Event Information Model.....	48
Figure 14: Federation Information Model.....	55
Figure 15: Instance Diagram for Abstract Access Control Information Model.....	61
Figure 16: Access Control Information Model: [XACML] Binding.....	63

296 1 Introduction

297 An ebXML Registry is an information system that securely manages any content type and the
298 standardized metadata that describes it.

299 The ebXML Registry provides a set of services that enable sharing of content and metadata between
300 organizational entities in a federated environment.

301 This document defines the types of metadata and content that can be stored in an ebXML Registry.

302 A separate document, ebXML Registry: Services and Protocols [ebRS], defines the services provided
303 by an ebXML Registry and the protocols used by clients of the registry to interact with these services.

304 1.1 Audience

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

- 306 • Implementers of ebXML Registry Services
- 307 • Implementers of ebXML Registry Clients

308 1.2 Terminology

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

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

315 1.3 Notational Conventions

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

318 1.3.1 UML Diagrams

319 Unified Modeling Language [UML] diagrams are used as a way to concisely describe concepts. They
320 are not intended to convey any specific Implementation or methodology requirements.

321 1.3.2 Identifier Placeholders

322 Listings may contain values that reference ebXML Registry objects by their id attribute. These id values
323 uniquely identify the objects within the ebXML Registry. For convenience and better readability, these
324 key values are replaced by meaningful textual variables to represent such id values.

325 For example, the placeholder in the listing below refers to the unique id defined for an example Service
326 object:

327

```
328 <rim:Service id="${EXAMPLE_ SERVICE_ID}">
```

329 1.3.3 Constants

330 Constant values are printed in the Courier New font always, regardless of whether they are defined
331 by this document or a referenced document.

332 **1.3.4 Bold Text**

333 Bold text is used in listings to highlight those aspects that are most relevant to the issue being
334 discussed. In the listing below, an example value for the contentLocator slot is shown in italics
335 if that is what the reader should focus on in the listing:

```
336  
337 <rim:Slot name="urn:oasis:names:tc:ebxml-  
338 regrep:rim:RegistryObject:contentLocator">  
339 ...  
340 </rim:Slot>
```

341

342 **1.3.5 Example Values**

343 These values are represented in *italic* font. In the listing below, an example value for the
344 contentLocator slot is shown in italics:

```
345  
346 <rim:Slot name="urn:oasis:names:tc:ebxml-  
347 regrep:rim:RegistryObject:contentLocator">  
348 <rim:ValueList>  
349 <rim:Value>http://example.com/myschema.xsd</rim:Value>  
350 </rim:ValueList>  
351 </rim:Slot>
```

352

353 **1.4 XML Schema Conventions**

354 This specification uses schema documents conforming to W3C XML Schema [Schema1] and normative
355 text to describe the syntax and semantics of XML-encoded objects and protocol messages. In cases of
356 disagreement between the ebXML Registry schema documents and schema listings in this
357 specification, the schema documents take precedence. Note that in some cases the normative text of
358 this specification imposes constraints beyond those indicated by the schema documents.

359 Conventional XML namespace prefixes are used throughout this specification to stand for their
360 respective namespaces as follows, whether or not a namespace declaration is present in the example.
361 The use of these namespace prefixes in instance documents is non-normative. However, for
362 consistency and understandability instance documents SHOULD use these namespace prefixes.

363 **1.4.1 Schemas Defined by ebXML Registry**

364

Prefix	XML Namespace	Comments
rim:	urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0	This is the Registry Information Model namespace [ebRIM]. The prefix is generally elided in mentions of Registry Information Model elements in text.
rs:	urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0	This is the ebXML Registry namespace that defines base types for registry service requests and responses [ebRS]. The prefix is generally elided in mentions of ebXML Registry protocol-related elements in text.
query:	urn:oasis:names:tc:ebxml-regrep:xsd:query:3.0	This is the ebXML Registry query namespace that is used in the query protocols used between clients and the QueryManager service [ebRS].

Prefix	XML Namespace	Comments
lcm:	urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0	This is the ebXML Registry Life Cycle Management namespace that is used in the life cycle management protocols used between clients and the LifeCycleManager service [ebRS].
cms:	urn:oasis:names:tc:ebxml-regrep:xsd:cms:3.0	This is the ebXML Registry Content Management Services namespace that is used in the content management protocols used between registry and pluggable content management services [ebRS].

365

366 1.4.2 Schemas Used By ebXML Registry

367

Prefix	XML Namespace	Comments
saml:	urn:oasis:names:tc:SAML:2.0:assertion	This is the SAML V2.0 assertion namespace [SAMLCore]. The prefix is generally elided in mentions of SAML assertion-related elements in text.
samlp:	urn:oasis:names:tc:SAML:2.0:protocol	This is the SAML V2.0 protocol namespace [SAMLCore]. The prefix is generally elided in mentions of XML protocol-related elements in text.
ecp:	urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp	This is the SAML V2.0 Enhanced Client Proxy profile namespace, specified in this document and in a schema [SAMLECP-xsd].
ds:	http://www.w3.org/2000/09/xmldsig#	This is the XML Signature namespace [XMLSig].
xenc:	http://www.w3.org/2001/04/xmlenc#	This is the XML Encryption namespace [XMLEnc].
SOAP-ENV:	http://schemas.xmlsoap.org/soap/envelope	This is the SOAP V1.1 namespace [SOAP1.1].
paos:	urn:liberty:paos:2003-08	This is the Liberty Alliance PAOS (reverse SOAP) namespace.
xsi:	http://www.w3.org/2001/XMLSchema-instance	This namespace is defined in the W3C XML Schema specification [Schema1] for schema-related markup that appears in XML instances.
wsse:	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd	This namespace is defined by the Web Services Security: SOAP Message Security 1.0 specification [WSS-SMS]. It is used by registry to secure soap message communication.

Prefix	XML Namespace	Comments
wsu:	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	This namespace is defined by the Web Services Security: SOAP Message Security 1.0 specification [WSS-SMS]. It is used by registry to secure soap message communication.

368

369 1.5 RepositoryItems and RegistryObjects

370 An ebXML Registry is capable of storing any type of electronic content such as XML documents, text
371 documents, images, sound and video. Instances of such content are referred to as a RepositoryItems.
372 RepositoryItems are stored in a content *repository* provided by the ebXML Registry.

373 In addition to the RepositoryItems, an ebXML Registry is also capable of storing standardized metadata
374 that MAY be used to further describe RepositoryItems. Instances of such metadata are referred to as a
375 RegistryObjects (or one of its sub-types, as described later in this document). RegistryObjects are
376 stored in the *registry* provided by the ebXML Registry.

377 To illustrate these concepts consider this familiar metaphor:

- 378 • An ebXML Registry is like your local library.
- 379 • The repository is like the bookshelves in the library.
- 380 • The repository items in the repository are like book on the bookshelves. The repository items can
381 contain any type of electronic content just like the books in the bookshelves can contain any type of
382 information.
- 383 • The registry is like the card catalog. It is organized for finding things quickly.
- 384 • A RegistryObject is like a card in the card catalog. All RegistryObjects conform to a standard just
385 like the cards in the card catalog conform to a standard.
- 386 • Every repository item MUST have a RegistryObject that describes it, just like every book must have
387 a card in the card catalog.

388 To summarize, ebXML Registry stores any type of content as RepositoryItems in a repository and
389 stores standardized metadata describing the content as RegistryObjects in a registry.

390 1.6 Canonical ClassificationSchemes

391 This specification uses several standard ClassificationSchemes as a mechanism to provides extensible
392 enumeration types. These ClassificationSchemes are referred to as *canonical ClassificationSchemes*.
393 The enumeration values within canonical ClassificationSchemes are defined using standard
394 ClassificationNodes that are referred to as *canonical ClassificationNodes*.

395 This section lists the canonical ClassificationSchemes that are required to be present in all ebXML
396 Registries. These Canonical ClassificationSchemes MAY be extended by adding additional
397 ClassificationNodes. However, a ClassificationNode defined normatively in the links below MUST NOT
398 be modified within a registry. In particular they MUST preserve their canonical id attributes in all
399 registries.

400 Note that all files listed in the Location column are relative to the following URL:

401 <http://www.oasis-open.org/committees/regrep/documents/3.0/canonical/>

402

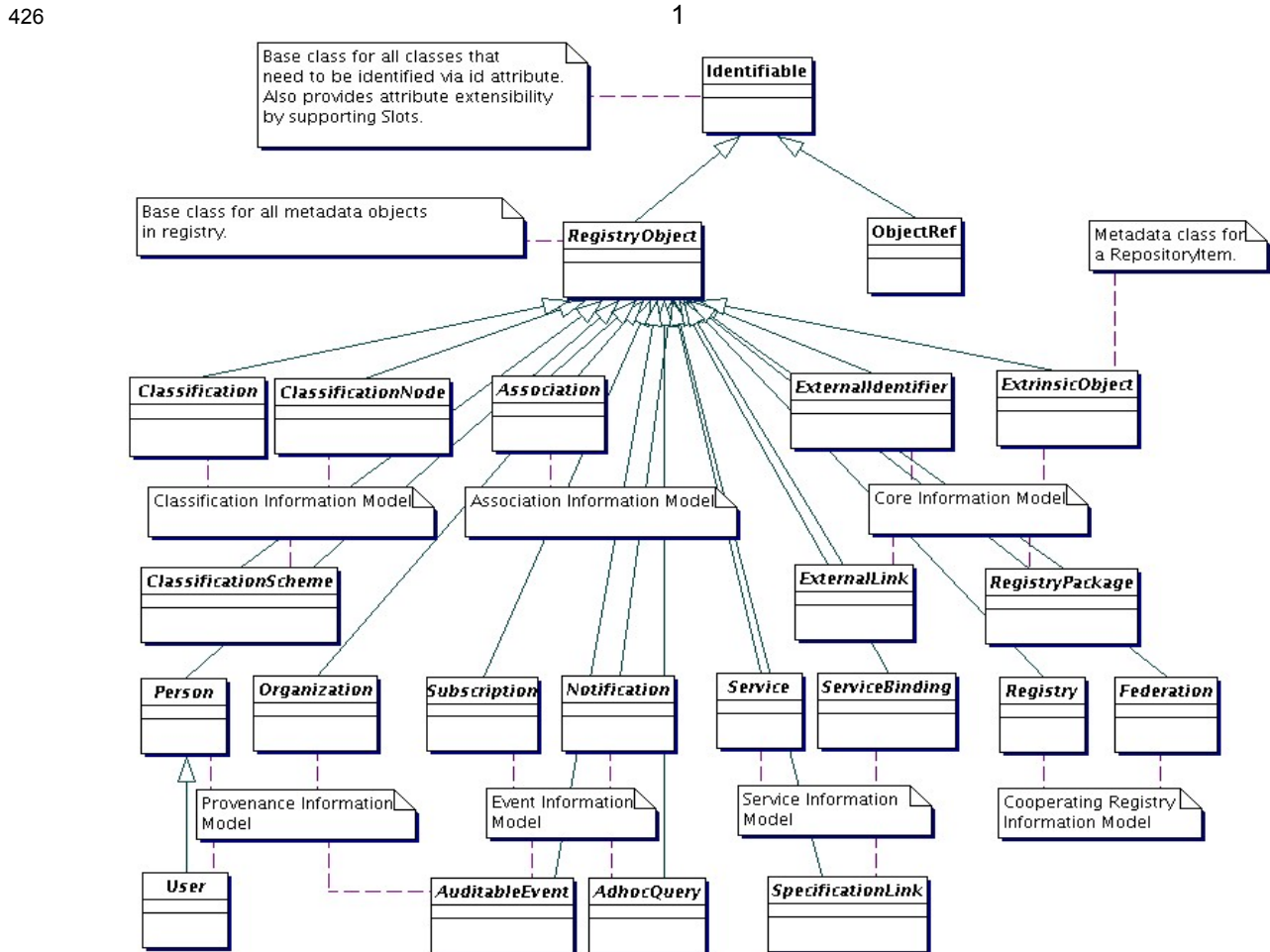
ClassificationScheme Name	Location / Description
AssociationType	SubmitObjectsRequest_AssociationTypeScheme.xml Defines the types of associations between RegistryObjects.

ClassificationScheme Name	Location / Description
ContentManagementService	SubmitObjectsRequest_CMSScheme.xml Defines the types of content management services.
DataType	SubmitObjectsRequest_DataTypeScheme Defines the data types for attributes in classes defined by this document.
DeletionScopeType	SubmitObjectsRequest_DeletionScopeTypeScheme.xml Defines the values for the deletionScope attribute in RemoveObjectsRequest protocol message.
EmailType	SubmitObjectsRequest_EmailTypeScheme.xml Defines the types of email addresses.
ErrorHandlingModel	SubmitObjectsRequest_ErrorHandlingModelScheme.xml Defines the types of error handling models for content management services.
ErrorSeverityType	SubmitObjectsRequest_ErrorSeverityTypeScheme.xml Defines the different error severity types encountered by registry during processing of protocol messages.
EventType	SubmitObjectsRequest_EventTypeScheme.xml Defines the types of events that can occur in a registry.
InvocationModel	SubmitObjectsRequest_InvocationModelScheme.xml Defines the different ways that a content management service may be invoked by the registry.
NodeType	SubmitObjectsRequest_NodeTypeScheme.xml Defines the different ways in which a ClassificationScheme may assign the value of the code attribute for its ClassificationNodes.
NotificationOptionType	SubmitObjectsRequest_NotificationOptionTypeScheme.xml Defines the different ways in which a client may wish to be notified by the registry of an event within a Subscription.
ObjectType	SubmitObjectsRequest_ObjectTypeScheme.xml Defines the different types of RegistryObjects a registry may support.
PhoneType	SubmitObjectsRequest_PhoneTypeScheme.xml Defines the types of telephone numbers.
QueryLanguage	SubmitObjectsRequest_QueryLangScheme Defines the query languages supported by a registry.
ResponseStatusType	SubmitObjectsRequest_ResponseStatusTypeScheme.xml Defines the different types of status for a RegistryResponse.
StatusType	SubmitObjectsRequest_StatusTypeScheme.xml Defines the different types of status for a RegistryObject.
SubjectGroup	SubmitObjectsRequest_SubjectGroupScheme Defines the groups that a User may belong to for access control purposes.

420 Detailed description of attributes of each class will be displayed in tabular form within the detailed
 421 description of each class.

422 **1.7.2.1 Class Identifiable**

423 The RegistryObject class and some other classes in RIM are derived from a class called *Identifiable*.
 424 This class provides the ability to identify objects by an id attribute and also provides attribute
 425 extensibility by allowing dynamic, instance-specific attributes called Slots.



427 **Figure 2: Information Model Inheritance View**

428

429 The RegistryObject sub-classes are shown in related groups as follows:

- 430
- 431 • Core Information Model: Defines core metadata classes in the model including the common base classes.
 - 432 • Association Information Model: Defines classes that enable RegistryObject instances to be associated with each other.
 - 433
 - 434 • Classification Information Model: Defines classes that enable RegistryObjects to be classified.
 - 435 • Provenance Information Model: Defines classes that enable the description of provenance or source information about a RegistryObject.
 - 436
 - 437 • Service Information Model: Defines classes that enable service description.
 - 438 • Event Information Model: Defines classes that enable the event subscription and notification feature defined in [eBRS].
 - 439

- 440 • Cooperating Registries Information Model: Defines classes that enable the cooperating registries
441 feature defined in [ebRS].
- 442 The remainder of this document will describe each of the above related group of classes in a dedicated
443 chapter named accordingly.

2 Core Information Model

This section covers the most commonly used information model classes defined by [ebRIM].

2.1 Attributes of Information Model Classes

Information model classes are defined in terms of their attributes. These attributes provide information on the state of the instances of these classes. Implementations of a registry typically map class attributes to attributes and elements in an XML store or columns in a relational store.

Since the model supports inheritance between classes, a class in the model inherits attributes from its super classes if any, in addition to defining its own specialized attributes.

The following is the description of the columns of many tables that summarize the attributes of a class:

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 Value	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

2.2 Data Types

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

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
Language	language	A string that identifies a local language. Values MUST be natural language identifiers as defined by [RFC 3066]	32 character
LongName	string	A long text string	256 characters
FreeFormText	string	A very long text string for free-form text	1024 characters
UUID	anyURI	A URI of the form urn:uuid:<uuid> where <uuid> MUST be a DCE 128 Bit Universally unique Id.	64 characters

ObjectRef	referenceURI	In XML Schema the referenceURI attribute value is a URI that references an ObjectRef within the XML document. If no such ObjectRef exists in the XML document then the value implicitly references a RegistryObject by the value of its id attribute within the registry.	64 characters
URI	anyURI	Used for URL and URN values	256 characters
URN	anyURI	Must be a valid URN	256 characters
Integer	integer	Used for integer values	4 bytes
DateTime	dateTime	Used for a timestamp value such as Date	
Set	sequence	As defined by OCL. An unordered Collection in which an object can occur only once.	
Bag	sequence	As defined by OCL. An unordered Collection in which the same object can occur multiple times.	
Sequence	sequence	As defined by OCL. An ordered Collection in which the same object can occur multiple times.	

458

459 2.3 Internationalization (I18N) Support

460 Some information model classes have String attributes that are I18N capable and may be localized into
 461 multiple native languages. Examples include the name and description attributes of the RegistryObject
 462 class in 2.5.

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

465 2.3.1 Class InternationalString

466 This class is used as a replacement for the String type whenever a String attribute needs to be I18N
 467 capable. An instance of the InternationalString class composes within it a Set of LocalizedString
 468 instances, where each String is specific to a particular locale.

469 2.3.1.1 Attribute Summary

470

Attribute	Data Type	Required	Default Value	Specified By	Mutable
localizedStrings	Set of LocalizedString	No		Client	Yes

471

472 2.3.1.2 Attribute localizedStrings

473 Each InternationalString instance MAY have a *localizedStrings* attribute that is a Set of zero or more
 474 LocalizedString instances.

475 2.3.2 Class LocalizedString

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

479 **2.3.2.1 Attribute Summary**

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

480

481 **2.3.2.2 Attribute lang**

482 Each LocalizedString instance MAY have a *lang* attribute that specifies the language used by that
483 LocalizedString.

484 **2.3.2.3 Attribute charset**

485 Each LocalizedString instance MAY have a *charset* attribute that specifies the name of the character
486 set used by that LocalizedString. The value of this attribute SHOULD be registered with IANA at:

487 <http://www.iana.org/assignments/character-sets>

488 **2.3.2.4 Attribute value**

489 Each LocalizedString instance MUST have a *value* attribute that specifies the string value used by that
490 LocalizedString.

491 **2.4 Class Identifiable**

492 The Identifiable class is the common super class for most classes in the information model. Information
493 model Classes whose instances have a unique identity are descendants of the Identifiable Class.

494 **2.4.1 Attribute Summary**

495

Attribute	Data Type	Required	Default Value	Specified By	Mutable
home	URI	No	Base URI of local registry	Client	Yes
id	URN	Yes		Client or registry	No
slots	Set of Slot	No		Client	Yes

496 **2.4.2 Attribute id**

497 Each Identifiable instance MUST have a unique identifier which is used to refer to that object.

498 Note that classes in the information model that do not inherit from Identifiable class do not require a
499 unique id. Examples include classes such as TelephoneNumber, PostalAddress, EmailAddress and
500 PersonName.

501 An Identifiable instance MUST have an id that MUST conform to the rules defined in section title
502 "Unique ID Generation" in [ebRS].

503 **2.4.3 Attribute home**

504 An Identifiable instance MAY have a *home* attribute. The *home* attribute, if present, MUST contain the
505 base URL to the home registry for the RegistryObject instance. The home URL MUST be specified for
506 instances of the Registry class that is defined later in this specification.

- 507 The base URL of a registry is:
- 508 • Used as the URL prefix for SOAP and HTTP interface bindings to the registry.
 - 509 • Used to qualify the id of an Identifiable instance by its registry within a federated registry
 - 510 environment.

511 2.4.4 Attribute slots

512 An Identifiable instance MAY have a Set of zero or more Slot instances that are composed within the
 513 Identifiable instance. These Slot instances serve as extensible attributes that MAY be defined for the
 514 Identifiable instance.

515 2.5 Class RegistryObject

516 **Super Classes:** [Identifiable](#)

517 The RegistryObject class extends the Identifiable class and serves as a common super class for most
 518 classes in the information model.

519 2.5.1 Attribute Summary

520

Attribute	Data Type	Required	Default Value	Specified By	Mutable
classifications	Set of Classification	No		Client	Yes
description	InternationalString	No		Client	Yes
externalIdentifiers	Set of ExternalIdentifier	No		Client	Yes
id	URN	Yes for READs, No for WRITEs.		Client or registry	No
name	InternationalString	No		Client	Yes
objectType	ObjectRef	Yes for READs, No for WRITEs.		Client or Registry	No
status	ObjectRef	Yes for READs, No for WRITEs.		Registry	Yes
versionInfo	VersionInfo	Yes for READs, No for WRITEs.		Registry	No

521 2.5.2 Composed Object

522 A RegistryObject instance MAY have instances of other RegistryObjects and other classes composed
 523 within it as defined in this specification. In such a relationship the composing object is referred to as the
 524 *Composite* object as defined in section 3.4 of [UML]. The composed object is referred to in this
 525 document and other ebXML Registry specification as *Composed* object. The relationship between the
 526 Composite and Composed object is referred to as a composition relationship as defined in section 3.4.8 of
 527 [UML].

528 *Composition* relationship implies that deletes and copies of the Composite object are cascaded to
 529 implicitly delete or copy the composed object. In comparison a UML Aggregation implies no such
 530 cascading.

531 The following classes defined by [RIM] are composed types and follow the rules defined by UML

532 composition relationships. The classes are listed in the order of their being defined in this document.
533 Note that abstract classes are not included in this list since an abstract class cannot have any
534 instances.

- 535 • InternationalString
- 536 • LocalizedString
- 537 • VersionInfo
- 538 • Slot
- 539 • ExternalIdentifier
- 540 • Classification
- 541 • PostalAddress
- 542 • TelephoneNumber
- 543 • EmailAddress
- 544 • PersonName
- 545 • ServiceBinding
- 546 • SpecificationLink
- 547 • QueryExpression
- 548 • NotifyAction

549

550 **2.5.3 Attribute classifications**

551 Each RegistryObject instance MAY have a Set of zero or more Classification instances that are
552 composed within the RegistryObject. These Classification instances classify the RegistryObject.

553 **2.5.4 Attribute description**

554 Each RegistryObject instance MAY have textual description in a human readable and user-friendly
555 form. This attribute is I18N capable and therefore of type InternationalString.

556 **2.5.5 Attribute externalIdentifier**

557 Each RegistryObject instance MAY have a Set of zero or more ExternalIdentifier instances that are
558 composed within the RegistryObject. These ExternalIdentifier instances serve as alternate identifiers
559 for the RegistryObject.

560 **2.5.6 Attribute lid**

561 Each RegistryObject instance MUST have a `lid` (Logical Id) attribute . The lid is used to refer to a
562 logical RegistryObject in a version independent manner. All versions of a RegistryObject MUST have
563 the same value for the lid attribute. Note that this is in contrast with the `id` attribute that MUST be
564 unique for each version of the same logical RegistryObject. The lid attribute MAY be specified by the
565 submitter when creating the original version of a RegistryObject. If the submitter assigns the lid
566 attribute, she must guarantee that it is a globally unique URN. A registry MUST honor a valid submitter-
567 supplied LID. If the submitter does not specify a LID then the registry MUST assign a LID and the value
568 of the LID attribute MUST be identical to the value of the `id` attribute of the first (originally created)
569 version of the logical RegistryObject.

570 Note that classes in the information model that do not inherit from RegistryObject class do not require a
571 lid. Examples include Entity classes such as TelephoneNumber, PostalAddress, EmailAddress and
572 PersonName.

573 **2.5.7 Attribute name**

574 Each RegistryObject instance MAY have a human readable name. The name does not need to be
575 unique with respect to other RegistryObject instances. This attribute is I18N capable and therefore of
576 type InternationalString.

577 **2.5.8 Attribute objectType**

578 Each RegistryObject instance has an *objectType* attribute. The value of the objectType attribute MUST
579 be a reference to a ClassificationNode in the canonical ObjectType ClassificationScheme. A Registry
580 MUST support the object types as defined by the ObjectType ClassificationScheme. The canonical
581 ObjectType ClassificationScheme may easily be extended by adding additional ClassificationNodes to
582 the canonical ObjectType ClassificationScheme.

583 The *objectType* for almost all objects in the information model matches the ClassificationNode that
584 corresponds to the name of their class. For example the *objectType* for a Classification is a reference to
585 the ClassificationNode with code "Classification" in the canonical ObjectType ClassificationScheme.
586 The only exception to this rule is that the *objectType* for an ExtrinsicObject or an ExternalLink instance
587 MAY be defined by the submitter and indicates the type of content associated with that object.

588 A registry MUST set the correct objectType on a RegistryObject when returning it as a response to a
589 client request. A client MAY set the objectType on a RegistryObject when submitting the object. A
590 client SHOULD set the objectType when the object is an ExternalLink or an ExtrinsicObject since
591 content pointed to or described by these types may be of arbitrary objectType.

592 **2.5.9 Attribute status**

593 Each RegistryObject instance MUST have a life cycle status indicator. The status is assigned by the
594 registry. A registry MUST set the correct status on a RegistryObject when returning it as a response to
595 a client request. A client SHOULD NOT set the status on a RegistryObject when submitting the object
596 as this is the responsibility of the registry. A registry MUST ignore the status on a RegistryObject when
597 it is set by the client during submission or update of the object.

598 The value of the status attribute MUST be a reference to a ClassificationNode in the canonical
599 StatusType ClassificationScheme. A Registry MUST support the status types as defined by the
600 StatusType ClassificationScheme. The canonical StatusType ClassificationScheme MAY easily be
601 extended by adding additional ClassificationNodes to the canonical StatusType ClassificationScheme.

602 **2.5.9.1 Pre-defined RegistryObject Status Types**

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

604
605

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

606

607 **2.5.10 Attribute versionInfo**

608 Each RegistryObject instance MAY have a *versionInfo* attribute. The value of the versionInfo attribute
609 MUST be of type VersionInfo. The versionInfo attribute provides information about the specific version
610 of a RegistryObject. The versionInfo attribute is set by the registry.

611 **2.6 Class VersionInfo**

612 VersionInfo class encapsulates information about the specific version of a RegistryObject.
613 The attributes of the VersionInfo class are described below.

614 **2.6.1 Attribute Summary**

Attribute	Data Type	Required	Default Value	Specified By	Mutable
versionName	String16	Yes	1.1	Registry	Yes
comment	LongName	No		Registry	Yes

615

616 **2.6.2 Attribute versionName**

617 Each VersionInfo instance MUST have versionName. This attribute defines the version name
618 identifying the VersionInfo for a specific RegistryObject version. The value for this attribute MUST be
619 automatically generated by the Registry implementation.

620 **2.6.3 Attribute comment**

621 Each VersionInfo instance MAY have comment. This attribute defines the comment associated with the
622 VersionInfo for a specific RegistryObject version. The value of the comment attribute is indirectly
623 provided by the client as the value of the comment attribute of the <rim:Request> object. The value for
624 this attribute MUST be set by the Registry implementation based upon the <rim:Request> comment
625 attribute value provided by the client if any.

626 **2.7 Class ObjectRef**

627 **Super Classes:** [Identifiable](#)

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

631 An instance of the ObjectRef class is used to reference a RegistryObject. A RegistryObject MAY be
632 referenced via an ObjectRef instance regardless of its location within a registry or that of the object
633 referring to it.

634 **2.7.1 Attribute Summary**

635

Attribute	Data Type	Required	Default Value	Specified By	Mutable
id	URN	Yes		Client	Yes
home	URI	No	Base URI of local registry	Client	Yes
createReplica	Boolean	No	false	Client	Yes

636

637 **2.7.2 Attribute *id***

638 Every ObjectRef instance MUST have an *id* attribute. The *id* attribute MUST contain the value of the *id*
639 attribute of the RegistryObject being referenced.

640 **2.7.3 Attribute *home***

641 Every ObjectRef instance MAY optionally have a *home* attribute specified. The *home* attribute if
642 present MUST contain the base URI to the home registry for the referenced RegistryObject. The base
643 URI to a registry is described by the REST interface as defined in [ebRS].

644 **2.7.3.1 Local Vs. Remote ObjectRefs**

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

647 If the *home* attribute is null then its default value is the base URI to the current registry. When the
648 *home* attribute is null or matches the base URI of the current registry, then the ObjectRef is referred to
649 as a local ObjectRef.

650 **2.7.4 Attribute *createReplica***

651 Every ObjectRef instance MAY have a *createReplica* attribute. The *createReplica* attribute is a client
652 supplied hint to the registry. When *createReplica* is true a registry SHOULD create a local replica for
653 the RegistryObject being referenced if it happens to be a remote ObjectRef.

654 **2.8 Class Slot**

655 Slot instances provide a dynamic way to add arbitrary attributes to RegistryObject instances. This
656 ability to add attributes dynamically to RegistryObject instances enables extensibility within the
657 information model.

658 A slot is composed of a name, a slotType and a Bag of values.

659 **2.8.1 Attribute Summary**

660

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

661

662 **2.8.2 Attribute *name***

663 Each Slot instance MUST have a name. The name is the primary means for identifying a Slot instance
664 within a RegistryObject. Consequently, the name of a Slot instance MUST be locally unique within the
665 RegistryObject instance.

666 **2.8.3 Attribute *slotType***

667 Each Slot instance MAY have a slotType that allows different slots to be grouped together. The
668 slotType
669 attribute MAY also be used to indicate the data type or value domain for the slot value(s).

670 **2.8.4 Attribute values**

671 A Slot instance MUST have a Sequence of values. The Sequence of values MAY be empty. Since a
672 Slot represent an extensible attribute whose value MAY be a Sequence, therefore a Slot is allowed to
673 have a Sequence of values rather than a single value.

674 **2.9 Class ExtrinsicObject**

675 **Super Classes:** RegistryObject

676 The ExtrinsicObject class is the primary metadata class for a RepositoryItem.

677 **2.9.1 Attribute Summary**

678

Attribute	Data Type	Required	Default Value	Specified By	Mutable
contentVersionInfo	VersionInfo	Yes for READs, No for WRITEs.		Registry	No
isOpaque	Boolean	No	false	Client	No
mimeType	LongName	No	application/octet-stream	Client	No

679

680 Note that attributes inherited from super classes are not shown in the table above.

681 **2.9.2 Attribute contentVersionInfo**

682 Each ExtrinsicObject instance MAY have a *contentVersionInfo* attribute. The value of the
683 *contentVersionInfo* attribute MUST be of type VersionInfo. The *contentVersionInfo* attribute provides
684 information about the specific version of the RepositoryItem associated with an ExtrinsicObject. The
685 *contentVersionInfo* attribute is set by the registry.

686 **2.9.3 Attribute isOpaque**

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

691 **2.9.4 Attribute mimeType**

692 Each ExtrinsicObject instance MAY have a mimeType attribute defined. The mimeType provides
693 information on the type of repository item catalogued by the ExtrinsicObject instance. The value of this
694 attribute SHOULD be a registered MIME media type at <http://www.iana.org/assignments/media-types>.

695 **2.10 Class RegistryPackage**

696 **Super Classes:** RegistryObject

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

699 **2.10.1 Attribute Summary**

700 The RegistryPackage class defines no new attributes other than those that are inherited from
701 RegistryObject super class. The inherited attributes are not shown here.

702 2.11 Class ExternalIdentifier

703 **Super Classes:** [RegistryObject](#)

704 ExternalIdentifier instances provide the additional identifier information to RegistryObject such as
705 DUNS number, Social Security Number, or an alias name of the organization. The attribute
706 *identificationScheme* is used to reference the identification scheme (e.g., "DUNS", "Social Security #"),
707 and the attribute *value* contains the actual information (e.g., the DUNS number, the social security
708 number). Each RegistryObject MAY contain 0 or more ExternalIdentifier instances.

709 2.11.1 Attribute Summary

710

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

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

712 2.11.2 Attribute identificationScheme

713 Each ExternalIdentifier instance MUST have an identificationScheme attribute that references a
714 ClassificationScheme. This ClassificationScheme defines the namespace within which an identifier is
715 defined using the value attribute for the RegistryObject referenced by the RegistryObject attribute.

716 2.11.3 Attribute registryObject

717 Each ExternalIdentifier instance MUST have a *registryObject* attribute that references the parent
718 RegistryObject for which this is an ExternalIdentifier.

719 2.11.4 Attribute value

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

722 2.12 Class ExternalLink

723 **Super Classes:** [RegistryObject](#)

724 ExternalLinks use URIs to associate content in the registry with content that MAY reside outside the
725 registry. For example, an organization submitting an XML Schema could use an ExternalLink to
726 associate the XML Schema with the organization's home page.

727 2.12.1 Attribute Summary

728

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

729

730 2.12.2 Attribute externalURI

731 Each ExternalLink instance MUST have an externalURI attribute defined. The externalURI attribute
732 provides a URI to the external resource pointed to by this ExternalLink instance. If the URI is a URL
733 then a registry MUST validate the URL to be resolvable at the time of submission before accepting an
734 ExternalLink submission to the registry.

3 Association Information Model

735

736 A RegistryObject instance MAY be associated with zero or more RegistryObject instances. The
737 information model defines the Association class, an instance of which MAY be used to associate any
738 two RegistryObject instances.

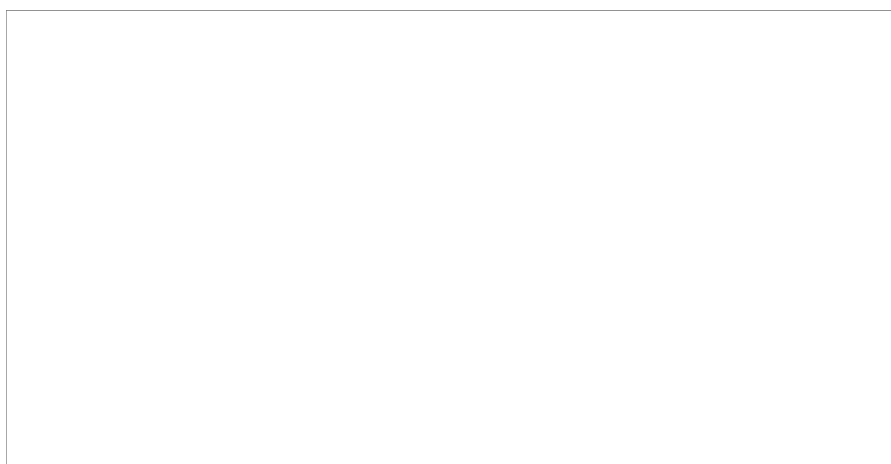
3.1 Example of an Association

739

740 One example of such an association is between two ClassificationScheme instances, where one
741 ClassificationScheme supersedes the other ClassificationScheme as shown in Figure 3. This may be
742 the case when a new version of a ClassificationScheme is submitted.

743 In Figure 3, we see how an Association is defined between a new version of the NAICS
744 ClassificationScheme and an older version of the NAICS ClassificationScheme.

745



746

747

Figure 3: Example of RegistryObject Association

3.2 Source and Target Objects

748

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

753 In the example in Figure 3, it is important to make the newer version of NAICS ClassificationScheme
754 be the sourceObject and the older version of NAICS be the targetObject because the associationType
755 implies that the sourceObject supersedes the targetObject (and not the other way around).

3.3 Association Types

756

757 Each Association MUST have an associationType attribute that identifies the type of that association.
758 The value of this attribute MUST be the id of a ClassificationNode under the canonical AssociationType
759 ClassificationScheme.

3.4 Intramural Association

760

761 A common use case for the Association class is when a User “u” creates an Association “a” between
762 two RegistryObjects “o1” and “o2” where Association “a” and RegistryObjects “o1” and “o2” are objects
763 that were created by the same User “u”. This is the simplest use case, where the Association is
764 between two objects that are owned by the same User that is defining the Association. Such
765 Associations are referred to as intramural Associations.

766 Figure 4 below, extends the previous example in Figure 3 for the intramural Association case.
767



768
769

Figure 4: Example of Intramural Association

770 **3.5 Extramural Association**

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

774 In this use case an Association is defined where either or both objects that are being associated are
775 owned by a User different from the User defining the Association. Such Associations are referred to as
776 extramural Associations.

777 Figure 5 below, extends the previous example in Figure 4 for the extramural Association case. Note
778 that it is possible for an extramural Association to have two distinct Users rather than three distinct
779 Users as shown in Figure 5. In such case, one of the two users owns two of the three objects involved
780 (Association, sourceObject and targetObject).

781

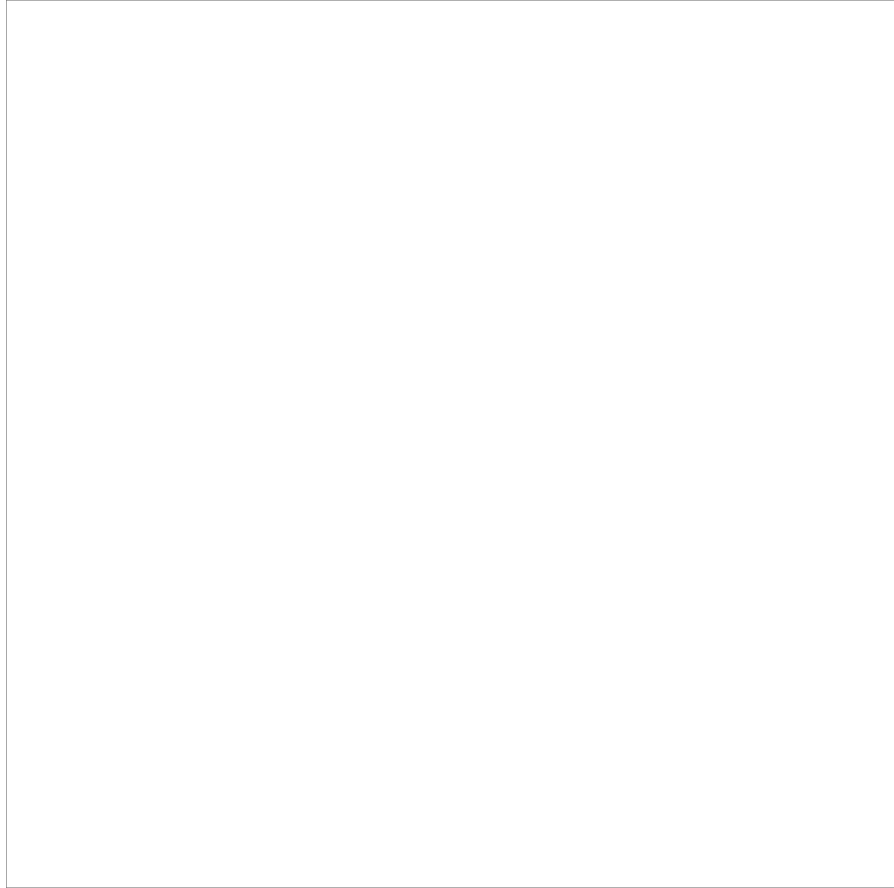


Figure 5: Example of Extramural Association

782
783

784 3.5.1 Controlling Extramural Associations

785 The owner of a RegistryObject MAY control who can create extramural associations to that
786 RegistryObject using custom access control policies using the reference access control feature
787 described in section 9.2.4.

788 3.6 Class Association

789 **Super Classes:** [RegistryObject](#)

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

792

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

794 3.6.1 Attribute Summary

795

Attribute	Data Type	Required	Default Value	Specified By	Mutable
associationType	ObjectRef	Yes		Client	No
sourceObject	ObjectRef	Yes		Client	No
targetObject	ObjectRef	Yes		Client	No

796

797 **3.6.2 Attribute associationType**

798 Each Association MUST have an *associationType* attribute that identifies the type of that association.
799 The value of the *associationType* attribute MUST be a reference to a ClassificationNode within the
800 canonical AssociationType ClassificationScheme. While the AssociationType scheme MAY easily be
801 extended, a Registry MUST support the canonical association types as defined by the canonical
802 AssociationType ClassificationScheme.

803 **3.6.3 Attribute sourceObject**

804 Each Association MUST have a *sourceObject* attribute that references the RegistryObject instance that
805 is the source of that Association.

806 **3.6.4 Attribute targetObject**

807 Each Association MUST have a *targetObject* attribute that references the RegistryObject instance that
808 is the target of that Association.

4 Classification Information Model

809

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

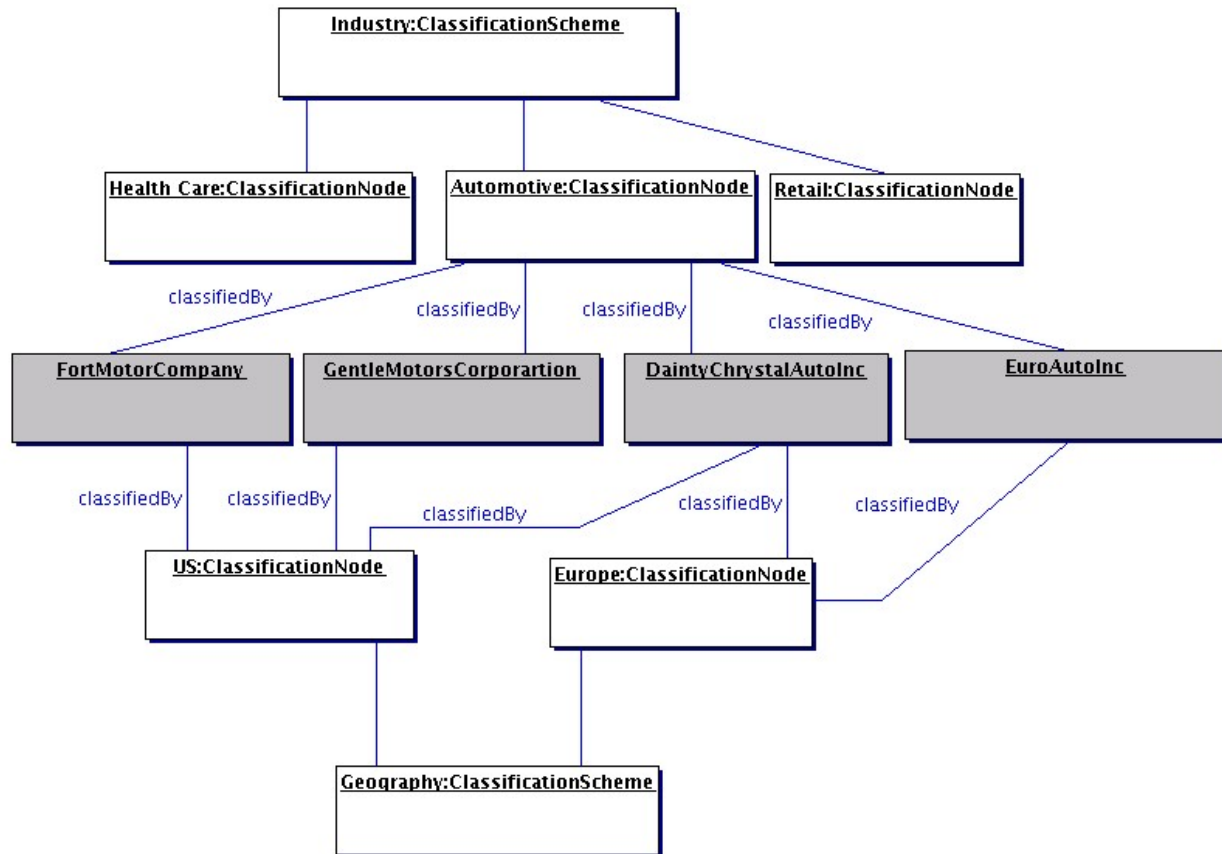
811 A RegistryObject MAY be classified in many ways. For example the RegistryObject for the same
812 Collaboration Protocol Profile (CPP) may be classified by its industry, by the products it sells and by its
813 geographical location.

814 A general ClassificationScheme can be viewed as a tree structure. In the example shown in Figure 6,
815 RegistryObject instances representing Collaboration Protocol Profiles are shown as shaded boxes.
816 Each Collaboration Protocol Profile represents an automobile manufacturer. Each Collaboration
817 Protocol Profile is classified by the ClassificationNode named "Automotive" under the
818 ClassificationScheme instance with name "Industry." Furthermore, the US Automobile manufacturers
819 are classified by the "US" ClassificationNode under the ClassificationScheme with name "Geography."
820 Similarly, a European automobile manufacturer is classified by the "Europe" ClassificationNode under
821 the ClassificationScheme with name "Geography."

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

824

825



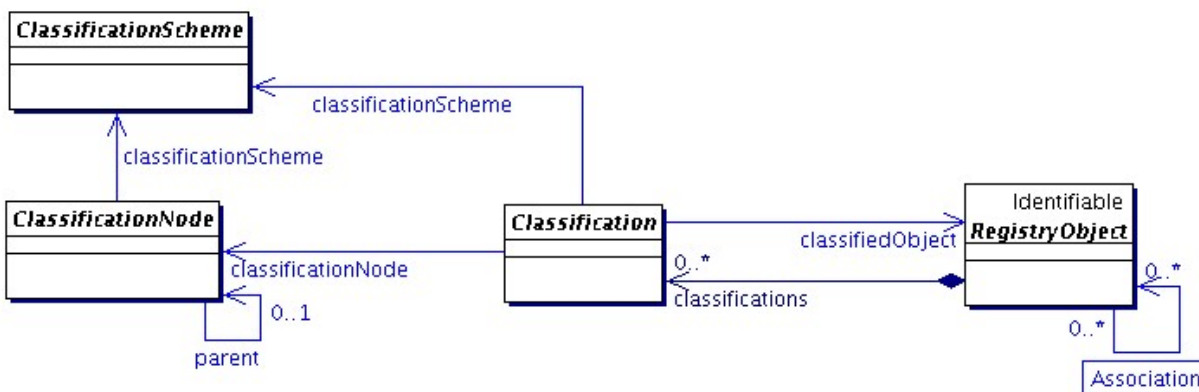
826

Figure 6: Example showing a Classification Tree

827 It is important to point out that the shaded nodes (FortMotorCompnay, GentleMotorsCorporation etc.)
828 are not part of the ClassificationScheme tree. The leaf nodes of the ClassificationScheme tree are
829 Health Care, Automotive, Retail, US and Europe. The shaded nodes are associated with the
830 ClassificationScheme tree via a Classification Instance that is not shown in the picture.

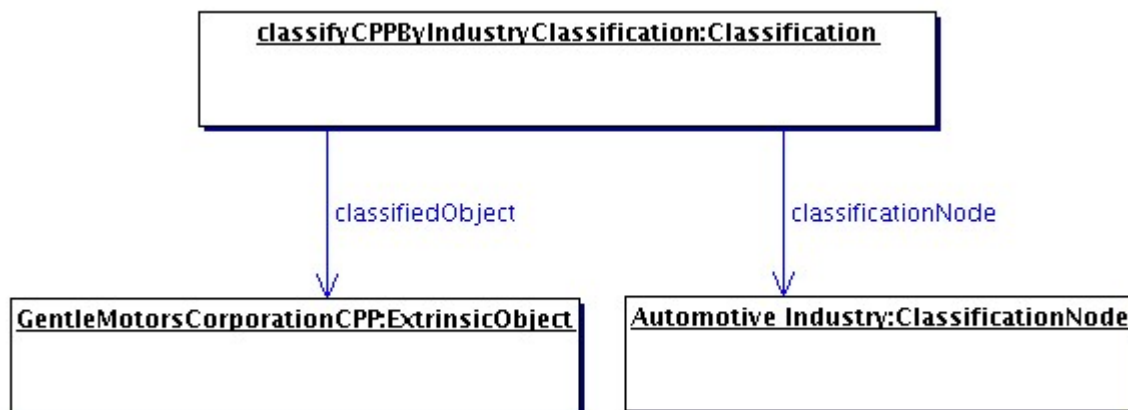
831 In order to support a general ClassificationScheme that can support single level as well as multi-level

832 Classifications, the information model defines the classes and relationships shown in Figure 7.
 833



834 **Figure 7: Information Model Classification View**

835
 836 A Classification is somewhat like a specialized form of an Association. Figure 8 shows an example of
 837 an ExtrinsicObject Instance for a Collaboration Protocol Profile (CPP) object that is classified by a
 838 ClassificationNode representing the Industry that it belongs to.
 839



840 **Figure 8: Classification Instance Diagram**

841

842 4.1 Class ClassificationScheme

843 **Super Classes:** [RegistryObject](#)

844 A ClassificationScheme instance describes a taxonomy. The taxonomy hierarchy may be defined
 845 internally to the registry by instances of ClassificationNode, or it may be defined externally to the
 846 Registry, in which case the structure and values of the taxonomy elements are not known to the
 847 Registry.

848 In the first case the classification scheme is said to be *internal* and in the second case the classification
 849 scheme is said to be *external*.

850 4.1.1 Attribute Summary

851

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

isInternal	Boolean	Yes		Client	No
nodeType	ObjectRef	Yes		Client	No

852 Note that attributes inherited by a ClassificationScheme class from the RegistryObject class are not
853 shown.

854 4.1.2 Attribute isInternal

855 When submitting a ClassificationScheme instance the submitter MUST declare whether the
856 ClassificationScheme instance represents an internal or an external taxonomy. This allows the registry
857 to validate the subsequent submissions of ClassificationNode and Classification instances in order to
858 maintain the type of ClassificationScheme consistent throughout its lifecycle.

859 4.1.3 Attribute nodeType

860 When submitting a ClassificationScheme instance the Submitting Organization MUST declare the
861 structure of taxonomy nodes within the ClassificationScheme via the nodeType attribute. The value of
862 the nodeType attribute MUST be a reference to a ClassificationNode within the canonical NodeType
863 ClassificationScheme. A Registry MUST support the node types as defined by the canonical NodeType
864 ClassificationScheme. The canonical NodeType ClassificationScheme MAY easily be extended by
865 adding additional ClassificationNodes to it.

866 The following canonical values are defined for the NodeType ClassificationScheme:

- 867 ○ **UniqueCode**: This value indicates that each node of the taxonomy has a unique code assigned
868 to it.
- 869 ○ **EmbeddedPath**: This value indicates that the unique code assigned to each node of the
870 taxonomy also encodes its path. This is the case in the NAICS taxonomy.
- 871 ○ **NonUniqueCode**: In some cases nodes are not unique, and it is necessary to use the full path
872 (from ClassificationScheme to the node of interest) in order to identify the node. For example,
873 in a geography taxonomy Moscow could be under both Russia and the USA, where there are
874 five cities of that name in different states.

875

876 4.2 Class ClassificationNode

877 **Super Classes:** [RegistryObject](#)

878 ClassificationNode instances are used to define tree structures where each node in the tree is a
879 ClassificationNode. Such ClassificationScheme trees are constructed with ClassificationNode instances
880 under a ClassificationScheme instance, and are used to define Classification schemes or ontologies.

881

882 4.2.1 Attribute Summary

883

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

884

885 4.2.2 Attribute parent

886 Each ClassificationNode MAY have a *parent* attribute. The parent attribute either references a parent
887 ClassificationNode or a ClassificationScheme instance in case of first level ClassificationNode
888 instances.

889 4.2.3 Attribute code

890 Each ClassificationNode MAY have a *code* attribute. The code attribute contains a code within a
891 standard coding scheme. The code attribute of a ClassificationNode MUST be unique with respect to
892 all sibling ClassificationNodes that are immediate children of the same parent ClassificationNode or
893 ClassificationScheme.

894 4.2.4 Attribute path

895 Each ClassificationNode MAY have a *path* attribute. A registry MUST set the path attribute for any
896 ClassificationNode that has a non-null code attribute value, when the ClassificationNode is retrieved
897 from the registry. The path attribute MUST be ignored by the registry when it is specified by the client
898 at the time the object is submitted to the registry. The path attribute contains the canonical path from
899 the root ClassificationScheme or ClassificationNode within the hierarchy of this ClassificationNode as
900 defined by the parent attribute. The path attribute of a ClassificationNode MUST be unique within a
901 registry. The path syntax is defined in 4.2.5.

902 4.2.5 Canonical Path Syntax

903 The path attribute of the ClassificationNode class contains an absolute path in a canonical
904 representation that uniquely identifies the path leading from the root ClassificationScheme or
905 ClassificationNode to that ClassificationNode.

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

907

```
908 canonicalPath ::= '/' rootSchemeOrNodeId nodePath  
909 nodePath      ::= '/' nodeCode  
910               | '/' nodeCode ( nodePath )?  
911
```

912 In the above grammar, rootSchemeOrNodeId is the id attribute of the root ClassificationScheme or
913 ClassificationNode instance, and nodeCode is defined by NCName production as defined by
914 <http://www.w3.org/TR/REC-xml-names/#NT-NCName>.

915

916 4.2.5.1 Example of Canonical Path Representation

917 The following canonical path represents what the *path* attribute would contain for the
918 ClassificationNode with code "United States" in the sample Geography scheme in section 4.2.5.2.

919

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

921 4.2.5.2 Sample Geography Scheme

922 Note that in the following examples, the *id* attributes have been chosen for ease of readability and are
923 therefore not valid id values.

924

```
925 <ClassificationScheme id='Geography-id' name="Geography"/>  
926  
927 <ClassificationNode id="NorthAmerica-id" parent="Geography-id"  
928 code="NorthAmerica" />  
929 <ClassificationNode id="UnitedStates-id" parent="NorthAmerica-id"  
930 code="UnitedStates" />  
931  
932 <ClassificationNode id="Asia-id" parent="Geography-id"  
933 code="Asia" />  
934 <ClassificationNode id="Japan-id" parent="Asia-id" code="Japan" />
```

935
936
937

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

4.3 Class Classification

Super Classes: RegistryObject

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

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

In Figure 6, Classification instances are not explicitly shown but are implied as associations between the RegistryObject instances (shaded leaf node) and the associated ClassificationNode.

4.3.1 Attribute Summary

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

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

4.3.2 Attribute classificationScheme

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

4.3.3 Attribute classificationNode

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

4.3.4 Attribute classifiedObject

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

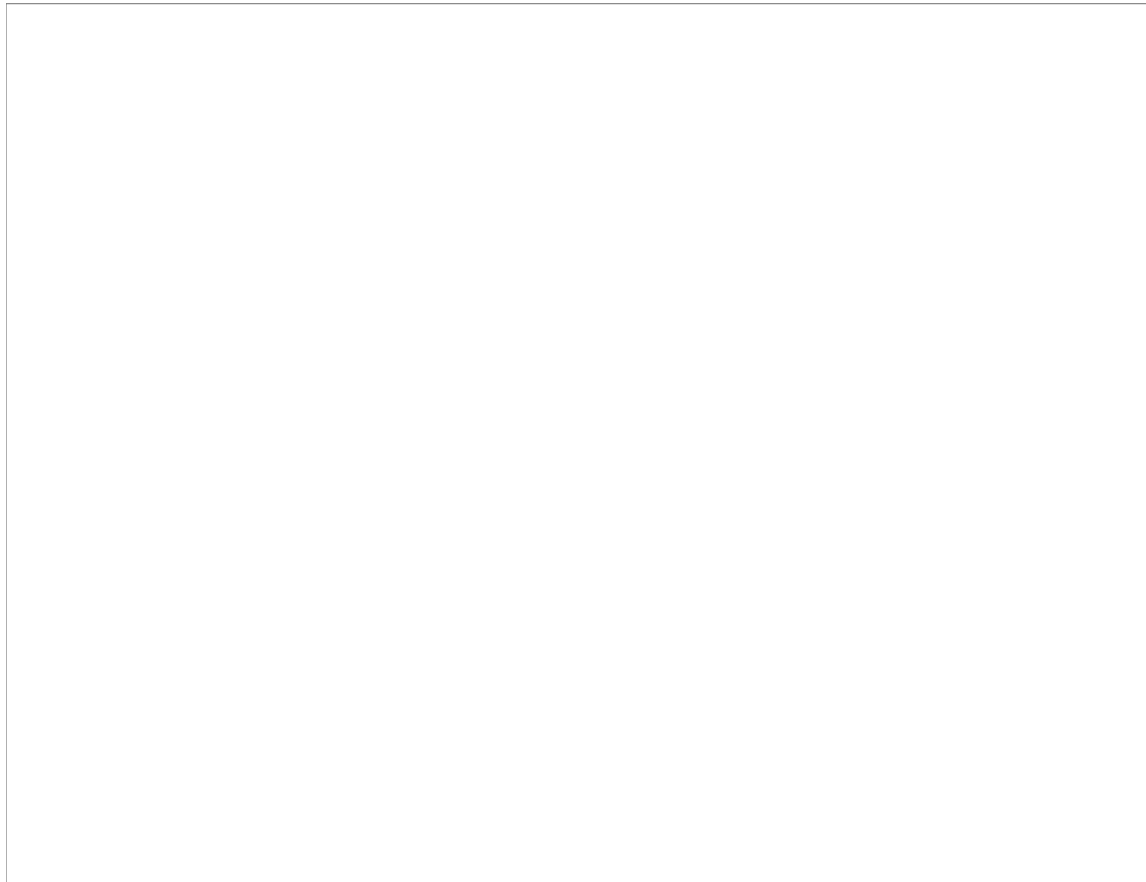
4.3.5 Attribute nodeRepresentation

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

968 **4.3.6 Context Sensitive Classification**

969 Consider the case depicted in Figure 9 where a Collaboration Protocol Profile for ACME Inc. is
970 classified by the “Japan” ClassificationNode under the “Geography” Classification scheme. In the
971 absence of the context for this Classification its meaning is ambiguous. Does it mean that ACME is
972 located in Japan, or does it mean that ACME ships products to Japan, or does it have some other
973 meaning? To address this ambiguity a Classification MAY optionally be associated with another
974 ClassificationNode (in this example named isLocatedIn) that provides the missing context for the
975 Classification. Another Collaboration Protocol Profile for MyParcelService MAY be classified by the
976 “Japan” ClassificationNode where this Classification is associated with a different ClassificationNode
977 (e.g., named shipsTo) to indicate a different context than the one used by ACME Inc.

978



979

980

Figure 9: Context Sensitive Classification

981

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

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

- 986 ○ A RegistryObject to be classified by defining an internal Classification that associates it with a
987 ClassificationNode in a ClassificationScheme.
- 988 ○ A RegistryObject to be classified by defining an external Classification that associates it with a
989 value in an external ClassificationScheme.
- 990 ○ A RegistryObject to be classified along multiple facets by having multiple Classifications that
991 associate it with multiple ClassificationNodes or value within a ClassificationScheme.

- A Classification defined for a RegistryObject to be qualified by the contexts in which it is being classified.

4.4 Example of Classification Schemes

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

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"	

Table 1: Sample Classification Schemes

1000
1001

5 Provenance Information Model

1002

1003 This chapter describes the classes that enable the description of
1004 the parties responsible for creating, publishing, or maintaining a RegistryObject or RepositoryItem.

1005 The term *provenance* in the English language implies the origin and history of ownership of things of
1006 value. When applied to the ebXML Registry, provenance implies information about the origin, history of
1007 ownership, custodianship, and other relationships between entities such as people and organizations
1008 and RegistryObjects.

1009 This includes information about:

- 1010 • The registered user that is the submitter of a RegistryObject or RepositoryItem.
- 1011 • The organization that is the submitter submitted the object on behalf of (Submitting Organization)
- 1012 • The organization that is responsible for the maintenance of the submitted object (Responsible
1013 Organization)
- 1014 • Any other persons that have some relationship with the submitted object

1015 5.1 Class Person

1016 **Super Classes:** [RegistryObject](#)

1017 Person instances represent persons or humans.

1018 5.1.1 Attribute Summary

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

1019

1020 5.1.2 Attribute addresses

1021 Each Person instance MAY have an attribute addresses that is a Set of PostalAddress instances. Each
1022 PostalAddress provides a postal address for that user. A Person SHOULD have at least one
1023 PostalAddress.

1024 5.1.3 Attribute emailAddresses

1025 Each Person instance MAY have an attribute emailAddresses that is a Set of EmailAddress instances.
1026 Each EmailAddress provides an email address for that person. A Person SHOULD have at least one
1027 EmailAddress.

1028 5.1.4 Attribute personName

1029 Each Person instance MAY have a *personName* attribute that provides the name for that user.

1030 5.1.5 Attribute telephoneNumbers

1031 Each Person instance MAY have a *telephoneNumbers* attribute that contains the Set of
1032 TelephoneNumber instances defined for that user. A Person SHOULD have at least one
1033 TelephoneNumber.

1034 **5.2 Class User**

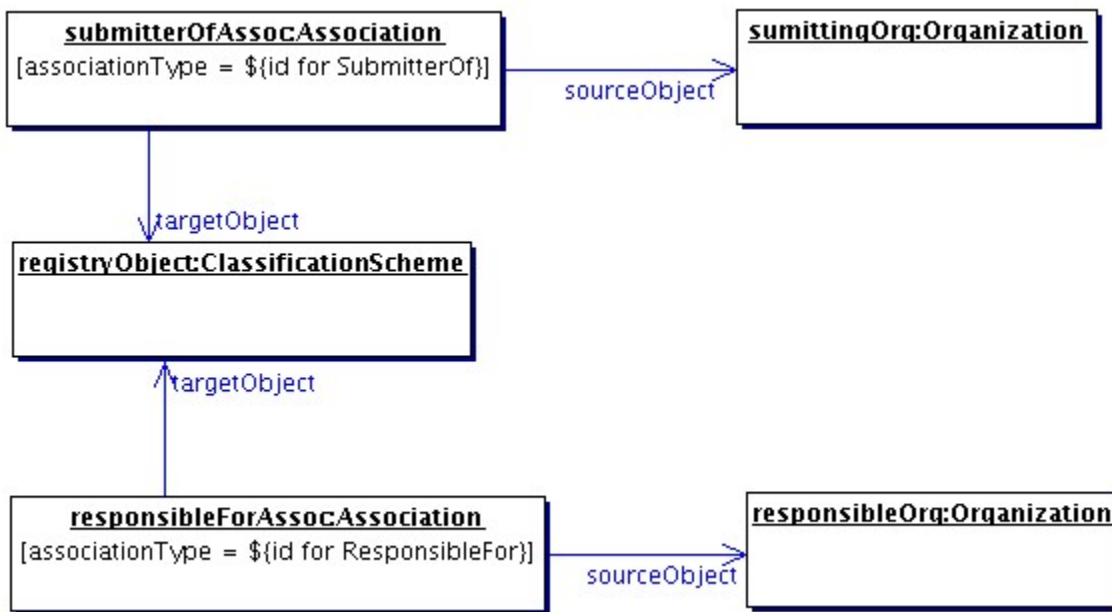
1035 **Super Classes:** [Person](#)

1036 User instances represent users that have registered with a registry. User instances are also used in an
 1037 AuditableEvent to keep track of the identity of the requestor that sent the request that generated the
 1038 AuditableEvent. User class is a sub-class of Person class that inherits all attributes of the Person class
 1039 and does not add any new attributes.

1040 **5.2.1 Associating Users With Organizations**

1041 A user MAY be affiliated with zero or more organizations. Each such affiliation is modeled in ebRIM
 1042 using an Association instance between a User instance and an Organization instance. The
 1043 associationType in such cases SHOULD be either the canonical "AffiliatedWith" associationType or a
 1044 ClassificationNode that is a descendant of the ClassificationNode representing the canonical
 1045 "AffiliatedWith" associationType.

1046



1047

Figure 10: User Affiliation With Organization Instance Diagram

1048

1049 **5.3 Class Organization**

1050 **Super Classes:** [RegistryObject](#)

1051 Organization instances provide information on organizations such as a Submitting Organization. Each
 1052 Organization instance MAY have a reference to a parent Organization.

1053 **5.3.1 Attribute Summary**

1054

Attribute	Data Type	Required	Default Value	Specified By	Mutable
addresses	Set of PostalAddress	No		Client	Yes
emailAddresses	Set of EmailAddress	No		Client	Yes
parent	ObjectRef	No		Client	Yes
primaryContact	ObjectRef	No		Client	No

telephoneNumbers	Set of TelephoneNumber	No		Client	Yes
------------------	------------------------	----	--	--------	-----

1055

1056 **5.3.2 Attribute addresses**

1057 Each Organization instance MAY have an *addresses* attribute that is a Set of PostalAddress instances.
 1058 Each PostalAddress provides a postal address for that organization. An Organization SHOULD have at
 1059 least one PostalAddress.

1060 **5.3.3 Attribute emailAddresses**

1061 Each Organization instance MAY have an attribute *emailAddresses* that is a Set of EmailAddress
 1062 instances. Each EmailAddress provides an email address for that Organization. An Organization
 1063 SHOULD have at least one EmailAddress.

1064 **5.3.4 Attribute parent**

1065 Each Organization instance MAY have a *parent* attribute that references the parent Organization
 1066 instance, if any, for that organization.

1067 **5.3.5 Attribute primaryContact**

1068 Each Organization instance SHOULD have a *primaryContact* attribute that references the Person
 1069 instance for the person that is the primary contact for that organization.

1070 **5.3.6 Attribute telephoneNumbers**

1071 Each Organization instance MUST have a *telephoneNumbers* attribute that contains the Set of
 1072 TelephoneNumber instances defined for that organization. An Organization SHOULD have at least one
 1073 telephone number.

1074 **5.4 Associating Organizations With RegistryObjects**

1075 An organization MAY be associated with zero or more RegistryObject instances. Each such association
 1076 is modeled in ebRIM using an Association instance between an Organization instance and a
 1077 RegistryObject instance. The associationType in such cases MAY be (but is not restricted to) either the
 1078 canonical "SubmitterOf" associationType or the canonical "ResponsibleFor" associationType. The
 1079 "SubmitterOf" associationType indicates the organization that submitted the RegistryObject (via a
 1080 User). The "ResponsibleFor" associationType indicates the organization that is designated as the
 1081 organization responsible for the ongoing maintenance of the RegistryObject.

1082 Associations between Organizations and RegistryObjects do not entitle organizations to any special
 1083 privileges with respect to the RegistryObject. Such privileges are defined by the Access Control
 1084 Policies defined for the RegistryObject as described in chapter 9.



1085
1086 **Figure 11: Organization to RegistryObject Association Instance Diagram**
1087

1088 **5.5 Class PostalAddress**

1089 PostalAddress defines attributes of a postal address.

1090 **5.5.1 Attribute Summary**
1091

Attribute	Data Type	Required	Default Value	Specified By	Mutable
city	ShortName	No		Client	Yes
country	ShortName	No		Client	Yes
postalCode	ShortName	No		Client	Yes
slots	Set of Slot	No		Client	Yes
stateOrProvince	ShortName	No		Client	Yes
street	ShortName	No		Client	Yes
streetNumber	String32	No		Client	Yes

1092
1093 **5.5.2 Attribute city**

1094 Each PostalAddress MAY have a *city* attribute identifying the city for that address.

1095 **5.5.3 Attribute country**

1096 Each PostalAddress MAY have a *country* attribute identifying the country for that address.

1097 **5.5.4 Attribute postalCode**

1098 Each PostalAddress MAY have a *postalCode* attribute identifying the postal code (e.g., zip code) for
1099 that address.

1100 **5.5.5 Attribute stateOrProvince**

1101 Each PostalAddress MAY have a *stateOrProvince* attribute identifying the state, province or region for
1102 that address.

1103 **5.5.6 Attribute street**

1104 Each PostalAddress MAY have a *street* attribute identifying the street name for that address.

1105 **5.5.7 Attribute streetNumber**

1106 Each PostalAddress MAY have a *streetNumber* attribute identifying the street number (e.g., 65) for the
1107 street address.

1108 **5.6 Class TelephoneNumber**

1109 This class defines attributes of a telephone number.

1110 **5.6.1 Attribute Summary**

1111

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

1112

1113 **5.6.2 Attribute areaCode**

1114 Each TelephoneNumber instance MAY have an *areaCode* attribute that provides the area code for that
1115 telephone number.

1116 **5.6.3 Attribute countryCode**

1117 Each TelephoneNumber instance MAY have a *countryCode* attribute that provides the country code for
1118 that telephone number.

1119 **5.6.4 Attribute extension**

1120 Each TelephoneNumber instance MAY have an *extension* attribute that provides the extension
1121 number, if any, for that telephone number.

1122 **5.6.5 Attribute number**

1123 Each TelephoneNumber instance MAY have a *number* attribute that provides the local number (without
1124 area code, country code and extension) for that telephone number.

1125 **5.6.6 Attribute phoneType**

1126 Each TelephoneNumber instance MAY have a *phoneType* attribute that provides the type for the
1127 TelephoneNumber. The value of the phoneType attribute MUST be a reference to a ClassificationNode
1128 in the canonical PhoneType ClassificationScheme.

1129 **5.7 Class EmailAddress**

1130 This class defines attributes of an email address.

1131 **5.7.1 Attribute Summary**

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

1132 **5.7.2 Attribute address**

1133 Each EmailAddress instance MUST have an *address* attribute that provides the actual email address.

1134 **5.7.3 Attribute type**

1135 Each EmailAddress instance MAY have a *type* attribute that provides the type for that email address.

1136 The value of the type attribute MUST be a reference to a ClassificationNode in the canonical

1137 EmailType ClassificationScheme as referenced in appendix .

1138 **5.8 Class PersonName**

1139 This class defines attributes for a person's name.

1140 **5.8.1 Attribute Summary**

1141

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

1142

1143 **5.8.2 Attribute firstName**

1144 Each PersonName SHOULD have a *firstName* attribute that is the first name of the person.

1145 **5.8.3 Attribute lastName**

1146 Each PersonName SHOULD have a *lastName* attribute that is the last name of the person.

1147 **5.8.4 Attribute middleName**

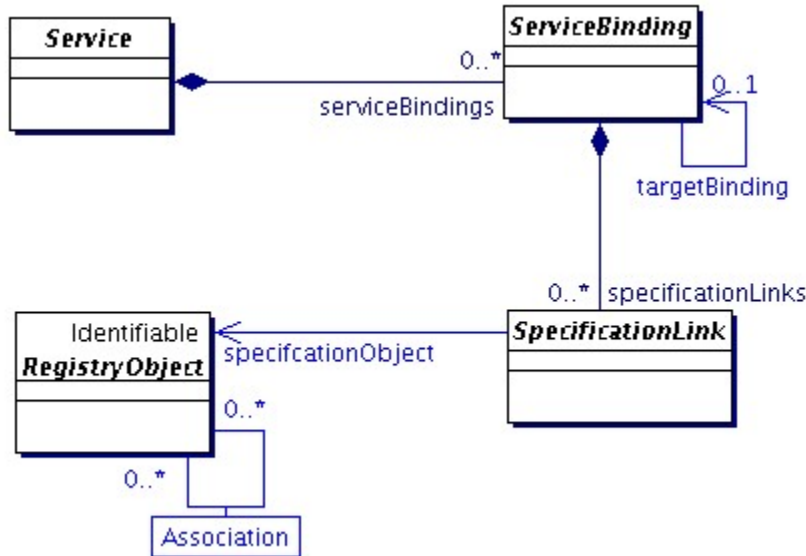
1148 Each PersonName SHOULD have a *middleName* attribute that is the middle name of the person.

1149

6 Service Information Model

1150 This chapter describes the classes in the information model that support the registration of service
 1151 descriptions. The service information model is flexible and supports the registration of web services as
 1152 well as other types of services.

1153



1154

Figure 12: Service Information Model

6.1 Class Service

1156 **Super Classes:** [RegistryObject](#)

1157 Service instances describe services, such as web services.

6.1.1 Attribute Summary

1159

Attribute	Data Type	Required	Default Value	Specified By	Mutable
serviceBindings	Set of ServiceBinding	Yes, Set may be empty		Client	Yes

1160

6.1.2 Attribute serviceBindings

1162 A Service MAY have a *serviceBindings* attribute that defines the service bindings that provide access to
 1163 that Service.

6.2 Class ServiceBinding

1165 **Super Classes:** [RegistryObject](#)

1166 ServiceBinding instances are RegistryObjects that represent technical information on a specific way to
 1167 access a Service instance. An example is where a ServiceBinding is defined for each protocol that may
 1168 be used to access the service.

1169 **6.2.1 Attribute Summary**

1170

Attribute	Data Type	Required	Default Value	Specified By	Mutable
accessURI	URI	No		Client	Yes
service	ObjectRef	Yes		Client	No
specificationLinks	Set of SpecificationLink	Yes, Set may be empty		Client	Yes
targetBinding	ObjectRef	No		Client	Yes

1171 **6.2.2 Attribute accessURI**

1172 A ServiceBinding MAY have an *accessURI* attribute that defines the URI to access that ServiceBinding.
 1173 This attribute is ignored if a *targetBinding* attribute is specified for the ServiceBinding. If the URI is a
 1174 URL then a registry MUST validate the URL to be resolvable at the time of submission before accepting
 1175 a ServiceBinding submission to the registry.

1176 **6.2.3 Attribute service**

1177 A ServiceBinding MUST have a *service* attribute whose value MUST be the id of its parent Service.

1178 **6.2.4 Attribute specificationLinks**

1179 A ServiceBinding MAY have a *specificationLinks* attribute defined that is a Set of references to
 1180 SpecificationLink instances. Each SpecificationLink instance links the ServiceBinding to a particular
 1181 technical specification that MAY be used to access the Service for the ServiceBinding.

1182 **6.2.5 Attribute targetBinding**

1183 A ServiceBinding MAY have a *targetBinding* attribute defined that references another ServiceBinding. A
 1184 targetBinding MAY be specified when a service is being redirected to another service. This allows the
 1185 rehosting of a service by another service provider.

1186 **6.3 Class SpecificationLink**

1187 **Super Classes:** [RegistryObject](#)

1188 A SpecificationLink provides the linkage between a ServiceBinding and one of its technical
 1189 specifications that describes how to use the service using the ServiceBinding. For example, a
 1190 ServiceBinding MAY have SpecificationLink instances that describe how to access the service using a
 1191 technical specification such as a WSDL document or a CORBA IDL document.

1192 **6.3.1 Attribute Summary**

1193

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

1194

1195 **6.3.2 Attribute serviceBinding**

1196 A SpecificationLink instance MUST have a *serviceBinding* attribute that provides a reference to its
1197 parent ServiceBinding instances. Its value MUST be the id of the parent ServiceBinding object.

1198 **6.3.3 Attribute specificationObject**

1199 A SpecificationLink instance MUST have a *specificationObject* attribute that provides a reference to a
1200 RegistryObject instance that provides a technical specification for the parent ServiceBinding. Typically,
1201 this is an ExtrinsicObject instance representing the technical specification (e.g., a WSDL document). It
1202 may also be an ExternalLink object in case the technical specification is a resource that is external to
1203 the registry.

1204 **6.3.4 Attribute usageDescription**

1205 A SpecificationLink instance MAY have a *usageDescription* attribute that provides a textual description
1206 of how to use the optional usageParameters attribute described next. The usageDescription is of type
1207 InternationalString, thus allowing the description to be in multiple languages.

1208 **6.3.5 Attribute usageParameters**

1209 A SpecificationLink instance MAY have a *usageParameters* attribute that provides a Bag of Strings
1210 representing the instance specific parameters needed to use the technical specification (e.g., a WSDL
1211 document) specified by this SpecificationLink object.

1212

7 Event Information Model

1213

1214 This chapter defines the information model classes that support the registry Event Notification feature.
 1215 These classes include AuditableEvent, Subscription, Selector and Action. They constitute the
 1216 foundation of the Event Notification information model.

1217 Figure 13 shows how a Subscription may be defined that uses a pre-configured AdhocQuery instance
 1218 as a selector to select the AuditableEvents of interest to the subscriber and one or more Actions to
 1219 deliver the selected events to the subscriber. The Action may deliver the events by using its endPoint
 1220 attribute to invoke a registered ServiceBinding to a registered Service or by sending the events to an
 1221 email address.

1222

1223

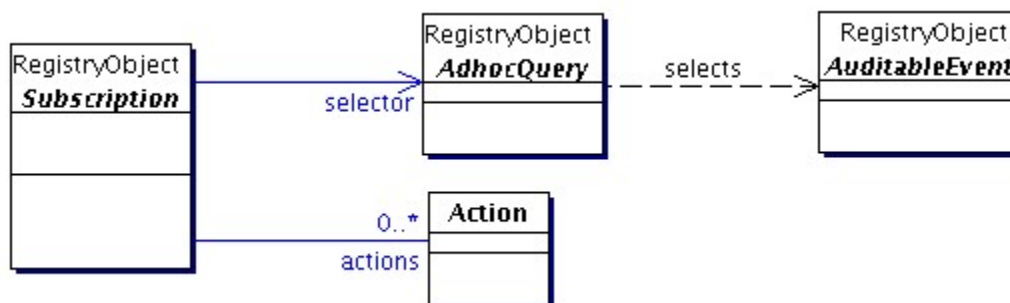


Figure 13: Event Information Model

1224

1225

7.1 Class AuditableEvent

1226

1227 **Super Classes:** [RegistryObject](#)

1227

1228 AuditableEvent instances provide a long-term record of events that effected a change in a
 1229 RegistryObject. A RegistryObject is associated with an ordered Set of AuditableEvent instances that
 1230 provide a complete audit trail for that RegistryObject.

1231 AuditableEvents are usually a result of a client-initiated request. AuditableEvent instances are
 1232 generated by the Registry Service to log such Events.

1233 Often such events effect a change in the life cycle of a RegistryObject. For example a client request
 1234 could Create, Update, Deprecate or Delete a RegistryObject. An AuditableEvent is typically created
 1235 when a request creates or alters the content or ownership of a RegistryObject. Read-only requests
 1236 typically do not generate an AuditableEvent.

7.1.1 Attribute Summary

1237

1238

Attribute	Data Type	Required	Default Value	Specified By	Mutable
eventType	ObjectRef	Yes		Registry	No
affectedObjects	Set of ObjectRef	Yes		Registry	No
requestId	URI	Yes		Registry	No
timestamp	dateTime	Yes		Registry	No
user	ObjectRef	Yes		Registry	No

1239

1240 7.1.2 Attribute eventType

1241 Each AuditableEvent MUST have an *eventType* attribute which identifies the type of event recorded by
1242 the AuditableEvent. The value of the eventType attribute MUST be a reference to a ClassificationNode
1243 in the canonical EventType ClassificationScheme. A Registry MUST support the event types as defined
1244 by the canonical EventType ClassificationScheme. The canonical EventType ClassificationScheme
1245 MAY easily be extended by adding additional ClassificationNodes to the canonical EventType
1246 ClassificationScheme.

1247 7.1.2.1 Pre-defined Auditable Event Types

1248 The following table lists pre-defined auditable event types. A Registry MUST support the event types
1249 listed below. A Registry MAY support additional event types as long as they are ClassificationNodes
1250 within the canonical EventType ClassificationScheme.

1251

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.
Undeprecated	An Event that marks the undeprecation of a RegistryObject.
Updated	An Event that that marks the updating of a RegistryObject.
Versioned	An Event that that marks the creation of a new version of a RegistryObject.

1252

1253 7.1.3 Attribute affectedObjects

1254 Each AuditableEvent MUST have an *affectedObjects* attribute that identifies the Set of RegistryObjects
1255 instances that were affected by this event.

1256 7.1.4 Attribute requestId

1257 Each AuditableEvent MUST have a *requestId* attribute that identifies the client request instance that
1258 affected this event.

1259 7.1.5 Attribute timestamp

1260 Each AuditableEvent MUST have a *timestamp* attribute that records the date and time that this event
1261 occurred.

1262 7.1.6 Attribute user

1263 Each AuditableEvent MUST have a *user* attribute that identifies the User that sent the request that
1264 generated this event affecting the RegistryObject instance.

1265 7.2 Class Subscription

1266 **Super Classes:** [RegistryObject](#)

1267 Subscription instances are RegistryObjects that define a User's interest in certain types of
1268 AuditableEvents. A User MAY create a subscription with a registry if he or she wishes to receive

1269 notification for a specific type of event.

1270 7.2.1 Attribute Summary

1271

Attribute	Data Type	Required	Default Value	Specified By	Mutable
actions	Set of Action	Yes, may be empty		Client	Yes
endTime	dateTime	No		Client	Yes
notificationInterval	duration	No	P1D (1 day)	Client	No
selector	ObjectRef	Yes		Client	No
startTime	dateTime	No	Current time	Client	Yes

1272

1273 7.2.2 Attribute actions

1274 A Subscription instance **MUST** have an *actions* attribute that is a Set of zero or more Action instances.
1275 An Action instance describes what action the registry must take when an event matching the
1276 Subscription transpires. The Action class is described in section 7.5.

1277 7.2.3 Attribute endTime

1278 This attribute denotes the time after which the subscription expires and is no longer active. If this
1279 attribute is missing the subscription never expires.

1280 7.2.4 Attribute notificationInterval

1281 This attribute denotes the duration that a registry **MUST** wait between delivering successive
1282 notifications to the client. The client specifies this attribute in order to control the frequency of
1283 notification communication between registry and client.

1284 7.2.5 Attribute selector

1285 This attribute defines the selection criteria that determine which events match this Subscription and are
1286 of interest to the User. The *selector* attribute references a pre-defined query that is stored in the
1287 registry as an instance of the AdhocQuery class. This AdhocQuery instance specifies or “selects”
1288 events that are of interest to the subscriber. The AdhocQueryClass is described in section 7.3.

1289 7.2.5.1 Specifying Selector Query Parameters

1290 The selector query **MAY** be configured as a parameterized stored query as defined by [ebRS]. A
1291 Subscription **MUST** specify the parameter values for stored parameterized queries as Slots as defined
1292 in section title “Specifying Query Invocation Parameters” in [ebRS]. These parameter value Slots if
1293 specified **MUST** be specified on the Subscription object.

1294 7.2.6 Attribute startTime

1295 This attribute denotes the time at which the subscription becomes active. If this attribute is missing
1296 subscription starts immediately.

1297 7.3 Class AdhocQuery

1298 **Super Classes:** [RegistryObject](#)

1299 The AdhocQuery class is a container for an ad hoc query expressed in a query syntax that is supported

1300 by an ebXML Registry. Instances of this class MAY be used for discovery of RegistryObjects within the
 1301 registry. Instances of AdhocQuery MAY be stored in the registry like other RegistryObjects. Such stored
 1302 AdhocQuery instances are similar in purpose to the concept of stored procedures in relational
 1303 databases.

1304 7.3.1 Attribute Summary

1305

Attribute	Data Type	Required	Default Value	Specified By	Mutable
queryExpression	QueryExpression	Required when defining a new AdhocQuery. Not required when invoking a stored query.		Client	No

1306

1307 7.3.2 Attribute queryExpression

1308 Each AdhocQuery instance MAY have a *queryExpression* attribute that contains the query expression
 1309 for the AdhocQuery depending upon the use case as follows. When an AdhocQuery is submitted to the
 1310 registry it MUST contain a queryExpression. When a stored AdhocQuery is included in an
 1311 AdhocQueryRequest to invoke a stored query as defined by the stored query feature defined in [ebRS]
 1312 it SHOULD NOT contain a queryExpression.

1313 7.4 Class QueryExpression

1314 The QueryExpression class is an extensible wrapper that can contain a query expression in any
 1315 supported query syntax such as SQL or Filter Query syntax.

1316 7.4.1 Attribute Summary

1317

Attribute	Data Type	Required	Default Value	Specified By	Mutable
queryLanguage	ObjectRef	Required		Client	No
<any>	anyType	Required		Client	No

1318 7.4.2 Attribute queryLanguage

1319 The queryLanguage attribute specifies the query language that the query expression conforms to. The
 1320 value of this attribute MUST be a reference to a ClassificationNode within the canonical
 1321 QueryLanguage ClassificationScheme. A Registry MUST support the query languages as defined by
 1322 the canonical QueryLanguage ClassificationScheme. The canonical QueryLanguage
 1323 ClassificationScheme MAY easily be extended by adding additional ClassificationNodes to it to allow a
 1324 registry to support additional query language syntaxes.

1325 7.4.3 Attribute <any>

1326 This attribute is extensible and therefor MAY be of any type depending upon the queryLanguage
 1327 specified. For SQL queryLanguage it MUST be an SQL query string. For Filter query it MUST be a
 1328 FilterQueryType defined by [RR-QUERY-XSD].

1329 7.5 Class Action

1330 The Action class is an abstract super class that specifies what the registry must do when an event
 1331 matching the action's Subscription transpires. A registry uses Actions within a Subscription to

1332 asynchronously deliver event Notifications to the subscriber.
 1333 If no Actions are defined within the Subscription it implies that the user does not wish to be notified
 1334 asynchronously by the registry and instead intends to periodically poll the registry and pull the pending
 1335 Notifications.
 1336 This class does not currently define any attributes.

1337 7.6 Class NotifyAction

1338 **Super Classes:** [Action](#)

1339 The NotifyAction class is a sub-class of Action class. An instance of NotifyAction represents an Action
 1340 that the registry MUST perform in order to notify the subscriber of a Subscription of the events of
 1341 interest to that subscriber.

1342 7.6.1 Attribute Summary

1343

Attribute	Data Type	Required	Default Value	Specified By	Mutable
endPoint	URI	YES		Client	
notificationOption	ObjectRef	No	Reference to ObjectRefs ClassificationNode	Client	Yes

1344

1345 7.6.2 Attribute endPoint

1346 This attribute specifies a URI that identifies a service end point that MAY be used by the registry to
 1347 deliver notifications. Currently this attribute can either be a "mailto" URI (e.g.
 1348 mailto:someone@acme.com) or a "urn:uuid" URI.

1349 If endpoint is a "mailto" URI then the registry MUST use the specified email address to deliver the
 1350 notification via email. Email configuration parameters such as the "from" email address and SMTP
 1351 server configuration MAY be specified in a registry specific manner.

1352 If endpoint is a "urn:uuid" URI then it MUST be a reference to a ServiceBinding object to a Service that
 1353 implements the RegistryClient interface as defined by [ebRS]. In this case the registry MUST deliver
 1354 the notification by web service invocation as defined by the ServiceBinding object.

1355 7.6.3 Attribute notificationOption

1356 This attribute controls the specific type of event notification content desired by the subscriber. It is used
 1357 by the subscriber to control the granularity of event notification content communicated by the registry to
 1358 the subscriber. The value of the notificationOption attribute MUST be a reference to a
 1359 ClassificationNode within the canonical NotificationOptionType ClassificationScheme. A Registry MUST
 1360 support the notificationOption types as defined by the NotificationOptionType ClassificationScheme.
 1361 The canonical NotificationOptionType ClassificationScheme MAY easily be extended by adding
 1362 additional ClassificationNodes to it.

1363 7.6.3.1 Pre-defined notificationOption Values

1364 The following canonical values are defined for the NotificationOptionType ClassificationScheme:

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.

1365

1366 7.7 Class Notification

1367 **Super Classes:** [RegistryObject](#)

1368 The Notification class represents a Notification from the registry regarding an event that matches a
1369 Subscription. A registry may use a Notification instance to notify a client of an event that matches a
1370 Subscription they have registered. This is a *push* model of notification. A client may also *pull* events
1371 from the registry using the AdhocQuery protocol defined by [ebRS].

1372 7.7.1 Attribute Summary

1373

Attribute	Data Type	Required	Default Value	Specified By	Mutable
subscription	ObjectRef	YES		Registry	No
registryObjectList	Set of Identifiable	No		Registry	No

1374

1375 7.7.2 Attribute subscription

1376 This attribute specifies a reference to a Subscription instance within the registry. This is the
1377 Subscription that matches the event for which this Notification is about.

1378 7.7.3 Attribute registryObjectList

1379 This attribute specifies a Set of ObjectRefs or a Set of RegistryObject instances that represent the
1380 objects that were impacted by the event that matched the Subscription. The registry MUST include
1381 ObjectRef or RegistryObject instances as Set elements depending upon the notificationOption
1382 specified for the Subscription.

8 Cooperating Registries Information Model

1383

1384 This chapter describes the classes in the information model that support the cooperating registries
1385 capability defined by [ebRS].

8.1 Class Registry

1386

1387 **Super Classes:** [RegistryObject](#)

1388 Registry instances are used to represent a single physical OASIS ebXML Registry.

8.1.1.1 Attribute Summary

1389

1390

Attribute	Data Type	Required	Default Value	Specified By	Mutable
catalogingLatency	duration	No	P1D (1 day)	Registry	Yes
conformanceProfile	String16	No	"registry Lite"	Registry	Yes
operator	ObjectRef	Yes		Registry	Yes
replicationSyncLatency	duration	No	P1D (1 day)	Registry	Yes
specificationVersion	Sring8	Yes		Registry	Yes

1391

8.1.2 Attribute catalogingLatency

1392

1393 Each Registry instance MAY have an attribute named *catalogingLatency* that specifies the maximum
1394 latency between the time a submission is made to the registry and the time it gets cataloged by any
1395 cataloging services defined for the objects within the submission.

8.1.3 Attribute conformanceProfile

1396

1397 Each Registry instance MAY have an attribute named *conformanceProfile* that declares the
1398 conformance profile that the registry supports. The conformance profiles choices are "registryLite" and
1399 "registryFull" as defined by [ebRS].

8.1.4 Attribute operator

1400

1401 Each Registry instance MUST have an attribute named *operator* that is a reference to the Organization
1402 instance representing the organization for the registry's operator. Since the same Organization MAY
1403 operate multiple registries, it is possible that the home registry for the Organization referenced by
1404 operator may not be the local registry.

8.1.5 Attribute replicationSyncLatency

1405

1406 Each Registry instance MAY have an attribute named *replicationSyncLatency* that specifies the
1407 maximum latency between the time when an original object changes and the time when its replica
1408 object within the registry gets updated to synchronize with the new state of the original object.

8.1.6 Attribute specificationVersion

1409

1410 Each Registry instance MUST have an attribute named *specificationVersion* that is the version of the
1411 ebXML Registry Services Specification [ebRS].

1412 8.2 Class Federation

1413 **Super Classes:** [RegistryObject](#)

1414 Federation instances are used to represent a registry federation.

1415 8.2.1.1 Attribute Summary

1416

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

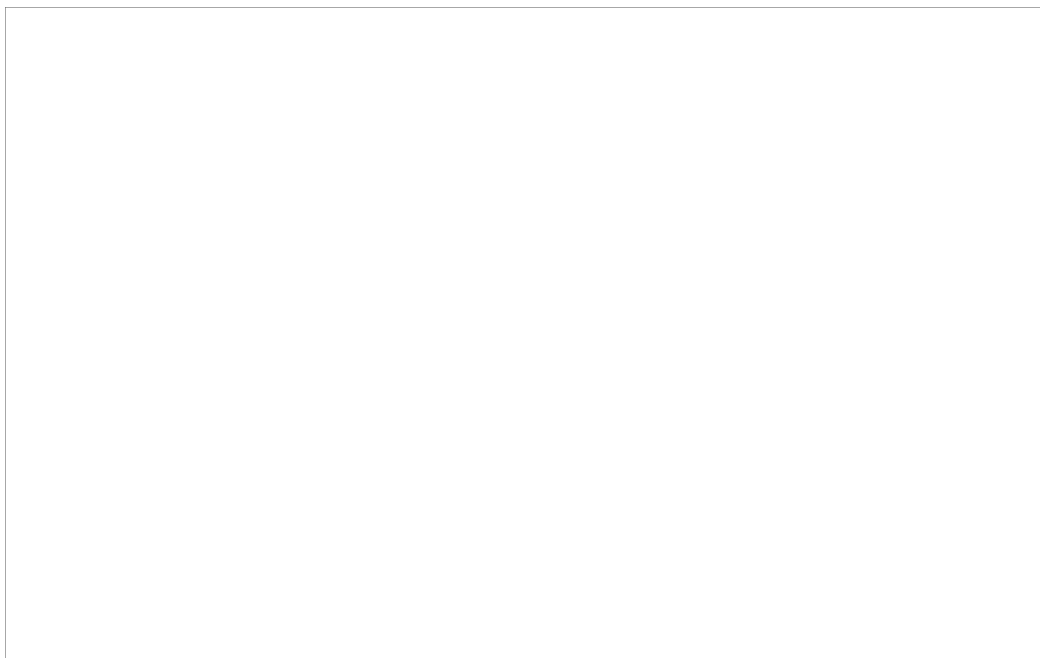
1417

1418 8.2.2 Attribute replicationSyncLatency

1419 Each Federation instance MAY specify a *replicationSyncLatency* attribute that describes the time
1420 duration that is the amount of time within which a member of this Federation MUST synchronize itself
1421 with the current state of the Federation. Members of the Federation MAY use this parameter to
1422 periodically synchronize the federation metadata they MUST cache locally about the state of the
1423 Federation and its members. Such synchronization MAY be based upon the registry event notification
1424 capability.

1425 8.2.3 Federation Configuration

1426 A federation is created by the creation of a Federation instance. Membership of a registry within a
1427 federation is established by creating an Association between the Registry instances for the registry
1428 seeking membership with the Federation instance. The Association MUST have its associationType be
1429 the id of the canonical ClassificationNode "HasFederationMember", the federation instance as its
1430 sourceObject and the Registry instance as its targetObject as shown in Figure 14.



1431

1432

1433

1434

Figure 14: Federation Information Model

1436

9 Access Control Information Model

1437 This chapter defines the Access Control Information Model used by the registry to control access to
1438 RegistryObjects and RepositoryItems managed by it. The Access Control features of the registry
1439 require that it function as both a Policy Enforcement Point (PEP) and a Policy Decision Point (PDP) as
1440 defined in [XACML].

1441 This specification first defines an abstract Access Control Model that enables access control policies to
1442 be defined and associated with RegistryObjects.

1443 Next, it defines a normative and required binding of that abstract model to [XACML].

1444 Finally, it defines how a registry MAY support additional bindings to custom access control
1445 technologies.

9.1 Terminology

1447 The Access Control Model attempts to reuse terms defined by [XACML] wherever possible. The
1448 definitions of some key terms are duplicated here from [XACML] for convenience of the reader:

1449

Term	Description
Access	Performing an action . An example is a user performing a <i>delete action</i> on a RegistryObject.
Access Control	Controlling access in accordance with a policy . An example is preventing a user from performing a <i>delete action</i> on a RegistryObject that is not owned by that user.
Action	An operation on a resource . An example is the <i>delete action</i> on a RegistryObject.
Attribute	Characteristic of a subject, resource, action . Some examples are: <ul style="list-style-type: none"> • <i>id attribute</i> of a subject • <i>role attribute</i> of a subject • <i>group attribute</i> of a subject • <i>id attribute</i> of a RegistryObject resource
Policy	A set of rules . May be a component of a policy set
PolicySet	A set of policies , other policy sets . May be a component of another policy set
Resource	Data, service or system component. Examples are: <ul style="list-style-type: none"> • <i>A RegistryObject resource</i> • <i>A RepositoryItem resource</i>
Subject	An actor whose attributes may be referenced by within a Policy definition. Example: <ul style="list-style-type: none"> • A User instance within the registry

1450

9.2 Use Cases for Access Control Policies

1451 The following are some common use cases for access control policy:
1452

1453 **9.2.1 Default Access Control Policy**

1454 Define a default access control policy that gives *read access* to any one and access to all actions to
1455 owner of the resource and Registry Administrator. This access control policy implicitly applies to any
1456 resource that does not explicitly have a custom Access Control Policy defined for it.

1457 **9.2.2 Restrict Read Access To Specified Subjects**

1458 Define a custom access control policy to restrict *read access* to a resource to specified user(s), group(s)
1459 and/or role(s).

1460 **9.2.3 Grant Update and/or Delete Access To Specified Subjects**

1461 Define a custom access control policy to grant *update* and/or *delete access* to a resource to specified
1462 user(s), group(s) and/or role(s).

1463 **9.2.4 Reference Access Control**

1464 Define a custom access control policy to restrict *reference access* to a resource to specified user(s),
1465 group(s) and/or role(s). For example a custom access control policy MAY be defined to control who can
1466 create an extramural association to a RegistryObject. Another example is to control who can add
1467 members to a RegistryPackage.

1468 **9.3 Resources**

1469 A registry MUST control access to the following types of resources:

1470

- 1471 • *RegistryObject resource* is any instance of RegistryObject class or its sub-classes. Each
1472 RegistryObject resource references an Access Control Policy that controls all access to that
1473 object.
- 1474 • *RepositoryItem resource* is any instance of RepositoryItem class. By default, access control
1475 to a RepositoryItem is managed by the same Access Control Policy as its ExtrinsicObject.


1476

1477 A registry MUST support the following resource attributes.

1478 **9.3.1 Resource Attribute: *owner***

1479 The *owner* attribute of a Resource carries the value of id attribute of the User instance within the
1480 registry that represents the owner of the resource.

1481 **9.3.2 Resource Attribute: *selector***

1482 The *selector* attribute of a Resource carries a string representing a query  define by a sub-type of
1483 AdhocQueryType in [ebRS]. The registry MUST use this query is a filter to select the resources that
1484 match it.

1485 **9.3.3 Resource Attribute: *<attribute>***

1486 The resource attribute *<attribute>* represents any attribute defined by the RegistryObject type or one of
1487 its sub-types. For example, it could be the targetObject attribute in case the resource is an Association
1488 object.

1489 **9.4 Actions**

1490 A registry MUST support the following actions as operations on RegistryObject and RepositoryItem
1491 resources managed by the registry.

1492 **9.4.1 Create Action**

1493 The *create action* creates a RegistryObject or a RepositoryItem. A submitObjects operation performed
1494 on the LifeCycleManager interface of the registry result in a *create action*.

1495 **9.4.2 Read Action**

1496 The *read action* reads a RegistryObject or a RepositoryItem without having any impact on its state. An
1497 operation performed on the QueryManager interface of the registry result in a *read action*. A registry
1498 MUST first perform the query for the read action and then MUST filter out all resources matching the
1499 query for which the client does not have access for the read action.

1500 **9.4.3 Update Action**

1501 The *update action* updates or modifies the state of a RegistryObject or a RepositoryItem. An
1502 updateObjects operation performed on the LifeCycleManager interface of the registry result in a *update*
1503 *action*. A registry MUST evaluate access control policy decision based upon the state of the resource
1504 *before* and not the *after* performing the update action.

1505 **9.4.4 Delete Action**

1506 The *delete action* deletes a RegistryObject or a RepositoryItem. A removeObjects operation performed
1507 on the LifeCycleManager interface of the registry results in a *delete action*.

1508 **9.4.5 Approve Action**

1509 The *approve action* approves a RegistryObject. An approveObjects operation performed on the
1510 LifeCycleManager interface of the registry result in an *approve action*.

1511 **9.4.6 Reference Action**

1512 The *reference action* creates a reference to a RegistryObject. A submitObjects or updateObjects
1513 operation performed on the LifeCycleManager interface of the registry MAY result in a *reference action*.
1514 An example of a reference action is when an Association is created that references a RegistryObject
1515 resource as its source or target object.

1516 **9.4.7 Deprecate Action**

1517 The *deprecate action* deprecates a RegistryObject. A deprecateObjects operation performed on the
1518 LifeCycleManager interface of the registry result in a *deprecate action*.

1519 **9.4.8 Undeprecate Action**

1520 The *undeprecate action* undeprecates a previously deprecated RegistryObject. An undeprecateObjects
1521 operation performed on the LifeCycleManager interface of the registry result in an *undeprecate action*.

1522 **9.4.9 Action Attribute: *action-id***

1523 This attribute identifies the specific action being performed by the subject on one or more resources. A
1524 Registry MUST support access control for all the types of actions identified in this document above.

1525 **9.4.10 Action Attribute: *reference-source***

1526 This attribute is only relevant to the "Reference" action. This attribute MAY be used to specify the
1527 object from which the reference is being made to the resource being protected. The value of this
1528 attribute MUST be the value of the id attribute for the object that is the source of the reference.

1529 **9.4.11 Action Attribute: reference-source-attribute**

1530 This attribute is only relevant to the “Reference” action. This attribute MAY be used to specify the
1531 attribute name within the Class that the reference-source object is an instance of. The value of this
1532 attribute MUST be the name of an attribute within the RIM Class that is the Class for the reference
1533 source object.

1534 For example, if the reference source object is an Association instance then the reference-source-
1535 attribute MAY be used to specify the values “sourceObject” or “targetObject” to restrict the references
1536 to be allowed from only specific attributes of the source object. This enables, for example, a policy to
1537 only allow reference to objects under its protection only from the sourceObject attribute of an
1538 Association instance.

1539 **9.5 Subjects**

1540 A registry MUST support the following Subject attributes within its Access Control Policies. In addition a
1541 registry MAY support additional subject attributes.

1542 **9.5.1 Attribute *id***

1543 The *identity* attribute of a Subject carries the value of *id* attribute of a User instance within the registry.

1544 **9.5.2 Attribute *group***

1545 The *group* attribute of a Subject carries the value of the code attribute of a ClassificationNode within
1546 the canonical SubjectGroup ClassificationScheme (see appendix) within the registry. A registry MUST
1547 NOT allow anyone but a subject with the canonical RegistryAdministrator role to assign roles to users.

1548 **9.5.2.1 Assigning Groups To Users**

1549 Arbitrary groups MAY be defined by extending the canonical SubjectGroup ClassificationScheme.
1550 Groups MAY be assigned to registered users by classifying their User instance with a
1551 ClassificationNode within the canonical SubjectGroup ClassificationScheme.

1552 **9.5.3 Attribute *role***

1553 The *role* attribute of a Subject carries the value of the code attribute of a ClassificationNode within the
1554 canonical SubjectRole ClassificationScheme (see appendix) within the registry.

1555 **9.5.3.1 Assigning Roles To Users**

1556 Arbitrary roles MAY be defined by extending the canonical SubjectRole ClassificationScheme. Roles
1557 MAY be assigned to registered users by classifying their User instance with a ClassificationNode within
1558 the canonical SubjectRole ClassificationScheme. A registry MUST NOT allow anyone but a subject
1559 with the canonical RegistryAdministrator role to assign roles to users. A registry MAY use registry
1560 specific means to assign RegistryAdministrator roles.

1561 **9.6 Abstract Access Control Model**

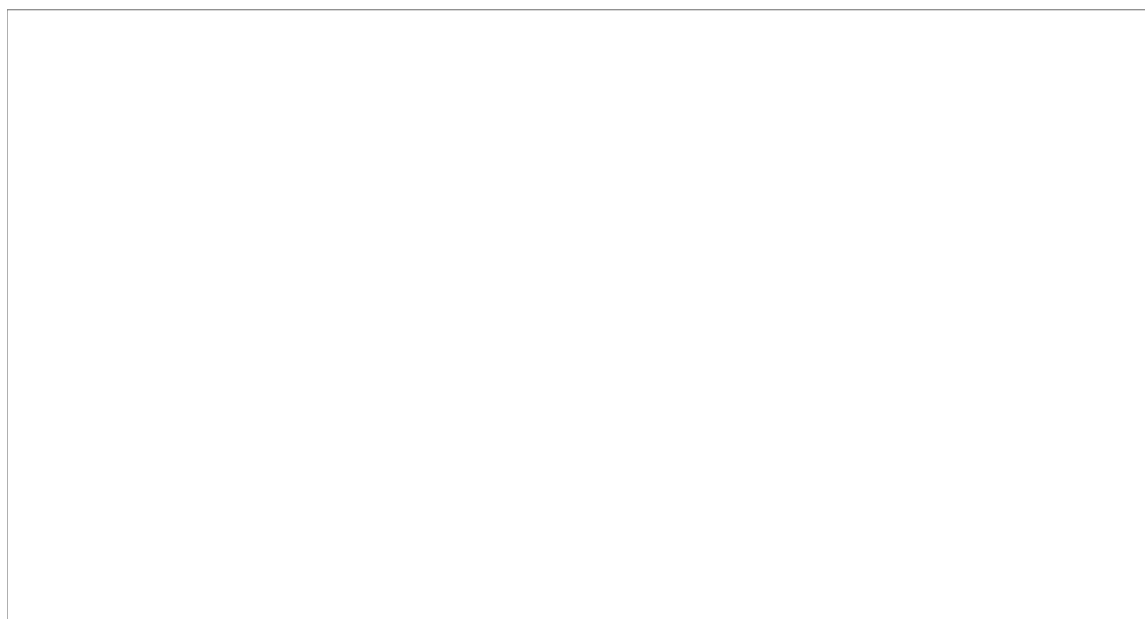
1562 Every RegistryObject is associated with exactly one Access Control Policy that governs “who” is
1563 authorized to perform “what” action on that RegistryObject. The abstract Access Control Model allows
1564 the Access Control Policy to be defined in any arbitrary format as long as it is represented in the
1565 registry as a repositoryItem and its corresponding ExtrinsicObject. The objectType attribute of this
1566 ExtrinsicObject MUST reference a descendent of the “xacml” node (e.g. “Policy” or PolicySet”) in the
1567 canonical ObjectType ClassificationScheme. This distinguishes XACML “Policy” or PolicySet” Access
1568 Control Policy objects from other ExtrinsicObject instances.

1569 **9.6.1 Access Control Policy for a RegistryObject**

1570 A RegistryObject MAY be associated with an Access Control Policy by a special Association with the
1571 canonical associationType of AccessControlPolicyFor. This association has the reference to the
1572 ExtrinsicObject representing the Access Control Policy as the value of its sourceObject and has the
1573 reference to the RegistryObject as the value of its targetObject attribute.

1574 If a RegistryObject does not have an Access Control Policy explicitly associated with it, then it is
1575 implicitly associated with the default Access Control Policy defined for the registry.

1576



1577

1578

Figure 15: Instance Diagram for Abstract Access Control Information Model

1579 Figure 15 shows an instance diagram where an Organization instance *org* references an
1580 ExtrinsicObject instance *accessControlPolicy* as its Access Control Policy object. The
1581 *accessControlPolicy* object has its objectType attribute referencing a node in the canonical ObjectType
1582 ClassificationScheme that represents a supported Access Control Policy format. The
1583 *accessControlPolicy* ExtrinsicObject has a repositoryItem defining its access control policy information
1584 in a specific format.

1585 **9.6.2 Access Control Policy for a RepositoryItem**

1586 By default, access control to a RepositoryItem is managed by the Access Control Policy associated with
1587 its ExtrinsicObject that provides metadata for the RepositoryItem. A RepositoryItem MAY have an
1588 Access Control Policy separate from its ExtrinsicObject. In such case, the Access Control Policy for the
1589 RepositoryItem is referenced via a Special Slot on its ExtrinsicObject. This special Slot has
1590 "repositoryItemACP" as its name and the id of the ExtrinsicObject representing the Access Control
1591 Policy for the RepositoryItem as its value.

1592 **9.6.3 Default Access Control Policy**

1593 A registry MUST support the default Access Control Policy.

1594 The default Access Control Policy applies to any RegistryObject that does not explicitly have an Access
1595 Control Policy associated with it.

- 1596 • The following list summarizes the default Access Control Policy semantic that a registry
1597 SHOULD implement:
- 1598 • Only a Registered User is granted access to create actions.

- 1599 • An unauthenticated Registry Client is granted access to read actions. The Registry MUST
1600 assign the default RegistryGuest role to such Registry Clients.
- 1601 • A Registered User has access to all actions on Registry Objects submitted by the Registered
1602 User.
- 1603 • The Registry Administrator and Registry Authority have access to all actions on all Registry
1604 Objects.
- 1605

1606 A registry MAY have a default access control policy that differs from the above semantics.

1607 **9.6.4 Root Access Control Policy**

1608 A registry SHOULD have a root Access Control Policy that bootstraps the Access Control Model by
1609 controlling access to Access Control Policies.

1610 As described in Figure 15, an access control policy is an ExtrinsicObject that contains a pointer to a
1611 repository item. The access control policies themselves are created, updated, and deleted.

1612 To define who may create access control policies pertaining to specified resources, it is necessary to
1613 have one or more administrative Access Control Policies. Such policies restrict Registry Users from
1614 creating access control policies to unauthorized resources. This version of the Registry specifications
1615 defines a single Root Access Control Policy that allows all actions on Access Control Policies for a
1616 resource under the following conditions:

- 1617 • Subject is the owner of the resource
- 1618 • Subject has a role of RegistryAdministrator

1619 **9.6.5 Performance Implications**

1620 Excessive use of custom Access Control Policies MAY result in slower processing of registry requests
1621 in some registry implementations. It is therefor suggested that, whenever possible, a submitter
1622 SHOULD reuse an existing Access Control Policy. Submitters SHOULD use good judgement on when
1623 to reuse or extend an existing Access Control Policy and when to create a new one.

1624 **9.7 Access Control Model: XACML Binding**

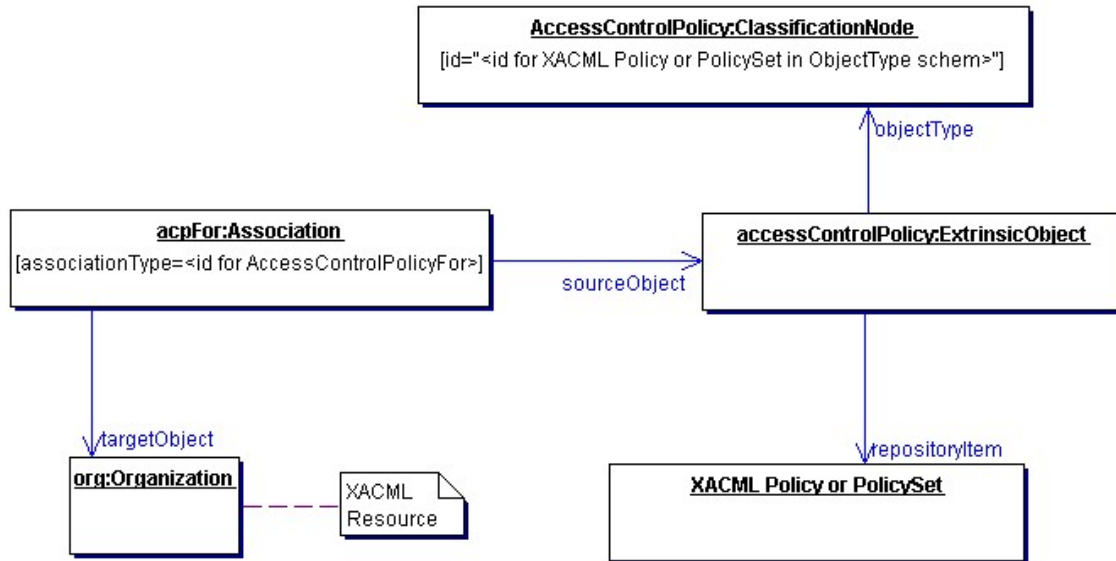
1625 A registry MAY support custom access control policies based upon a normative though optional binding
1626 of the Access Control Model to [XACML].

1627 This section defines the normative though optional binding of the abstract Access Control Model to
1628 [XACML]. This section assumes the reader is familiar with [XACML].

1629 This binding to [XACML] enables a flexible access control mechanism that supports access control
1630 policy definition from the simples to the most sophisticated use cases.

1631 In this binding the policyInfo repositoryItem in the abstract Access Control Model MUST be one of the
1632 following:

- 1633 • A PolicySet as defined by [XACML]
- 1634 • A Policy as defined by [XACML]
- 1635



1636
1637

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

1638 9.7.1 Resource Binding

1639 [XACML] defines an element called ResourceAttributeDesignator that identifies the type of resource
1640 attribute being specified in a ResourceMatch or Apply element.

1641 The resource attributes defined by the abstract Access Control Model map to the following
1642 ResourceAttributeDesignator definitions:

1643

Resource Attribute	AttributeId	Data Type
owner	urn:oasis:names:tc:ebxml-regrep:rim:acp:resource:owner	http://www.w3.org/2001/XMLSchema#anyURI
selector	urn:oasis:names:tc:ebxml-regrep:rim:acp:resource:selector	http://www.w3.org/2001/XMLSchema#string
<attribute>	urn:oasis:names:tc:ebxml-regrep:rim:acp:resource:<attribute>	Depends upon the specific attribute.

1644
1645
1646

Table 2: Resource Binding to [XACML]

Data Type	XACML Data Type Identifier URI	Description
Boolean	http://www.w3.org/2001/XMLSchema#boolean	
String	http://www.w3.org/2001/XMLSchema#string	Used strings of all lengths
ObjectRef	http://www.w3.org/2001/XMLSchema#anyURI	
URI	http://www.w3.org/2001/XMLSchema#anyURI	
Integer	http://www.w3.org/2001/XMLSchema#integer	
Date Time	http://www.w3.org/2001/XMLSchema#dateTime	

1647

1648 9.7.2 Action Binding

1649 [XACML] defines an element called ActionAttributeDesignator that identifies the type of action being
1650 specified within in an ActionMatch or Apply element.

1651 The actions defined by the abstract Access Control Model map to the following AttributeId and

1652 AttributeValue in the ActionMatch definitions:

1653

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
Reference	urn:oasis:names:tc:xacml:1.0:action:action-id	reference
Deprecate	urn:oasis:names:tc:xacml:1.0:action:action-id	deprecate
Undeprecate	urn:oasis:names:tc:xacml:1.0:action:action-id	undeprecate

1654

Table 3: Action Binding to [XACML]

1655

Action Attribute	ActionAttributeDesignator.AttributeId	DataType
id	urn:oasis:names:tc:xacml:1.0:action:action-id	http://www.w3.org/2001/XMLSchema#anyURI
reference-source	urn:oasis:names:tc:ebxml-regrep:rim:acp:subject:reference-source	http://www.w3.org/2001/XMLSchema#string
reference-source-attribute	urn:oasis:names:tc:ebxml-regrep:rim:acp:subject:reference-source-attribute	http://www.w3.org/2001/XMLSchema#string

1656

1657 9.7.3 Subject Binding

1658 [XACML] defines an element called SubjectAttributeDesignator that identifies the type of subject
1659 attribute being specified in a SubjectMatch or Apply element.

1660

1661 The subjects defined by the abstract Access Control Model map to the following
1662 SubjectAttributeDesignator definitions:

1663

Subject Attribute	SubjectAttributeDesignator	DataType
id	urn:oasis:names:tc:xacml:1.0:subject:subject-id	http://www.w3.org/2001/XMLSchema#anyURI
roles	urn:oasis:names:tc:ebxml-regrep:rim:acp:subject:roles	http://www.w3.org/2001/XMLSchema#string
groups	urn:oasis:names:tc:ebxml-regrep:rim:acp:subject:groups	http://www.w3.org/2001/XMLSchema#string
<attribute>	urn:oasis:names:tc:ebxml-regrep:rim:acp:subject:<attribute>	As defined by attribute definition. Can be any attribute of the User instance for the subject.

1664

Table 4: Subject Binding to [XACML]

1665 9.7.4 Function classification-node-compare

1666 It is often necessary to test whether a resource matches a specific objectType or its sub-types. A client
1667 MAY use the special XACML function named *classification-node-compare* to perform such
1668 comparisons.

1669 A registry MUST support a special XACML function named *classification-node-compare* whose
1670 canonical id is *urn:oasis:names:tc:ebxml-regrep:rim:acp:function:classification-node-compare*. A client
1671 MAY use this function within XACML Access control Policies to perform ClassificationNode

1672 comparisons in a taxonomy-aware manner. The following example shows how a ResourceMatch may
1673 be specified within an XACML Access Control Policy to perform such comparisons.

1674

```
1675 <!-- match ExtrinsicObject -->  
1676 <ResourceMatch  
1677 MatchId="urn:oasis:names:tc:ebxml-  
1678 regrep:rim:acp:function:classification-node-compare">  
1679 <!--Specify the id for canonical ClassificationNode for  
1680 ExtrinsicObject objectType-->  
1681 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">  
1682 urn:oasis:names:tc:ebxml-  
1683 regrep:ObjectType:RegistryObject:ExtrinsicObject  
1684 </AttributeValue>  
1685  
1686 <!--Specify the objectType of resource to compare with objectType  
1687 ExtrinsicObject -->  
1688 <ResourceAttributeDesignator DataType =  
1689 "http://www.w3.org/2001/XMLSchema#string"  
1690 AttributeId = "urn:oasis:names:tc:ebxml-  
1691 regrep:rim:acp:resource:objectType"/>  
1692 </ResourceMatch>
```

1693

1694 9.7.5 Constraints on XACML Binding

1695 This specification normatively defines the following constraints on the binding of the Access Control
1696 Model to [XACML]. These constraints MAY be relaxed in future versions of this specification.

- 1697 • All Policy and PolicySet definitions MUST reside within an ebXML Registry as RepositoryItems.

1698 9.7.6 Example: Default Access Control Policy

1699 The following Policy defines the default access control policy. This Policy MUST implicitly apply to any
1700 resource that does not have an explicit Access Control Policy defined.

1701 It consists of 3 rules, which in plain English are described as follows:

1702

- 1703 • Any subject can perform read action on any resource
- 1704 • A subject may perform any action on a resource for which they are the owner.
- 1705 • A subject with role of RegistryAdministrator may perform any action on any resource.

1706

1707 The non-normative listing of the default Access Control Policy follows:

1708

```
1709 <?xml version="1.0" encoding="UTF-8"?>  
1710 <PolicySet PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-  
1711 combining-algorithm:permit-overrides"  
1712 PolicySetId="urn:oasis:names:tc:ebxml-  
1713 regrep:3.0:rim:acp:policy:default-access-control-policy"  
1714 xmlns="urn:oasis:names:tc:xacml:1.0:policy"  
1715 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
1716 xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy cs-xacml-  
1717 schema-policy-01.xsd">  
1718 <Description>This PolicySet defines the default Access Control Policy  
1719 for all registry resources.</Description>  
1720 <Target>  
1721 <Subjects>  
1722 <AnySubject/>  
1723 </Subjects>  
1724 <Resources>  
1725 <AnyResource/>
```

```

1726     </Resources>
1727     <Actions>
1728         <AnyAction/>
1729     </Actions>
1730 </Target>
1731     <Policy PolicyId="urn:oasis:names:tc:ebxml-
1732 regrep:3.0:rim:acp:policy:policyid:permit-anyone-to-read"
1733 RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1734 algorithm:permit-overrides">
1735     <Target>
1736         <Subjects>
1737             <AnySubject/>
1738         </Subjects>
1739         <Resources>
1740             <AnyResource/>
1741         </Resources>
1742         <Actions>
1743             <AnyAction/>
1744         </Actions>
1745     </Target>
1746     <Rule Effect="Permit" RuleId="urn:oasis:names:tc:ebxml-
1747 regrep:3.0:rim:acp:rule:ruleid:permit-anyone-to-read">
1748         <Description>Any Subject can perform read action on any
1749 resource.</Description>
1750         <Target>
1751             <Subjects>
1752                 <AnySubject/>
1753             </Subjects>
1754             <Resources>
1755                 <AnyResource/>
1756             </Resources>
1757             <Actions>
1758                 <Action>
1759                     <ActionMatch
1760 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1761                         <AttributeValue
1762 DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue
1763 >
1764                             <ActionAttributeDesignator
1765 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1766 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1767                             </ActionMatch>
1768                         </Action>
1769                     </Actions>
1770                 </Target>
1771             </Rule>
1772     </Policy>
1773     <Policy PolicyId="urn:oasis:names:tc:ebxml-
1774 regrep:3.0:rim:acp:policy:policyid:permit-anyone-to-reference"
1775 RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1776 algorithm:permit-overrides">
1777     <Target>
1778         <Subjects>
1779             <AnySubject/>
1780         </Subjects>
1781         <Resources>
1782             <AnyResource/>
1783         </Resources>
1784         <Actions>
1785             <AnyAction/>
1786         </Actions>
1787     </Target>

```

```

1788     <Rule Effect="Permit" RuleId="urn:oasis:names:tc:ebxml-
1789 regrep:3.0:rim:acp:rule:ruleid:permit-anyone-to-reference">
1790     <Description>Any Subject can perform reference action on any
1791 resource as long as it is not deprecated.</Description>
1792     <Target>
1793     <Subjects>
1794     <AnySubject/>
1795     </Subjects>
1796     <Resources>
1797     <AnyResource/>
1798     </Resources>
1799     <Actions>
1800     <Action>
1801     <ActionMatch
1802 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1803     <AttributeValue
1804 DataType="http://www.w3.org/2001/XMLSchema#string">reference</Attribute
1805 Value>
1806     <ActionAttributeDesignator
1807 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1808 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1809     </ActionMatch>
1810     </Action>
1811     </Actions>
1812     </Target>
1813     <Condition
1814 FunctionId="urn:oasis:names:tc:xacml:1.0:function:not">
1815     <Apply
1816 FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1817     <Apply
1818 FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-one-and-only">
1819     <ResourceAttributeDesignator
1820 AttributeId="urn:oasis:names:tc:ebxml-
1821 regrep:3.0:rim:acp:resource:status"
1822 DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1823     </Apply>
1824     <!-- Compare with the id for deprecated status -->
1825     <AttributeValue
1826 DataType="http://www.w3.org/2001/XMLSchema#anyURI">urn:oasis:names:tc:e
1827 bxml-regrep:StatusType:Deprecated</AttributeValue>
1828     </Apply>
1829     </Condition>
1830     </Rule>
1831   </Policy>
1832   <Policy PolicyId="urn:oasis:names:tc:ebxml-
1833 regrep:3.0:rim:acp:policy:policyid:permit-owner-all"
1834 RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1835 algorithm:permit-overrides">
1836     <Target>
1837     <Subjects>
1838     <AnySubject/>
1839     </Subjects>
1840     <Resources>
1841     <AnyResource/>
1842     </Resources>
1843     <Actions>
1844     <AnyAction/>
1845     </Actions>
1846     </Target>
1847     <Rule Effect="Permit" RuleId="urn:oasis:names:tc:ebxml-
1848 regrep:3.0:rim:acp:rule:ruleid:permit-owner-all">
1849     <Description>A Subject with role of ContenOwner can perform any
1850 action on resources owned by them.</Description>

```

```

1851     <Target>
1852         <Subjects>
1853             <AnySubject/>
1854         </Subjects>
1855         <Resources>
1856             <AnyResource/>
1857         </Resources>
1858         <Actions>
1859             <AnyAction/>
1860         </Actions>
1861     </Target>
1862     <Condition
1863 FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1864         <Apply
1865 FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-one-and-only">
1866             <SubjectAttributeDesignator
1867 AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
1868 DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1869             </Apply>
1870         <Apply
1871 FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-one-and-only">
1872             <ResourceAttributeDesignator
1873 AttributeId="urn:oasis:names:tc:ebxml-
1874 regrep:3.0:rim:acp:resource:owner"
1875 DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1876             </Apply>
1877         </Condition>
1878     </Rule>
1879 </Policy>
1880 <Policy PolicyId="urn:oasis:names:tc:ebxml-
1881 regrep:3.0:rim:acp:policy:policyid:permit-registryadministrator-all"
1882 RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1883 algorithm:permit-overrides">
1884     <Target>
1885         <Subjects>
1886             <AnySubject/>
1887         </Subjects>
1888         <Resources>
1889             <AnyResource/>
1890         </Resources>
1891         <Actions>
1892             <AnyAction/>
1893         </Actions>
1894     </Target>
1895     <Rule Effect="Permit" RuleId="urn:oasis:names:tc:ebxml-
1896 regrep:3.0:rim:acp:rule:ruleid:permit-registryadministrator-all">
1897         <Description>A Subject with role of RegistryAdministrator can
1898 perform any action on any resource.</Description>
1899         <Target>
1900             <Subjects>
1901                 <Subject>
1902                     <SubjectMatch
1903 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1904                         <AttributeValue
1905 DataType="http://www.w3.org/2001/XMLSchema#string"/>urn:oasis:names:tc:
1906 ebxml-
1907 regrep:classificationScheme:SubjectRole/RegistryAdministrator</Attribut
1908 eValue>
1909                             <SubjectAttributeDesignator
1910 AttributeId="urn:oasis:names:tc:ebxml-regrep:3.0:rim:acp:subject:roles"
1911 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1912                         </SubjectMatch>
1913                     </Subject>

```

```

1914     </Subjects>
1915     <Resources>
1916         <AnyResource/>
1917     </Resources>
1918     <Actions>
1919         <AnyAction/>
1920     </Actions>
1921 </Target>
1922 </Rule>
1923 </Policy>
1924 </PolicySet>
1925
1926

```

1927 **9.7.7 Example: Custom Access Control Policy**

1928 The following Policy defines a custom access control policy to restrict *read access* to a resource to
1929 specified user or role. It also grants update access to specified role.
1930 It consists of 3 rules, which in plain English are described as follows:

- 1931
- 1932 1. A subject may perform any action on a resource for which they are the owner. This reuses a
- 1933 Policy by reference from the default Access Control PolicySet.
- 1934 2. A subject with the role of RegistryAdministrator may perform any action on any resource. This
- 1935 reuses a Policy by reference from the default Access Control PolicySet.
- 1936 3. A subject with specified id may perform read actions on the resource. This restricts read
- 1937 access to the specified subject.
- 1938 4. A subject with role of Manager may perform update actions on the resource. This relaxes
- 1939 update access restrictions to the specified subject.
- 1940

1941 The listing of the custom Access Control Policy follows:

```

1942
1943 <?xml version="1.0" encoding="UTF-8"?>
1944 <PolicySet PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
1945 combining-algorithm:permit-overrides"
1946 PolicySetId="urn:oasis:names:tc:ebxml-
1947 regrep:3.0:rim:acp:policy:restricted-access-control-policyset"
1948 xmlns="urn:oasis:names:tc:xacml:1.0:policy"
1949 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1950 xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy cs-xacml-
1951 schema-policy-01.xsd">
1952   <Description>This PolicySet restricts the default Access Control
1953   Policy to limit read access to specified subjects.</Description>
1954   <Target>
1955     <Subjects>
1956       <AnySubject/>
1957     </Subjects>
1958     <Resources>
1959       <AnyResource/>
1960     </Resources>
1961     <Actions>
1962       <AnyAction/>
1963     </Actions>
1964   </Target>
1965   <PolicyIdReference>urn:oasis:names:tc:ebxml-
1966   regrep:3.0:rim:acp:policy:policyid:permit-owner-all</PolicyIdReference>
1967   <PolicyIdReference>urn:oasis:names:tc:ebxml-
1968   regrep:3.0:rim:acp:policy:policyid:permit-registryadministrator-
1969   all</PolicyIdReference>

```

```

1970     <Policy PolicyId="urn:oasis:names:tc:ebxml-
1971 regrep:3.0:rim:acp:policy:permit-delete-access-control-policy"
1972 RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1973 algorithm:permit-overrides" xmlns="urn:oasis:names:tc:xacml:1.0:policy"
1974 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1975 xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy cs-xacml-
1976 schema-policy-01.xsd">
1977     <Description>Allow Subject with specifed id to perform delete
1978 action on any resource.</Description>
1979     <Target>
1980         <Subjects>
1981             <AnySubject/>
1982         </Subjects>
1983         <Resources>
1984             <AnyResource/>
1985         </Resources>
1986         <Actions>
1987             <AnyAction/>
1988         </Actions>
1989     </Target>
1990     <Rule Effect="Permit" RuleId="urn:oasis:names:tc:ebxml-
1991 regrep:3.0:rim:acp:rule:ruleid:permit-delete-rule">
1992     <Description>Allow Subject with specifed id to perform delete
1993 action on any resource.</Description>
1994     <Target>
1995         <Subjects>
1996             <Subject>
1997                 <SubjectMatch
1998 MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1999                 <AttributeValue
2000 DataType="http://www.w3.org/2001/XMLSchema#anyURI">urn:freebxml:registr
2001 y:predefinedusers:farrukh</AttributeValue>
2002                 <SubjectAttributeDesignator
2003 AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
2004 DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
2005                 </SubjectMatch>
2006             </Subject>
2007         </Subjects>
2008         <Resources>
2009             <AnyResource/>
2010         </Resources>
2011         <Actions>
2012             <Action>
2013                 <ActionMatch
2014 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
2015                 <AttributeValue
2016 DataType="http://www.w3.org/2001/XMLSchema#string">delete</AttributeVal
2017 ue>
2018                 <ActionAttributeDesignator
2019 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
2020 DataType="http://www.w3.org/2001/XMLSchema#string"/>
2021                 </ActionMatch>
2022             </Action>
2023         </Actions>
2024     </Target>
2025 </Rule>
2026 </Policy>

```

```

2027     <Policy PolicyId="urn:oasis:names:tc:ebxml-
2028 regrep:3.0:rim:acp:policy:permit-update-access-control-policy"
2029 RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
2030 algorithm:permit-overrides" xmlns="urn:oasis:names:tc:xacml:1.0:policy"
2031 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2032 xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy cs-xacml-
2033 schema-policy-01.xsd">
2034     <Description>Allow Subjects with ProjectLead role to perform update
2035 action on any resource.</Description>
2036     <Target>
2037         <Subjects>
2038             <AnySubject/>
2039         </Subjects>
2040         <Resources>
2041             <AnyResource/>
2042         </Resources>
2043         <Actions>
2044             <AnyAction/>
2045         </Actions>
2046     </Target>
2047     <Rule Effect="Permit" RuleId="urn:oasis:names:tc:ebxml-
2048 regrep:3.0:rim:acp:rule:ruleid:permit-update-rule">
2049         <Description>Allow Subjects with ProjectLead role to perform read
2050 action on any resource.</Description>
2051         <Target>
2052             <Subjects>
2053                 <Subject>
2054                     <SubjectMatch
2055 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
2056                     <AttributeValue
2057 DataType="http://www.w3.org/2001/XMLSchema#string">/urn:oasis:names:tc:
2058 ebxml-
2059 regrep:classificationScheme:SubjectRole/ProjectMember/ProjectLead</Attr
2060 ibuteValue>
2061                     <SubjectAttributeDesignator
2062 AttributeId="urn:oasis:names:tc:ebxml-regrep:3.0:rim:acp:subject:roles"
2063 DataType="http://www.w3.org/2001/XMLSchema#string"/>
2064                     </SubjectMatch>
2065                 </Subject>
2066             </Subjects>
2067             <Resources>
2068                 <AnyResource/>
2069             </Resources>
2070             <Actions>
2071                 <Action>
2072                     <ActionMatch
2073 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
2074                     <AttributeValue
2075 DataType="http://www.w3.org/2001/XMLSchema#string">update</AttributeVal
2076 ue>
2077                     <ActionAttributeDesignator
2078 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
2079 DataType="http://www.w3.org/2001/XMLSchema#string"/>
2080                     </ActionMatch>
2081                 </Action>
2082             </Actions>
2083         </Target>
2084     </Rule>
2085 </Policy>
2086 </PolicySet>
2087

```

2088 9.7.8 Example: Package Membership Access Control

2089 The following Policy defines an access control policy for controlling who can add members to a
2090 RegistryPackage. It makes use of the Reference action.

2091 It consists of 3 rules, which in plain English are described as follows:

2092

- 2093 1. Any subject can perform read action on any resource. Referenced from default access control
2094 policy.
- 2095 2. A subject may perform any action on a resource for which they are the owner. Referenced
2096 from default access control policy.
- 2097 3. A subject with role of RegistryAdministrator may perform any action on any resource.
2098 Referenced from default access control policy
- 2099 4. A subjects with role ProjectLead may perform addmember action on any resource associated
2100 with this ACP.
2101

2102 The following is a non-normative example listing of this custom Access Control Policy:

2103

```
2104 <?xml version="1.0" encoding="UTF-8"?>
2105 <PolicySet PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
2106 combining-algorithm:permit-overrides"
2107 PolicySetId="urn:oasis:names:tc:ebxml-
2108 regrep:3.0:rim:acp:policy:folderACP1"
2109 xmlns="urn:oasis:names:tc:xacml:1.0:policy"
2110 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2111 xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy cs-xacml-
2112 schema-policy-01.xsd">
2113   <Description>This PolicySet restricts adding members to
2114 RegistryPackage resource to Role ProjectLead</Description>
2115   <Target>
2116     <Subjects>
2117       <AnySubject/>
2118     </Subjects>
2119     <Resources>
2120       <AnyResource/>
2121     </Resources>
2122     <Actions>
2123       <AnyAction/>
2124     </Actions>
2125   </Target>
2126   <PolicyIdReference>urn:oasis:names:tc:ebxml-
2127 regrep:3.0:rim:acp:policy:policyid:permit-anyone-to-
2128 read</PolicyIdReference>
2129   <PolicyIdReference>urn:oasis:names:tc:ebxml-
2130 regrep:3.0:rim:acp:policy:policyid:permit-owner-all</PolicyIdReference>
2131   <PolicyIdReference>urn:oasis:names:tc:ebxml-
2132 regrep:3.0:rim:acp:policy:policyid:permit-registryadministrator-
2133 all</PolicyIdReference>
2134   <Policy PolicyId="urn:oasis:names:tc:ebxml-
2135 regrep:3.0:rim:acp:policy:permit-projectLead-addMember"
2136 RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
2137 algorithm:permit-overrides" xmlns="urn:oasis:names:tc:xacml:1.0:policy"
2138 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2139 xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy cs-xacml-
2140 schema-policy-01.xsd">
2141     <Description>Allow Subjects with ProjectLead role to add members to
2142 any resource associated with this ACP.</Description>
2143     <Target>
2144       <Subjects>
2145         <AnySubject/>
2146       </Subjects>
```



```

2147     <Resources>
2148         <AnyResource/>
2149     </Resources>
2150     <Actions>
2151         <AnyAction/>
2152     </Actions>
2153 </Target>
2154     <Rule Effect="Permit" RuleId="urn:oasis:names:tc:ebxml-
2155 regrep:3.0:rim:acp:rule:ruleid:permit-projectLead-addMember-rule">
2156         <Description>Allow Subjects with ProjectLead role to add members
2157 to any resource.</Description>
2158         <Target>
2159             <Subjects>
2160                 <Subject>
2161                     <!-- Match role ProjectLead -->
2162                     <SubjectMatch
2163 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
2164                         <AttributeValue
2165 DataType="http://www.w3.org/2001/XMLSchema#string">/urn:oasis:names:tc:
2166 ebxml-
2167 regrep:classificationScheme:SubjectRole/ProjectMember/ProjectLead</Attr
2168 ibuteValue>
2169                             <SubjectAttributeDesignator
2170 AttributeId="urn:oasis:names:tc:ebxml-regrep:3.0:rim:acp:subject:roles"
2171 DataType="http://www.w3.org/2001/XMLSchema#string"/>
2172                             </SubjectMatch>
2173                         </Subject>
2174                     </Subjects>
2175                 <Resources>
2176                     <AnyResource/>
2177                 </Resources>
2178                 <Actions>
2179                     <Action>
2180                         <!-- Match "reference" action -->
2181                         <ActionMatch
2182 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
2183                             <AttributeValue
2184 DataType="http://www.w3.org/2001/XMLSchema#string">reference</Attribute
2185 Value>
2186                                 <ActionAttributeDesignator
2187 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
2188 DataType="http://www.w3.org/2001/XMLSchema#string"/>
2189                                 </ActionMatch>
2190                             </Action>
2191                         </Actions>
2192                     </Target>
2193                 <!--
2194                     Match condition where all the following are true:
2195                     1. reference is being made via the attribute sourceObject
2196 (from an Association instance)
2197                     2. The associationType attribute of the Association matches
2198 the id for associationType HasMameber
2199
2200                     Above is equivalent to saying Match any HasMember
2201 associations where the resource
2202                     (the RegistryPackage) is the sourceObject.
2203                     -->
2204                 <Condition
2205 FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
2206                     <Apply
2207 FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">

```

```

2208         <AttributeValue
2209         DataType="http://www.w3.org/2001/XMLSchema#string">SourceObject</Attrib
2210         uteValue>
2211         <Apply
2212         FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
2213         <ActionAttributeDesignator
2214         AttributeId="urn:oasis:names:tc:ebxml-
2215         regrep:3.0:rim:acp:action:reference-source-attribute"
2216         DataType="http://www.w3.org/2001/XMLSchema#string"/>
2217         </Apply>
2218         </Apply>
2219         <Apply
2220         FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
2221         <AttributeValue
2222         DataType="http://www.w3.org/2001/XMLSchema#anyURI">urn:oasis:names:tc:e
2223         bxml-regrep:AssociationType:HasMember</AttributeValue>
2224         <Apply
2225         FunctionId="urn:oasis:names:tc:xacml:1.0:function:anyURI-one-and-only">
2226         <ActionAttributeDesignator
2227         AttributeId="urn:oasis:names:tc:ebxml-
2228         regrep:3.0:rim:acp:action:reference-source-attribute-
2229         filter:associationType"
2230         DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
2231         </Apply>
2232         </Apply>
2233         </Condition>
2234         </Rule>
2235         </Policy>
2236     </PolicySet>
2237

```

2238 9.7.9 Resolving Policy References

2239 An XACML PolicySet MAY reference XACML Policy objects defined outside the repository item
2240 containing the XACML PolicySet. A registry implementation MUST be able to resolve such references.
2241 To resolve such references efficiently a registry SHOULD be able to find the repository item containing
2242 the referenced Policy without having to load and search all Access Control Policies in the repository.
2243 This section describes the normative behavior that enables a registry to resolve policy references
2244 efficiently.

2245 A registry SHOULD define a Content Cataloging Service for the canonical XACML PolicySet
2246 objectType. The PolicySet cataloging service MUST automatically catalog every PolicySet upon
2247 submission to contain a special Slot with name ComposedPolicies. The value of this Slot MUST be a
2248 Set where each element in the Set is the id for a Policy object that is composed within the PolicySet.

2249 Thus a registry is able to use an ad hoc query to find the repositoryItem representing an XACML
2250 PolicySet that contains the Policy that is being referenced by another PolicySet.

2251 9.7.10 ebXML Registry as a XACML Policy Store

2252 So far we have defined how ebXML registries MAY use [XACML] to define Access Control Policies to
2253 control access to RegistryObject and RepositoryItem resources.

2254 An important side effect of the normative binding of the Access Control Model to [XACML] is that
2255 enterprises MAY also use ebXML Registry as a [XACML] Policy store to manage Policies for protecting
2256 resources outside the registry.

2257 In this use case, enterprises may submit [XACML] Policies and PolicySets as ExtrinsicObject-
2258 RepositoryItem pairs. These Policies may be accessed or referenced by their URL as defined by the
2259 HTTP binding of the ebXML Registry Services interface in [ebRS].

2260 **9.8 Access Control Model: Custom Binding**

2261 A registry MAY support bindings to policies describes in formats other than [XACML]. The use of such
2262 policies sacrifices interoperability and is therefore discouraged. In such cases the RepositoryItem for
2263 the policy information MAY be in any format supported by the registry in an implementation specific
2264 manner.

2265

10 References

2266

10.1 Normative References

- 2267 **[RFC2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, IETF
2268 RFC 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>.
- 2269 **[ebRS]** ebXML Registry Services Specification Version 3.0
2270 [http://www.oasis-open.org/committees/regrep/documents/3.0/specs/regrep-rs-](http://www.oasis-open.org/committees/regrep/documents/3.0/specs/regrep-rs-3.0-cs-01.pdf)
2271 [3.0-cs-01.pdf](http://www.oasis-open.org/committees/regrep/documents/3.0/specs/regrep-rs-3.0-cs-01.pdf)
- 2272 **[UUID]** DCE 128 bit Universal Unique Identifier
2273 http://www.opengroup.org/onlinepubs/009629399/apdx.htm#tagcjh_20
- 2274 **[RFC 3066]** H. Alvestrand, ed. *RFC 3066: Tags for the Identification of Languages* 1995.
2275 <http://www.ietf.org/rfc/rfc3066.txt>
- 2276 **[XPath]** XML Path Language (XPath) Version 1.0
2277 <http://www.w3.org/TR/xpath>
- 2278 **[XACML]** OASIS eXtensible Access Control Markup Language (XACML) Version 1.0
2279 [http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-](http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-01.pdf)
2280 [01.pdf](http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-01.pdf)
- 2281 **[NCName]** Namespaces in XML 19990114
2282 <http://www.w3.org/TR/REC-xml-names/#NT-NCName>

2283

10.2 Informative References

- 2284 **[ISO]** ISO 11179 Information Model
2285 [http://208.226.167.205/SC32/jtc1sc32.nsf/576871ad2f11bba785256621005419](http://208.226.167.205/SC32/jtc1sc32.nsf/576871ad2f11bba785256621005419d7/b83fc7816a6064c68525690e0065f913?OpenDocument)
2286 [d7/b83fc7816a6064c68525690e0065f913?OpenDocument](http://208.226.167.205/SC32/jtc1sc32.nsf/576871ad2f11bba785256621005419d7/b83fc7816a6064c68525690e0065f913?OpenDocument)
- 2287 **[UML]** Unified Modeling Language
2288 <http://www.uml.org>
2289 <http://www.omg.org/cgi-bin/doc?formal/03-03-01>

2290

A. Acknowledgments

2291

The editors would like to acknowledge the contributions of the OASIS ebXML Registry Technical Committee, whose voting members at the time of publication are listed as contributors on the title page of this document.

2292

2293

2294

Finally, the editors wish to acknowledge the following people for their contributions of material used as input to the OASIS ebXML Registry specifications:

2295

2296

Name	Affiliation
Aziz Abouelfoutouh	Government of Canada
Ed Buchinski	Government of Canada
Asuman Dogac	Middle East Technical University, Ankara Turkey
Michael Kass	NIST
Richard Lessard	Government of Canada
Evan Wallace	NIST
David Webber	Individual

2297

B. Notices

2299 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
2300 might be claimed to pertain to the implementation or use of the technology described in this document
2301 or the extent to which any license under such rights might or might not be available; neither does it
2302 represent that it has made any effort to identify any such rights. Information on OASIS's procedures
2303 with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of
2304 rights made available for publication and any assurances of licenses to be made available, or the result
2305 of an attempt made to obtain a general license or permission for the use of such proprietary rights by
2306 implementors or users of this specification, can be obtained from the OASIS Executive Director.

2307 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
2308 applications, or other proprietary rights which may cover technology that may be required to implement
2309 this specification. Please address the information to the OASIS Executive Director.

2310 **Copyright © OASIS Open 2004. All Rights Reserved.**

2311 This document and translations of it may be copied and furnished to others, and derivative works that
2312 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published
2313 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright
2314 notice and this paragraph are included on all such copies and derivative works. However, this
2315 document itself does not be modified in any way, such as by removing the copyright notice or
2316 references to OASIS, except as needed for the purpose of developing OASIS specifications, in which
2317 case the procedures for copyrights defined in the OASIS Intellectual Property Rights document must be
2318 followed, or as required to translate it into languages other than English.

2319 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
2320 successors or assigns.

2321 This document and the information contained herein is provided on an "AS IS" basis and OASIS
2322 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
2323 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS
2324 OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR
2325 PURPOSE.