



OASIS ebXML RegRep Version 4.0 Part 1: Registry Information Model (ebRIM)

Committee Specification Draft 02

12 May 2011

Specification URIs:

This version:

<http://docs.oasis-open.org/regrep/regrep-core/v4.0/csd02/regrep-core-rim-v4.0-csd02.odt>
(Authoritative)
<http://docs.oasis-open.org/regrep/regrep-core/v4.0/csd02/regrep-core-rim-v4.0-csd02.pdf>
<http://docs.oasis-open.org/regrep/regrep-core/v4.0/csd02/regrep-core-rim-v4.0-csd02.html>

Previous version:

<http://docs.oasis-open.org/regrep/regrep-core/v4.0/csd01/regrep-core-rim-v4.0-csd01.odt>
(Authoritative)
<http://docs.oasis-open.org/regrep/regrep-core/v4.0/csd01/regrep-core-rim-v4.0-csd01.pdf>
<http://docs.oasis-open.org/regrep/regrep-core/v4.0/csd01/regrep-core-rim-v4.0-csd01.html>

Latest version:

<http://docs.oasis-open.org/regrep/regrep-core/v4.0/regrep-core-rim-v4.0.odt> (Authoritative)
<http://docs.oasis-open.org/regrep/regrep-core/v4.0/regrep-core-rim-v4.0.pdf>
<http://docs.oasis-open.org/regrep/regrep-core/v4.0/regrep-core-rim-v4.0.html>

Technical Committee:

OASIS ebXML Registry TC

Chairs:

Kathryn Breininger, Boeing
Farrukh Najmi, Wellfleet Software

Editors:

Farrukh Najmi, Wellfleet Software
Nikola Stojanovic, Individual

Related work:

This specification replaces or supersedes the [OASIS ebXML RegRep 3.0 specifications](#).

This specification consists of the following documents, schemas, and ontologies:

- [Part 0: Overview Document](#) - provides a global overview and description of all the other parts

- [Part 1: Registry Information Model \(ebRIM\)](#) (this document) - specifies the types of metadata and content that can be stored in an ebXML RegRep
- [Part 2: Services and Protocols \(ebRS\)](#) - specifies the services and protocols for ebXML RegRep
- [Part 3: XML Schema](#) - specifies the XML Schema for ebXML RegRep
- [Part 4: WSDL](#) - specifies the WSDL interface descriptions for ebXML RegRep
- [Part 5: XML Definitions](#) - specifies the canonical XML data for ebXML RegRep as well as example XML documents used in the specification

Declared XML namespaces:

See Part 0: Overview Document

Abstract:

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

A separate document, *OASIS ebXML RegRep Version 4.0 Part 2: Services and Protocols (ebRS)*, defines the services and protocols for an ebXML RegRep.

Status:

This document was last revised or approved by the [OASIS ebXML Registry TC](#) on the above date. The level of approval is also listed above. Check the "Latest Version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at <http://www.oasis-open.org/committees/regrep/>.

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

Citation format:

When referencing this specification the following citation format should be used:

[regrep-rim-v4.0] *OASIS ebXML RegRep Version 4.0 Part 1: Registry Information Model (ebRIM)*. 12 May 2011. OASIS Committee Specification Draft 02. <http://docs.oasis-open.org/regrep/regrep-core/v4.0/csd02/regrep-core-rim-v4.0-csd02.odt>.

Notices

Copyright © OASIS Open 2010-2011. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full [Policy](#) may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

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

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

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of [OASIS](#), the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <http://www.oasis-open.org/who/trademark.php> for above guidance.

Table of Contents

1	Introduction.....	9
1.1	Terminology.....	9
1.2	XML Schema.....	9
1.3	Information Model Types: Inheritance View.....	9
1.4	Extending ebRIM.....	10
1.5	Canonical ClassificationSchemes.....	11
1.6	Document Organization.....	11
2	Core Information Model.....	13
2.1	InternationalStringType.....	13
2.1.1	Syntax.....	13
2.1.2	Example.....	14
2.1.3	Description.....	14
2.2	LocalizedStringType.....	14
2.2.1	Syntax.....	14
2.2.2	Example.....	14
2.2.3	Description.....	14
2.3	ExtensibleObjectType.....	15
2.3.1	Syntax.....	15
2.3.2	Example.....	15
2.3.3	Description.....	15
2.4	SlotType.....	15
2.4.1	Syntax.....	16
2.4.2	Example.....	16
2.4.3	Description.....	16
2.5	ValueType.....	16
2.5.1	Syntax.....	16
2.5.2	Description.....	17
2.6	IdentifiableObjectType.....	17
2.6.1	Syntax.....	18
2.6.2	Example.....	18
2.6.3	Description.....	18
2.7	RegistryObjectType.....	18
2.7.1	Syntax.....	18
2.7.2	Description.....	19
2.8	VersionInfoType.....	21
2.8.1	Syntax.....	21
2.8.2	Example.....	21
2.8.3	Description.....	21
2.9	objectReferenceType.....	22
2.9.1	Syntax.....	22
2.9.2	Example.....	22
2.9.3	Description.....	22
2.10	ObjectRefType.....	24
2.10.1	Syntax.....	24
2.10.2	Description.....	24
2.11	DynamicObjectRefType.....	25
2.11.1	Syntax.....	25

2.11.2	Description.....	25
2.12	ExtrinsicObjectType.....	25
2.12.1	Syntax.....	25
2.12.2	Example.....	26
2.12.3	Description.....	26
2.13	CommentType.....	27
2.13.1	Syntax.....	27
2.13.2	Example.....	27
2.13.3	Description.....	27
2.14	RegistryPackageType.....	27
2.14.1	Syntax.....	28
2.14.2	Example.....	28
2.14.3	Description.....	29
2.15	ExternalIdentifierType.....	29
2.15.1	Syntax.....	29
2.15.2	Example.....	29
2.15.3	Description.....	30
2.16	ExternalLinkType.....	30
2.16.1	Syntax.....	30
2.16.2	Example.....	31
2.16.3	Description.....	31
3	Association Information Model.....	32
3.1	Source and Target Objects.....	32
3.2	Type of an Association.....	32
3.3	AssociationType.....	32
3.3.1	Syntax.....	32
3.3.2	Example.....	33
3.3.3	Description.....	33
3.4	Access Control.....	33
4	Classification Information Model.....	34
4.1	TaxonomyElementType.....	36
4.1.1	Syntax.....	36
4.1.2	Description.....	36
4.2	ClassificationSchemeType.....	37
4.2.1	Syntax.....	37
4.2.2	Example.....	37
4.2.3	Description.....	37
4.3	ClassificationNodeType.....	38
4.3.1	Syntax.....	38
4.3.2	Description.....	38
4.3.3	Canonical Path Syntax.....	39
4.4	ClassificationType.....	39
4.4.1	Syntax.....	40
4.4.2	Example.....	40
4.4.3	Description.....	40
5	Provenance Information Model.....	42
5.1	PostalAddressType.....	42
5.1.1	Syntax.....	42
5.1.2	Example.....	43
5.1.3	Description.....	43
5.2	TelephoneNumberType.....	43

5.2.1	Syntax	44
5.2.2	Example	44
5.2.3	Description	44
5.3	EmailAddressType	44
5.3.1	Syntax	45
5.3.2	Example	45
5.3.3	Description	45
5.4	PartyType	45
5.4.1	Syntax	45
5.4.2	Description	46
5.5	PersonType	46
5.5.1	Syntax	46
5.5.2	Example	46
5.5.3	Description	47
5.6	PersonNameType	47
5.6.1	Syntax	47
5.6.2	Example	47
5.6.3	Description	47
5.7	OrganizationType	48
5.7.1	Syntax	48
5.7.2	Example	48
5.7.3	Description	48
5.8	Associating Organization With Persons	49
5.9	Associating Organization With Organizations	49
5.10	Associating Organizations With RegistryObjects	49
6	Service Information Model	50
6.1	ServiceType	50
6.1.1	Syntax	50
6.1.2	Example	50
6.1.3	Description	51
6.2	ServiceEndpointType	51
6.2.1	Syntax	51
6.2.2	Example	51
6.2.3	Description	51
6.3	ServiceBindingType	52
6.3.1	Syntax	52
6.3.2	Example	52
6.3.3	Description	52
6.4	ServiceInterfaceType	52
6.4.1	Syntax	52
6.4.2	Example	53
6.4.3	Description	53
7	Query Information Model	54
7.1	QueryDefinitionType	54
7.1.1	Syntax	54
7.1.2	Example	55
7.1.3	Description	55
7.2	ParameterType	55
7.2.1	Syntax	55
7.2.2	Example	56
7.2.3	Description	56

7.3	QueryExpressionType.....	57
7.3.1	Syntax.....	57
7.3.2	Description.....	57
7.4	StringQueryExpressionType.....	58
7.4.1	Syntax.....	58
7.4.2	Example.....	58
7.4.3	Description.....	58
7.5	XMLQueryExpressionType.....	58
7.5.1	Syntax.....	58
7.5.2	Example.....	59
7.5.3	Description.....	59
7.6	QueryType.....	59
7.6.1	Syntax.....	59
7.6.2	Example.....	59
7.6.3	Description.....	60
8	Event Information Model.....	61
8.1	AuditableEventType.....	61
8.1.1	Syntax.....	62
8.1.2	Example.....	62
8.1.3	Description.....	62
8.2	ActionType.....	63
8.2.1	Syntax.....	63
8.2.2	Description.....	63
8.3	SubscriptionType.....	64
8.3.1	Syntax.....	64
8.3.2	Example.....	64
8.3.3	Description.....	65
8.4	DeliveryInfoType.....	65
8.4.1	Syntax.....	66
8.4.2	Description.....	66
8.5	NotificationType.....	67
8.5.1	Syntax.....	67
8.5.2	Example.....	67
8.5.3	Description.....	67
9	Federation Information Model.....	69
9.1	Federation Configuration.....	69
9.2	RegistryType.....	69
9.2.1	Syntax.....	70
9.2.2	Example.....	70
9.2.3	Description.....	70
9.3	FederationType.....	71
9.3.1	Syntax.....	71
9.3.2	Example.....	72
9.3.3	Description.....	72
10	Access Control Information Model.....	73
10.1	Defining an Access Control Policy.....	74
10.2	Assigning Access Control Policy to a RegistryObject.....	74
10.2.1	Default Access Control Policy for a RegistryObject.....	74
10.2.2	Access Control Policy Inheritance.....	75
10.2.3	Performance Implications.....	75
10.3	Defining a Contextual Role.....	75

10.3.1 RoleType.....	76
10.3.2 Example.....	76
10.3.3 Description.....	76
10.4 Assigning a Contextual Role to a Subject.....	76
10.5 Action Matching.....	77
10.5.1 Action Attribute: reference-source.....	78
10.5.2 Action Attribute: reference-source-attribute.....	78
10.6 Subject Matching.....	78
10.6.1 Matching Subjects By Id.....	79
10.6.2 Matching Subject By Role.....	79
10.7 Resource Matching.....	80
10.7.1 Matching a Resource By Id.....	81
10.7.2 Matching a Resource Using XPATH Expression.....	81
10.8 Canonical XACML Functions.....	82
10.8.1 Function AssociationExists.....	82
10.8.2 Function ClassificationNodeCompare.....	83
10.8.3 Function matches-role.....	83
10.9 Constraints on XACML Binding.....	84
10.10 Resolving Policy References.....	84

Illustration Index

Illustration 1: Information Model Inheritance View.....	10
Illustration 2: Core Information Model.....	13
Illustration 3: Association Example.....	32
Illustration 4: Classification Example.....	35
Illustration 5: Classification Information Model.....	36
Illustration 6: Provenance Information Model.....	42
Illustration 7: Service Information Model.....	50
Illustration 8: Query Information Model.....	54
Illustration 9: Event Information Model.....	61
Illustration 10: Federation Information Model.....	69
Illustration 11: Assigning Access Control Policy to a RegistryObject.....	74

Index of Tables

1 Introduction

All text is normative unless otherwise indicated.

This document specifies the ebXML RegRep registry information model. For a general overview of ebXML RegRep and other related parts of the specification please refer to Part 0 [regrep-overview-v4.0].

1.1 Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in IETF [RFC 2119].

1.2 XML Schema

The ebXML Registry Information Model is defined as an XML Schema in the file "/xsd/rim.xsd" in the specification distribution zip file. It defines the metadata types and their relationships within ebXML RegRep specifications.

1.3 Information Model Types: Inheritance View

The central type in the model is the RegistryObjectType. An instance of RegistryObjectType represents an ebRIM metadata object.

Illustration 1 shows the inheritance or "Is-A" relationships between the various types derived from RegistryObjectType in the information model. Note that it does not show the other types of relationships, such as "Has-A" relationships, as they will be presented in subsequent diagrams. The attributes and elements of each type are also not shown to conserve page space. Detailed description of attributes and elements of each type will be displayed in tabular form within the detailed description of each type.

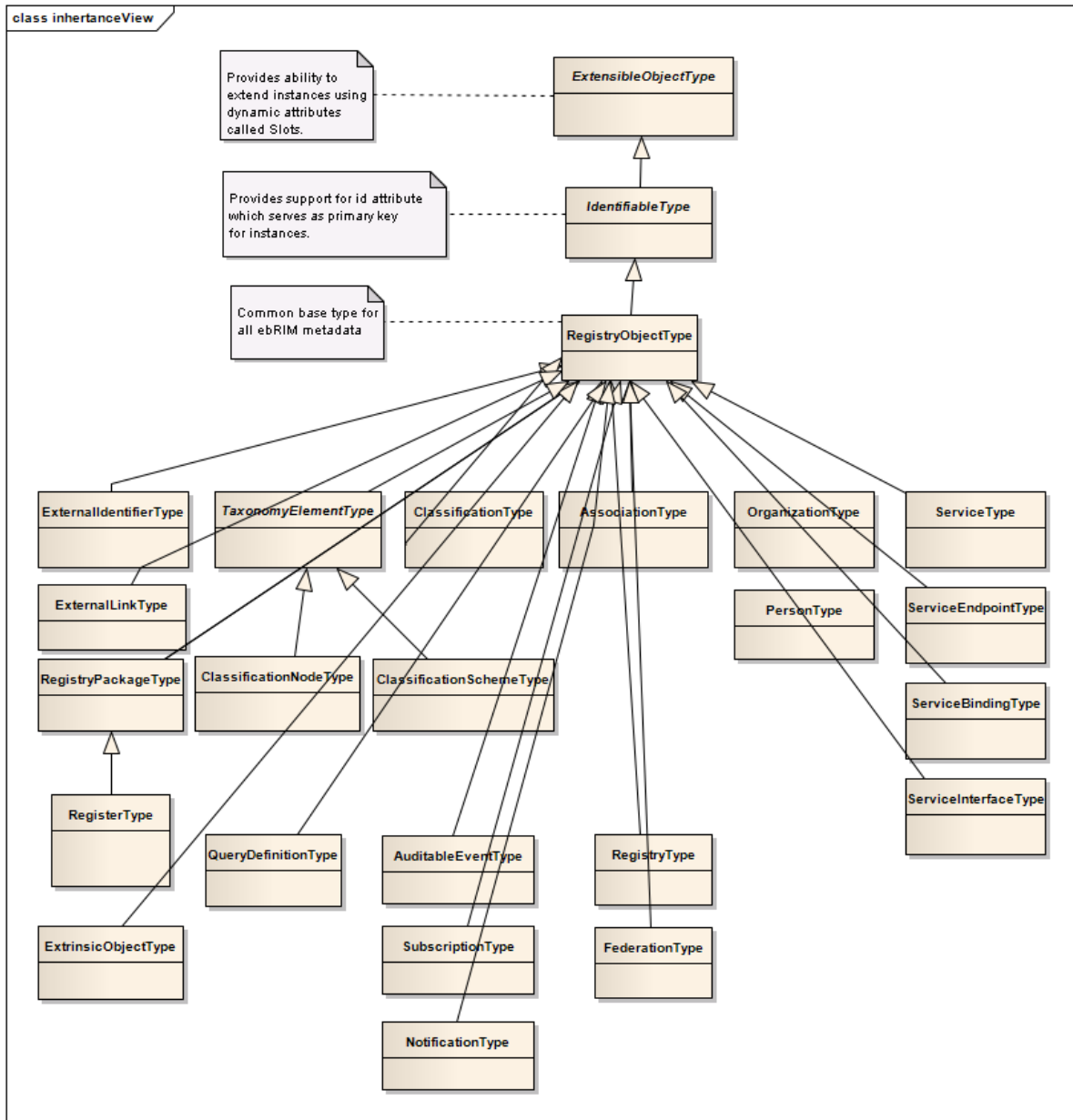


Illustration 1: Information Model Inheritance View

23 1.4 Extending ebRIM

24 The XML Schema for ebRIM uses XML Schema type substitution feature to allow use of schema type
25 extensions.

26 A deployment or profile specification of ebXML RegRep MAY define new types that extend the types
27 defined in this specification as long as the XML Schema for ebRIM supports such extension.

28 A server MAY support the schema type extensibility feature. The following requirements are defined for a
29 server that supports the schema type extensibility feature:

- 30 ● The server protocols as defined by [regrep-rs-v4.0] MUST support extended types in a manner
31 equivalent to pre-defined types. Specifically they MUST support submit, update, versioning and
32 removal of extended types derived directly or indirectly from RegistryObjectType
- 33 ● The server MUST be able to faithfully persist instances of extended types including all extension
34 attributes and elements without any information loss
- 35 ● The server MUST be able to faithfully return instances of extension types including extension
36 attributes and elements within a query response without any information loss
- 37 ● This specification does not prescribe how a server may support the addition of new extension
38 types to the server

39 1.5 Canonical ClassificationSchemes

40 ClassificationSchemes are defined in detail in the [Classification Information Model](#). They are used by the
41 specification for a wide variety of purposes within the ebXML RegRep specifications.

42 This specification uses several standard ClassificationSchemes referred to as *canonical*
43 *ClassificationSchemes*. The values defined within canonical ClassificationSchemes are defined using
44 standard ClassificationNodes that are referred to as *canonical ClassificationNodes*.

45 The directory “/xml/minDB” within the specification distribution zip file contains the canonical
46 ClassificationSchemes defined by the ebXML RegRep specifications. The canonical
47 ClassificationSchemes and ClassificationNodes are typically described using the rim:Description element
48 within these files.

49 These canonical ClassificationSchemes MUST be present in all conforming ebXML RegRep servers.
50 These Canonical ClassificationSchemes MAY be extended by adding additional ClassificationNodes.
51 However, a ClassificationNode defined normatively in the canonical ClassificationScheme definitions
52 MUST NOT be modified within a registry. In particular they MUST preserve their canonical id attributes in
53 all servers.

54 1.6 Document Organization

55 The types in the information model are presented in related groups as follows:

- 56 ● Core Information Model: Defines core metadata types in the model including the abstract base
57 types
- 58 ● Association Information Model: Defines types that enable objects to be associated with each other
- 59 ● Classification Information Model: Defines types that enable objects to be classified
- 60 ● Provenance Information Model: Defines types that enable the description of provenance or source
61 information about an object
- 62 ● Service Information Model: Defines types that enable service description
- 63 ● Query Information Model: Defines types that enable definition and invocation of queries
- 64 ● Event Information Model: Defines types that enable the event subscription and notification feature
65 defined in [regrep-rs-v4.0]
- 66 ● Federation Information Model: Defines types that enable the federated registries feature defined in
67 [regrep-rs-v4.0]
- 68 ● Access Control Information Model: Defines types that enable access control and authorization for
69 ebXML RegRep

70 The remainder of this document will describe each of the above related group of information model types
71 in a dedicated chapter named accordingly.

72 2 Core Information Model

73 The core information model is centered around the RegistryObjectType type as shown in figure below.
74 Each type will be defined in detail in subsequent section.

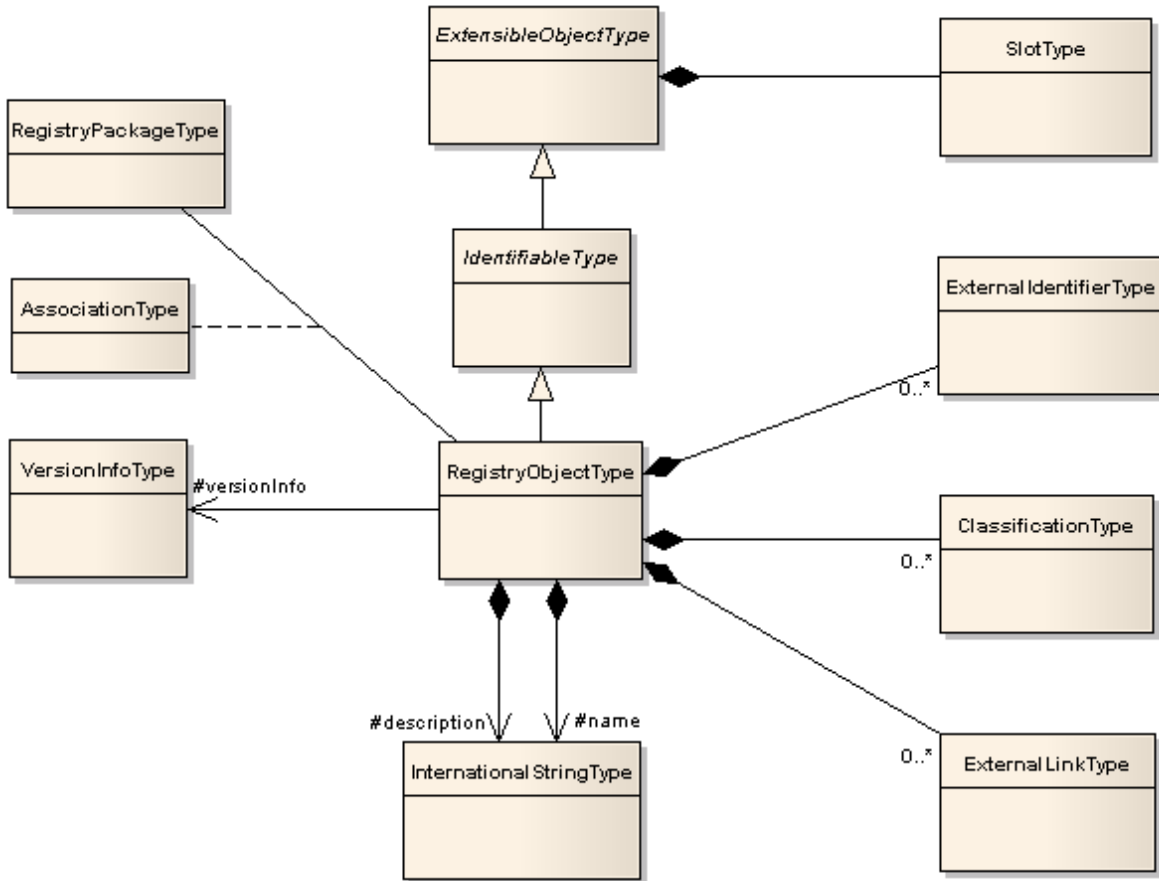


Illustration 2: Core Information Model

76 2.1 InternationalStringType

77 The InternationalStringType type is used throughout the schema whenever a textual value needs to be
78 represented in one or more local languages.

79 The InternationalStringType has a sequence of LocalizedString instances, where each LocalizedString
80 instance is specific to a particular locale.

81 2.1.1 Syntax

```
82 <complexType name="InternationalStringType">
83   <sequence>
84     <element name="LocalizedString" type="tns:LocalizedString"
85       minOccurs="0" maxOccurs="unbounded" />
86   </sequence>
87 </complexType>
```

88 2.1.2 Example

```
89 <rim:Name>  
90   <rim:LocalizedString  
91     xml:lang="en-US" value="freebXMLRegistry"/>  
92 </rim:Name>
```

93 2.1.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
LocalizedString	LocalizedStringType	0..*		Client	Yes

94

- 95 ● Element LocalizedString - An InternationalStringType instance MAY have zero or more
96 LocalizedString elements where each defines a string value within a specific local language

97 2.2 LocalizedStringType

98 This type allows the definition of a string value using the specified local language. It is used within the
99 InternationalStringType as the type of the LocalizedString sub-element. Note that the character set for all
100 LocalizedStringType instances in an XML document is defined by the charset attribute within the *Content-
101 Type* mime header for the XML document as shown in example below:

102

```
103 Content-Type: text/xml; charset="UTF-8"  
104
```

105 2.2.1 Syntax

```
106 <complexType name="LocalizedStringType">  
107   <attribute ref="xml:lang" default="en-US" use="optional"/>  
108   <attribute name="value" type="tns:FreeFormText" use="required"/>  
109 </complexType>
```

110 2.2.2 Example

```
111 <rim:LocalizedString  
112   xml:lang="en-US" value="freebXMLRegistry"/>
```

113 2.2.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
xml:lang	xs:language	0..1	en-US	Client	Yes
value	rim:FreeFormText	1		Client	Yes

114

- 115 ● Attribute xml:lang - Each LocalizedStringType instance MAY have a *xml:lang* attribute that
116 specifies the language used by that LocalizedStringType instance. The xml:lang attribute and
117 legal values for it are defined by [XML].

- Attribute value - Each LocalizedStringType instance MUST have a *value* attribute that specifies the string value used by that LocalizedStringType instance

2.3 ExtensibleObjectType

This type is the root type for most other types in rim.xsd. It allows extension properties called slots to be added to instances of this type using Slot sub-elements.

2.3.1 Syntax

```

124 <complexType name="ExtensibleObjectType" abstract="true">
125   <sequence>
126     <element name="Slot" type="tns:SlotType" minOccurs="0"
127       maxOccurs="unbounded"/>
128   </sequence>
129 </complexType>

```

2.3.2 Example

The following example shows how an OrganizationType instance which is of type ExtensibleObjectType MAY use Slot sub-elements to define a tax payer id for the organization.

```

134 <rim:RegistryObject xsi:type="rim:OrganizationType"
135   id="urn:freebxml:registry:Organization:freebXMLRegistry" ...>
136   <rim:Slot name="urn:foo:slot:taxPayerId">
137     <rim:SlotValue xsi:type="rim:StringValueType">
138       <rim:Value>1234567890</rim:Value>
139     </rim:SlotValue>
140   </rim:Slot>
141   ...
142 </rim:RegistryObject>

```

2.3.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
Slot	SlotType	0..*		Client	Yes

- Element Slot – Allows an extension property to be added to any ExtensibleObjectType instance

2.4 SlotType

Base Type: [ExtensibleObjectType](#)

The SlotType represents an extensible property for a RegistryObjectType instance . It can contain any type of information that may be represented in an XML document. It is an important extensibility mechanism with ebRIM.

A SlotType instance has a name and a value. The value is of type ValueType. ValueType is abstract and has several concrete sub-types defined within this specification.

Note that SlotType extends [ExtensibleObjectType](#) which means that a SlotType element may itself have SlotType sub-elements.

155 2.4.1 Syntax

```
156 <complexType name="SlotType">
157   <complexContent>
158     <extension base="tns:ExtensibleObjectType">
159       <sequence>
160         <element name="SlotValue" type="tns:ValueType"
161           minOccurs="0" maxOccurs="1"/>
162       </sequence>
163       <attribute name="name" type="tns:LongText" use="required"/>
164       <attribute name="type" type="tns:LongText" use="optional"/>
165     </extension>
166   </complexContent>
167 </complexType>
```

168 2.4.2 Example

169 The following example shows how a GML geometry value may be specified as a Slot.

```
170 <rim:Slot
171   name="geographicBoundingBox"
172   type="urn:ogc:def:dataType:ISO-19107:GM_Geometry">
173   <rim:SlotValue xsi:type="rim:AnyValueType">
174     <gml:Envelope srsName="urn:ogc:def:crs:OGC:2:WGS84">
175       <!--BB: POLYGON((0 0, 30 0, 30 30, 0 30, 0 0))-->
176       <gml:lowerCorner>0 0</gml:lowerCorner>
177       <gml:upperCorner>30 30</gml:upperCorner>
178     </gml:Envelope>
179   </rim:SlotValue>
180 </rim:Slot>
```

181 2.4.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
name	LongText	1		Client	Yes
SlotValue	ValueType	0..1		Client	Yes
type	LongText	0..1		Client	Yes

182

- 183 ● Attribute name – The name of this SlotType instance. The name of the slot MUST be unique
184 within the universe of slot names for all sibling slots within its parent object
- 185 ● Element SlotValue – This element is the container for the actual value for the SlotType instance
- 186 ● Attribute type – A string that specifies the type for the SlotType instance. The type may be used to
187 assign a category for the SlotType instance

188 2.5 ValueType

189 This type is abstract base type for the value of a SlotType instance.

190 2.5.1 Syntax

```
191 <complexType name="ValueType" abstract="true">
192 </complexType>
```


193 2.5.2 Description

194 The ValueType is an abstract base type that does not define any attributes or elements. This specification
195 defines several concrete sub-types that extend ValueType

- 196 ● AnyValueType – This concrete sub-type of ValueType is used as a container for any well-formed
197 XML element value in any namespace
- 198 ● BooleanValueType - This concrete sub-type of ValueType is used as a container for a boolean
199 value
- 200 ● CollectionValueType - This concrete sub-type of ValueType is used as a container for a collection
201 of values. It may be used to represent a SlotValue that is a collection of values where each value
202 is represented by a ValueType instance
 - 203 ○ Attribute collectionType – Defines the type of collection for the CollectionValueType. Must be
204 an objectReferenceType that references a ClassificationNode in the canonical
205 ClassificationScheme CollectionTypeScheme. A server MUST enforce the following
206 semantics associated with the following canonical collection types:
 - 207 ■ List – Server MUST maintain the order of the values in the collection
 - 208 ■ Set – Server MUST NOT allow duplicate values in the collection
 - 209 ■ Sorted Set – Server MUST NOT allow duplicate values in the collection and MUST
210 maintain a sort order according to the alphanumeric ordering of its elements according to
211 the default locale associated with the server
 - 212 ■ Bag – Server MUST allow duplicate values and MAY not maintain order of values
- 213 ● DateTimeValueType - This concrete sub-type of ValueType is used as a container for a dateTime
214 value
- 215 ● DurationValueType - This concrete sub-type of ValueType is used as a container for a duration
216 value
- 217 ● FloatValueType - This concrete sub-type of ValueType is used as a container for a float value
- 218 ● IntegerValueType - This concrete sub-type of ValueType is used as a container for an integer
219 value
- 220 ● InternationalStringValueType - This concrete sub-type of ValueType is used as a container for an
221 InternationalStringType value capable of holding strings in multiple locales
- 222 ● MapValueType - This concrete sub-type of ValueType is used as a container for a map value. A
223 map consists of Entry sub-elements where each Entry consists of an EntryKey and EntryValue
224 both of which are of type ValueType
- 225 ● SlotValueType – This concrete sub-type of ValueType is used as a container for a SlotType value
- 226 ● StringValueType – This concrete sub-type of ValueType is used as a container for a string value
- 227 ● VocabularyTermValueType - This concrete sub-type of ValueType is used as a container for a
228 VocabularyTermType value. It is used to reference a term in some externally defined coded
229 vocabulary (e.g. Dublin Core)

230 2.6 IdentifiableObjectType

231 **Base Type:** [ExtensibleObjectType](#)

232 This type extends ExtensibleObjectType and allows its instances to be uniquely identifiable by a unique id.

233 2.6.1 Syntax

```
234 <complexType name="IdentifiableType" abstract="true">
235   <complexContent>
236     <extension base="tns:ExtensibleObjectType">
237       <attribute name="id" type="string" use="required"/>
238     </extension>
239   </complexContent>
240 </complexType>
```

241 2.6.2 Example

```
242 <rim:RegistryObject xsi:type="rim:OrganizationType"
243   id="urn:freebxml:registry:Organization:freebXMLRegistry" ...>
244   ...
245 </rim:RegistryObject>
```

246 2.6.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
id	xs:string	1		Client	Yes

247

- 248 ● Attribute id – Specifies the unique identifier for an IdentifiableType instance.

249 2.7 RegistryObjectType

250 **Base Type:** [IdentifiableType](#)

251 This type extends IdentifiableObjectType and is the common base type for all *queryable* metadata
252 elements in ebRIM.

253 2.7.1 Syntax

```
254 <complexType name="RegistryObjectType">
255   <complexContent>
256     <extension base="tns:IdentifiableType">
257       <sequence>
258         <element name="Name" type="tns:InternationalStringType"
259           minOccurs="0" maxOccurs="1"/>
260         <element name="Description" type="tns:InternationalStringType"
261           minOccurs="0" maxOccurs="1"/>
262         <element name="VersionInfo" type="tns:VersionInfoType" minOccurs="0"
263           maxOccurs="1"/>
264         <element name="Classification" type="tns:ClassificationType"
265           minOccurs="0" maxOccurs="unbounded"/>
266         <element name="ExternalIdentifier" type="tns:ExternalIdentifierType"
267           minOccurs="0" maxOccurs="unbounded" />
268         <element name="ExternalLink" type="tns:ExternalLinkType"
269           minOccurs="0" maxOccurs="unbounded"/>
270       </sequence>
271       <attribute name="lid" type="string" use="optional"/>
272       <attribute name="objectType" type="tns:objectReferenceType"
273         use="optional"/>
274       <attribute name="owner" type="string" use="optional"/>
275       <attribute name="status" type="tns:objectReferenceType" use="optional"/>
276     </extension>
```

277
278

```
</complexContent>  
</complexType>
```

279 2.7.2 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
Classification	ClassificationType	0..*		Client	Yes
Description	InternationalStringType	0..1		Client	Yes
ExternalIdentifier	ExternalIdentifierType	0..*		Client	Yes
ExternalLink	ExternalLinkType	0..*		Client	Yes
lid	string	0..1		Client or Server	No
Name	InternationalStringType	0..1		Client	Yes
objectType	objectReferenceType	0..1		Client or Server	No
owner	string	0..1		Server	Yes
status	objectReferenceType	0..1		Server	Yes
VersionInfo	VersionInfoType	0..1		Server	No

280

- 281 ● Element Classification - A RegistryObjectType instance MAY have zero or more
282 ClassificationType instances that are composed within the RegistryObject. A ClassificationType
283 instance classify the RegistryObject using a value within a ClassificationScheme
- 284 ● Element Description - A RegistryObjectType instance MAY have textual description in a human
285 readable and user-friendly form. This element is of type InternationalStringType and therefor
286 capable of containing textual values in multiple local languages and character sets.
- 287 ● Element ExternalIdentifier - A RegistryObjectType instance MAY have zero or more
288 ExternalIdentifier instances that are composed within the RegistryObject. An ExternalIdentifier
289 instance represents an alternate identifier for the RegistryObject in addition to the identifier
290 specified by its id attribute value.
- 291 ● Attribute lid - A RegistryObjectType instance MUST have a lid (Logical Id) attribute. The lid is used
292 to refer to a logical RegistryObject in a version independent manner.
 - 293 ○ All versions of a RegistryObject MUST have the same value for the lid attribute. Note that this
294 is in contrast with the id attribute that MUST be unique for each version of the same logical
295 RegistryObject.
 - 296 ○ The lid attribute MUST be specified by the client when creating the original version of a
297 RegistryObject.
 - 298 ○ The lid attribute specified when submitting the original version of a RegistryObject MUST be
299 globally unique and MUST NOT be already in use as lid by another object.
- 300 ● Element Name - A RegistryObjectType instance MAY have a human readable name. The name
301 does not need to be unique with respect to other RegistryObjectType instances. This element is of

- 302 type InternationalStringType and therefor capable of containing textual values in multiple local
303 languages and character sets.
- 304 ● Attribute objectType - A RegistryObjectType instance has an *objectType* attribute.
- 305 ○ The value of the objectType attribute MUST be a reference to a ClassificationNode in the
306 canonical ObjectType ClassificationScheme.
- 307 ○ A server MUST support the object types as defined by the canonical ObjectType
308 ClassificationScheme. The canonical ObjectType ClassificationScheme may easily be
309 extended by adding additional ClassificationNodes to the canonical ObjectType
310 ClassificationScheme.
- 311 ○ The *objectType* attribute MUST be assigned by the server for all RegistryObjectType
312 instances that are not instances of ExtrinsicObjectType.
- 313 ○ The *objectType* attribute MAY be assigned by the client for all RegistryObjectType instances
314 that are instances of ExtrinsicObjectType
- 315 ○ If the client does not specify an objectType for an ExtrinsicObject then the server MUST set
316 its value to the id of the ClassificationNode representing ExtrinsicObject within the canonical
317 ObjectType ClassificationScheme.
- 318 ○ A server MUST set the correct objectType on a RegistryObject when returning it as a
319 response to a client request.
- 320 ● Attribute owner – Specifies the identifier associated with the registered user that owns the
321 RegistryObjectType instance. It is used for access control and may be referenced within custom
322 access control policies.
- 323 ● Attribute status - A RegistryObjectType instance MUST have a life cycle status indicator. The
324 status is assigned by the server. Profiles MAY define additional status values if needed as slots
325 on the RegistryObjectType instance. Such slots SHOULD have a type attribute with value
326 “urn:oasis:names:tc:ebxml-regrep:rim:Slot:type:status”.
- 327 ○ A server MUST set the correct status on a RegistryObject when returning it as a response to
328 a client request.
- 329 ○ A client SHOULD NOT set the status on a RegistryObject when submitting the object as this
330 is the responsibility of the server.
- 331 ○ A server MUST ignore the status on a RegistryObject when it is set by the client during
332 submission or update of the object.
- 333 ○ The value of the status attribute SHOULD be a reference to a ClassificationNode in the
334 canonical StatusType ClassificationScheme.
- 335 ○ A Registry MUST support the status types as defined by the StatusType
336 ClassificationScheme. The canonical StatusType ClassificationScheme MAY easily be
337 extended by adding additional ClassificationNodes to the canonical StatusType
338 ClassificationScheme.
- 339 ● Element VersionInfo - Provides information about the specific version of a RegistryObjectType
340 instance. The VersionInfo element is set by the server.
- 341 ○ A server MUST set a VersionInfo element for a RegistryObjectType instance. The
342 VersionInfo element MUST contain a versionName attribute whose value MUST be unique for
343 all versions of that logical RegistryObjectType.

344 2.8 VersionInfoType

345 This type represents information about a specific version of a RegistryObject or RepositoryItem. It is used
346 as type for the RegistryObjectType/VersionInfo and ExtrinsicObjectType/ContentVersionInfo elements in
347 the rim.xsd schema.

348 2.8.1 Syntax

```
349 <complexType name="VersionInfoType">  
350   <attribute name="versionName"  
351     type="tns:String16" use="optional" default="1.1"/>  
352   <attribute name="userVersionName" type="string" use="optional"/>  
353 </complexType>
```

354 2.8.2 Example

```
355 <rim:RegistryObject xsi:type="rim:OrganizationType" ...>  
356   ...  
357   <rim:VersionInfo versionName="1.1" userVersionName="1.1"/>  
358   ...  
359 </rim:RegistryObject>
```

360 2.8.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
userVersionName	LongText	0..1		Client	Yes
versionName	String16	0..1		Server	No

361

- 362 ● Attribute userVersionName - Represents a client-specified version name associated with the
363 VersionInfo for a specific RegistryObject version
 - 364 ○ A client MAY directly provide a value for the userVersionName attribute when submitting or
365 updating an object
 - 366 ○ A server MUST persist any client specified userVersionName for an object without altering it
367 in any form
- 368 ● Attribute versionName - Represents the registry assigned version name identifying the
369 VersionInfo for a specific RegistryObject version.
 - 370 ○ The value for this attribute SHOULD NOT be specified by the client
 - 371 ○ A server MAY silently ignore the value for this attribute if specified by the client
 - 372 ○ The value for this attribute MUST be automatically generated by the server and MUST be
373 defined for RegistryObjectType instances returned by server responses. The server is free to
374 choose any scheme for generating the value for this attribute as long as the value is uniquely
375 identifies a version for objects that have the same lid attribute value.

376 2.9 objectReferenceType

377 **Base Type:** xs:string

378 A RegistryObjectType instance typically has several references to other RegistryObjectType instances.
379 These references are represented by attributes of type rim:objectReferenceType within the XML Schema
380 for ebXML RegRep.

381 The RegistryObjectType instance that has a reference to another RegistryObjectType instance is referred
382 to as the *reference source* object. The RegistryObjectType instance that is being referenced is referred to
383 as the *reference target* object.

384 2.9.1 Syntax

```
385 <simpleType name="objectReferenceType">  
386   <restriction base="string"/>  
387 </simpleType>
```

388 2.9.2 Example

```
389 <rim:RegistryObject xsi:type="rim:OrganizationType"  
390   primaryContact="urn:acme:person:Danyal" ...>  
391   ...  
392 </rim:RegistryObject>
```

393

394 2.9.3 Description

395 Local and Remote References

396 The reference source and target objects MAY be in different ebXML RegRep servers. In such cases the
397 reference is referred to as a *remote reference*.

398 Static and Dynamic References

399 When a reference is fixed to a specific reference target it is referred to as a *static reference*. This
400 specification also supports a *dynamic reference* where the reference target is determined dynamically by a
401 query at the time the reference is resolved. Such a reference is referred to as a *dynamic reference*.

402 Both static and dynamic references may be to a local or remote object. Static references to local reference
403 targets are the most typical form of reference.

404 Encoding of objectReferenceType

405 A client MUST specify values for reference attributes of type objectReferenceType to be encoded as
406 described below:

- 407 ● A static reference to a local reference target SHOULD be encoded as the value of the id attribute
408 of the reference target.
409 The following example shows the reference attribute named primaryContact within Organization
410 element. Its value is the value of the id attribute of a Person element.

```
412 <rim:RegistryObject xsi:type="rim:OrganizationType"  
413   primaryContact="urn:acme:person:Danyal" ...>  
414   ...  
415 </rim:RegistryObject>  
416  
417 <rim:RegistryObject xsi:type="rim:PersonType" id="urn:acme:person:Danyal" ...>
```

```
418     ...
419 </rim:RegistryObject>
```

420

- 421 ● A dynamic reference to a local reference target SHOULD be encoded to contain the id of a
422 DynamicObjectRefType instance. The reference target is determined by the singleton result
423 returned by the Query within the DynamicObjectRef instance.

424

425 The following example shows the reference attribute named primaryContact within Organization
426 element. Its value is the value of the *id* attribute of a DynamicObjectRefType instance. The
427 DynamicObjectRefType instance has a Query that gets the latest version of the object identified
428 by the *lid* parameter of the Query. The query when invoked matches the latest version of the
429 Person object representing Danyal.

```
431 <rim:RegistryObject xsi:type="rim:OrganizationType"
432   primaryContact="urn:acme:dynamicRef:LatestVersionOfDanyal" ...>
433   ...
434   ...
435 </rim:RegistryObject>
436
437 <rim:ObjectRef xsi:type="rim:ObjectRefType"
438   id="urn:acme:dynamicRef:LatestVersionOfDanyal">
439   <rim:Query queryDefinition="urn:acme:QueryDefinition:FindLatestVersion">
440     <rim:Slot name="lid">
441       <rim:SlotValue xsi:type="rim:StringValueType">
442         <rim:Value>urn:acme:person:Danyal</rim:Value>
443       </rim:SlotValue>
444     </rim:Slot>
445   </rim:Query>
446 </rim:ObjectRef>
447
448 <rim:RegistryObject xsi:type="rim:PersonType"
449   lid="urn:acme:person:Danyal" id="urn:acme:person:Danyal:1.8"...>
450   <!-- latest version of object with lid "urn:acme:person:Danyal" -->
451   ...
452 </rim:RegistryObject>
```

453

- 454 ● A static or dynamic reference to a local reference target MAY be encoded to contain a Canonical
455 URL for the local object as defined by the REST binding in [regrep-rs-v4.0].
- 456 ● A static or dynamic reference to a remote reference target MUST be encoded to contain a
457 Canonical URL for the local object as defined by the REST binding in [regrep-rs-v4.0].

458

459 The following example shows the reference attribute named primaryContact within Organization
460 element. Its value is the HTTP GET URL for a remote PersonType instance. Note that the URL is
461 not encoded to handle special characters for sake of clarity.

```
463 <!-- Following object is in local server -->
464 <rim:RegistryObject xsi:type="rim:OrganizationType"
465   primaryContact="http://www.remoteRegistry.com/query?
466   id=urn:remoteServer:person:Danyal" ...>
467   ...
468   ...
469 </rim:RegistryObject>
470
471 <!-- Following object is in a remote server -->
472 <rim:RegistryObject xsi:type="rim:PersonType"
473   id="urn:remoteServer:person:Danyal" ...>
```

474 ...
475 </rim:RegistryObject>

476

477 2.10 ObjectRefType

478 **Base Type:** [ExtensibleObjectType](#)

479 This type represents an object reference as does the objectReferenceType. However, the two are used in
480 different situations. The objectReferenceType is used as the type for all reference attributes in ebRIM. The
481 ObjectRefType is used as type for elements rather than attributes. This type is used when there is a need
482 to have multiple object references within a schema type. An example of this is the ObjectRefList element
483 which is used in several places in the schema where a list of references to RegistryObjectType instances
484 are needed.

485 2.10.1 Syntax

```
486 <complexType name="ObjectRefType">  
487   <complexContent>  
488     <extension base="tns:ExtensibleObjectType">  
489       <attribute name="id" type="tns:objectReferenceType" use="required"/>  
490     </extension>  
491   </complexContent>  
492 </complexType>  
493  
494 <complexType name="ObjectRefListType">  
495   <sequence>  
496     <element name="ObjectRef"  
497       type="tns:ObjectRefType" minOccurs="0" maxOccurs="unbounded"/>  
498   </sequence>  
499 </complexType>  
500 <element name="ObjectRefList" type="tns:ObjectRefListType"/>
```

501 2.10.2 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
id	objectReferenceType	1		Client	Yes

502

- 503 ● Attribute *id* - Every ObjectRef instance MUST have an *id* attribute. The *id* attribute MUST contain
504 the value of the *id* attribute of the RegistryObject being referenced.

505 2.11 DynamicObjectRefType

506 **Base Type:** [ObjectRefType](#)

507 This type represents a dynamic object reference. It extends the ObjectRefType and add a Query sub-
508 element. This query is used to determine the reference target at the time the reference is resolved.

509 2.11.1 Syntax

```
510 <complexType name="DynamicObjectRefType">  
511   <complexContent>
```



```

512     <extension base="tns:ObjectRefType">
513         <sequence>
514             <element name="Query" type="tns:QueryType"
515                 minOccurs="1" maxOccurs="1"/>
516         </sequence>
517     </extension>
518 </complexContent>
519 </complexType>

```

520 2.11.2 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
Query	QueryType	1		Client	Yes

521

- 522 ● Element Query – Specifies the query that MUST be invoked in order to determine the reference
- 523 target.
- 524 ○ This query MUST match zero or one RegistryObjectType instances.
- 525 ○ When the query matches zero RegistryObjectType instances, the dynamic object reference is
- 526 considered to be unresolved.
- 527 ○ A server MUST return a ConfigurationException fault message if the query matches more
- 528 than 1 RegistryObjectType instances.

529 2.12 ExtrinsicObjectType

530 **Base Type:** [RegistryObjectType](#)

531 This type is a common base type for new extended types defined by profiles of ebRIM or by clients. The
532 ExtrinsicObjectType also allows arbitrary content to be associated with it. Such arbitrary content is referred
533 to as a Repository Item.

534 2.12.1 Syntax

```

535 <complexType name="ExtrinsicObjectType">
536     <complexContent>
537         <extension base="tns:RegistryObjectType">
538             <sequence>
539                 <element name="ContentVersionInfo" type="tns:VersionInfoType"
540                     minOccurs="0" maxOccurs="1"/>
541                 <choice minOccurs="0" maxOccurs="1">
542                     <element name="RepositoryItemRef" type="tns:SimpleLinkType"/>
543                     <element name="RepositoryItem"
544                         xmime:expectedContentTypes="*/*" type="base64Binary">
545                 </choice>
546             </sequence>
547             <attribute name="mimeType" type="tns:LongText" use="optional" />
548         </extension>
549     </complexContent>
550 </complexType>

```

551 2.12.2 Example

```

552 <rim:RegistryObject xsi:type="rim:ExtrinsicObjectType" mimeType="text/xml"
553     objectType="urn:freebxml:registry:sample:profile:cpp:objectType:c++:CPP"

```

```

554     lid="urn:freebxml:registry:sample:profile:cpp:instance:cpp1"
555     id="urn:freebxml:registry:sample:profile:cpp:instance:cpp1" >
556     <ContentVersionInfo versionName="311" userVersionName="1.1"/>
557     <RepositoryItem>...binary encoding of repository item</RepositoryItem>
558 </rim:RegistryObject>

```

559 2.12.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
ContentVersionInfo	VersionInfoType	0..1		Server	No
mimeType	LongText	0..1	application/octet-stream	Client	No
RepositoryItem	xs:base64Binary	0..1		Client	Yes
RepositoryItemRef	SimpleLinkType	0..1		Client	No

560

- 561 ● Element ContentVersionInfo - Provides information about the specific version of a RepositoryItem
562 that is associated with an ExtrinsicObjectType instance. The ContentVersionInfo element is set by
563 the server.
 - 564 ○ A server MUST NOT set a ContentVersionInfo element for an ExtrinsicObjectType instance
565 that does not have a RepositoryItem.
 - 566 ○ A server MUST set a ContentVersionInfo element for an ExtrinsicObjectType instance that
567 has a RepositoryItem. The ContentVersionInfo element MUST contain a versionName
568 attribute whose value MUST be unique for all versions of that RepositoryItem.
- 569 ● Attribute mimeType - An ExtrinsicObjectType instance MAY have a mimeType attribute defined.
570 The mimeType provides information on the type of repository item cataloged by the
571 ExtrinsicObject instance. The value of this attribute SHOULD be a registered MIME media type at
572 <http://www.iana.org/assignments/media-types>.
- 573 ● Element repositoryItem – Provides a base64 binary encoded representation of the repository item
574 associated with the ExtrinsicObjectType instance (if any).
- 575 ● Element repositoryItemRef – This element MAY be specified as an alternative to the
576 repositoryItem element. Its type is SimpleLinkType. It uses xlink:simpleAttrs to specify a reference
577 to a file on the client's local file system. This provides client libraries an alternative way to specify
578 local files as repository item. The client library MUST convert a repositoryItemRef element to a
579 repositoryItem element prior to submitting it to the server.

580 2.13 CommentType

581 **Extends:** [ExtrinsicObjectType](#)

582 This type represents a comment that may be associated with a RegistryObjectType instance. A comment
583 associated with a RegistryObject models the familiar yellow [POST-IT note](#) metaphor used in attaching
584 comments to paper documents.

585 2.13.1 Syntax

```

586 <complexType name="CommentType">
587   <complexContent>
588     <extension base="tns:ExtrinsicObjectType">
589     </extension>
590   </complexContent>

```

591 </complexType>

592 2.13.2 Example

```
593 <rim:RegistryObject xsi:type="rim:CommentType"
594   lid="urn:freebxml:registry:sample:comment1"
595   id="urn:freebxml:registry:sample:comment1" >
596   <rim:Description>
597     <rim:LocalizedString
598       xml:lang="en-US" value="This change request is rejected because it is
599   too complex a change."/>
600   </rim:Description>
601 </rim:RegistryObject>
```

602 2.13.3 Description

603 No new attributes or elements are added by this type. The following requirements are defined for this type:

- 604 ● An authorized client MAY attach one or more comments to any RegistryObjectType instance
605 using an Association between the RegistryObjectType instance and the CommentType instance
- 606 ○ Since a CommentType is itself a RegistryObjectType, a client MAY attach one or more
607 comments to any CommentType instance
- 608 ● The type of the Association MUST reference the canonical HasComment ClassificationNode
609 within the Canonical AssociationType ClassificationScheme
- 610 ● The sourceObject of the Association MUST be the RegistryObjectType instance
- 611 ● The targetObject of the Association MUST be the CommentType instance

612 2.14 RegistryPackageType

613 **Extends:** RegistryObjectType

614 This type allows for grouping of related RegistryObjectType instances. It serves a similar role as a folder in
615 the familiar file-folder metaphor available in most operating systems.

- 616 ● A RegistryObjectType instance MAY be a member of multiple RegistryPackageType instances.
- 617 ● A RegistryPackageType instance MAY have multiple RegistryObjectType instances as its
618 members.
- 619 ● Membership of a RegistryObjectType instance in a RegistryPackageType instance is established
620 via an AssociationType instance where the type attribute references the canonical "HasMember"
621 AssociationType within the canonical AssociationTypeScheme ClassificationScheme.
- 622 ● As a convenience, the RegistryPackageType allows a RegistryObjectList to be specified by the
623 client as a sub-element during submission of a RegistryPackage. The RegistryObjectList contains
624 the set of RegistryObjectType instances that are members of the RegistryPackageType instance.

625 2.14.1 Syntax

```
626 <complexType name="RegistryPackageType">
627   <complexContent>
628     <extension base="tns:RegistryObjectType">
629       <sequence>
630         <element name="RegistryObjectList" type="tns:RegistryObjectListType"
631           minOccurs="0" maxOccurs="1"/>

```

```

632     </sequence>
633     </extension>
634     </complexContent>
635 </complexType>

```

2.14.2 Example

The following example shows the use of a RegistryObjectList to specify the members of a RegistryPackageType instance during submission.

```

639 <rim:RegistryObject xsi:type="rim:RegistryPackageType"
640     id="urn:acme:RegistryPackage:photos" ...>
641     ...
642     <rim:RegistryObjectList>
643     <rim:RegistryObject xsi:type="rim:ExtrinsicObjectType"
644     mimeType="image/jpeg" id="urn:acme:RegistryPackage:photos:summer-
645     2008:wellfleet-beach.jpg"
646     <repositoryItem>
647     ..binary encoding of photo repository item
648     </repositoryItem>
649     </rim:RegistryObject>
650     </rim:RegistryObjectList>
651 </rim:RegistryObject>

```

The following example shows the equivalent syntax for representing the membership relationship between a RegistryPackage and its members. This representation uses “HasMember” AssociationType instances to establish the membership relationship.

```

657 <rim:RegistryObject xsi:type="rim:RegistryPackageType"
658     id="urn:acme:RegistryPackage:photos" .../>
659
660 <rim:RegistryObject xsi:type="rim:ExtrinsicObjectType" mimeType="image/jpeg"
661     id="urn:acme:RegistryPackage:photos:summer-2008:wellfleet-beach.jpg"
662     <repositoryItem>
663     ..binary encoding of photo repository item
664     </repositoryItem>
665 </rim:RegistryObject>
666
667 <Association
668     sourceObject="urn:acme:RegistryPackage:photos"
669     targetObject="urn:acme:RegistryPackage:photos:summer-2008:wellfleet-
670     beach.jpg"
671     type="urn:oasis:names:tc:ebxml-regrep:AssociationType:HasMember"/>
672

```

2.14.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
RegistryObjectList	RegistryObjectList Type	0..1		Client	Yes

- Element RegistryObjectList – This element allows clients to specify members of the RegistryPackage instance using a simpler alternative to “HasMember” AssociationType instances.
 - A server MUST replace the RegistryObjectList with AssociationType instances such that each RegistryObjectType instance is replaced with an AssociationType instance with type

679 "urn:oasis:names:tc:ebxml-regrep:AssociationType:HasMember", with sourceObject
 680 specifying the id of the RegistryPackage instance and with targetObject specifying the id of
 681 the RegistryObjectType instance

682 2.15 ExternalIdentifierType

683 **Base Type:** RegistryObjectType

684 This type allows any number of additional identifiers to be specified for a RegistryObjectType instance.
 685 The identifier value is defined using the *value* attribute within the context of a ClassificationScheme
 686 referenced via the *identificationScheme* attribute.

687 2.15.1 Syntax

```
688 <complexType name="ExternalIdentifierType">
689   <complexContent>
690     <extension base="tns:RegistryObjectType">
691       <attribute name="registryObject"
692         type="tns:objectReferenceType" use="optional"/>
693       <attribute name="identificationScheme"
694         type="tns:objectReferenceType" use="required"/>
695       <attribute name="value" type="tns:LongText" use="required"/>
696     </extension>
697   </complexContent>
698 </complexType>
```

699 2.15.2 Example

700 The following examples shows an Organization instance with its tax payer id specified using an
 701 ExternalIdentifierType instance.

```
702 <rim:RegistryObject xsi:type="rim:OrganizationType" ...>
703   ...
704   <rim:ExternalIdentifier ...
705     identificationScheme="urn:acme:ClassificationScheme:TaxPayerId"
706     value="1234567890"/>
707   </rim:ExternalIdentifier>
708   ...
709 </rim:RegistryObject>
```

710 2.15.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
identificationScheme	objectReferenceType	1		Client	Yes
registryObject	objectReferenceType	0..1		Client	No
value	LongText	1		Client	Yes

711

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

- 716 ● Attribute registryObject - Each ExternalIdentifier instance MAY have a *registryObject* attribute
717 specified. This attribute references the parent RegistryObjectType instance for which this is an
718 ExternalIdentifier.
- 719 ○ This attribute MUST be specified when a client submits an ExternalIdentifier separately from
720 its parent RegistryObjectType instance
- 721 ○ This attribute MAY be unspecified when a client submits an ExternalIdentifier as a sub-
722 element of its parent RegistryObjectType instance. In such cases the server MUST set this
723 attributes value to the value of the id attribute of the parent RegistryObjectType instance.
- 724 ○ Attribute value - Each ExternalIdentifier instance MUST have a *value* attribute that provides
725 the identifier value for this ExternalIdentifier (e.g., the tax payer id in example above).

726 2.16 ExternalLinkType

727 **Base Type:** RegistryObjectType

728 This type allows a link to external content to be added to a RegistryObjectType instance.

729 2.16.1 Syntax

```
730 <complexType name="ExternalLinkType">
731   <complexContent>
732     <extension base="tns:RegistryObjectType">
733       <sequence>
734         <element name="ExternalRef"
735           type="tns:SimpleLinkType" minOccurs="1" maxOccurs="1"/>
736       </sequence>
737       <attribute name="registryObject"
738         type="tns:objectReferenceType" use="optional"/>
739     </extension>
740   </complexContent>
741 </complexType>
```

742 2.16.2 Example

743 The following examples shows an Organization instance with an ExternalLink that links to its web site URL
744 via its ExternalRef sub-element.

```
745 <rim:RegistryObject xsi:type="rim:OrganizationType" ...>
746   ...
747   <rim:ExternalLink ...
748     objectType="urn:oasis:names:tc:ebxml-
749   regrep:Objectype:RegistryObject:ExtrinsicObject:XML:WSDL"
750     mimeType="text/xml"/>
751     <ExternalRef xlink:href="http://www.acme.com"/>
752   </rim:ExternalLink>
753   ...
754 </rim:RegistryObject>
```

755 2.16.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
ExternalRef	SimpleLinkType	1		Client	Yes
registryObject	objectReferenceType	0..1		Client or Server	No

756

757
758
759

- Element ExternalRef - Each ExternalLink instance MUST have an ExternalRef sub-element defined. This element provides a URI to the external resource pointed to by this ExternalLink instance.

760
761
762
763

- Attribute registryObject – references the parent RegistryObjectType instance within which the ExternalLinkType instance is composed. The value MUST be provided by client when an ExternalLink is submitted separate from its parent object. The value MUST be set by the server if the ExternalLink is submitted as part of the submission of its parent object.

3 Association Information Model

764

765 A RegistryObjectType instance MAY be associated or related with zero or more RegistryObjectType
766 instances. The information model defines the AssociationType type, an instance of which MAY be used to
767 associate any two RegistryObjectType instances. It also defines an Association element for that type.

768 In the example below, an AssociationType instance with type "...Supercedes" is used to indicate that the
769 NAICS2001 ClassificationScheme supercedes the NAICS1997 ClassificationScheme.

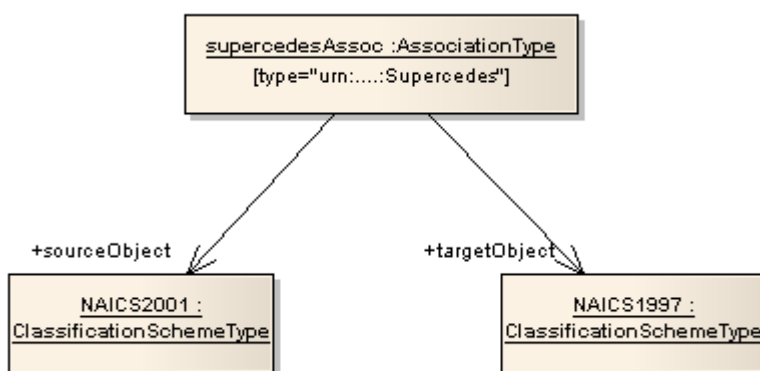


Illustration 3: Association Example

3.1 Source and Target Objects

771

772 An AssociationType instance represents an association between a source RegistryObjectType instance
773 and a target RegistryObjectType instance. These are referred to as *sourceObject* and *targetObject* for the
774 AssociationType instance. It is important which object is the sourceObject and which is the targetObject
775 as it determines the directional semantics of an Association.

3.2 Type of an Association

776

777 An AssociationType instance MUST have a type attribute that identifies the type of that association. The
778 value of this attribute is typically the id of a ClassificationNode under the canonical AssociationType
779 ClassificationScheme.

3.3 AssociationType

780

781 **Base Type:** RegistryObjectType

3.3.1 Syntax

782

```
783 <complexType name="AssociationType">
784   <complexContent>
785     <extension base="tns:RegistryObjectType">
786       <attribute name="type"
787         type="tns:objectReferenceType" use="required"/>
788       <attribute name="sourceObject"
789         type="tns:objectReferenceType" use="required"/>
790       <attribute name="targetObject"
791         type="tns:objectReferenceType" use="required"/>
792     </extension>

```



```
793     </complexContent>
794 </complexType>
```

795 3.3.2 Example

796 The following examples shows an Organization instance that has an “OffersService” association with a
797 Service that it offers.

```
798 <rim:RegistryObject xsi:type="rim:OrganizationType"
799   id="urn:acme:Organization:acme-inc" ... />
800 <rim:RegistryObject xsi:type="rim:ServiceType"
801   id="urn:acme:Service:stock-quote" ... />
802 <rim:RegistryObject xsi:type="rim:AssociationType"
803   id="urn:acme:Association:acme-example-relationship"
804   sourceObject="urn:acme:Organization:acme-inc"
805   targetObject="urn:acme:Service:stock-quote"
806   type="urn:oasis:names:tc:ebxml-regrep:AssociationType:OffersService" .../>
```

807 3.3.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
sourceObject	objectReferenceType	1		Client	Yes
targetObject	objectReferenceType	1		Client	Yes
type	objectReferenceType	1		Client	Yes

808

- 809 ● Attribute sourceObject - Each Association MUST have a *sourceObject* attribute that references
810 the RegistryObjectType instance that is the source of that Association.
- 811 ● Attribute targetObject - Each Association MUST have a *targetObject* attribute that references the
812 RegistryObjectType instance that is the target of that Association.
- 813 ● Attribute type - Each Association MUST have a *type* attribute that identifies the type of that
814 association.
 - 815 ○ The value of the type attribute MUST be a reference to a ClassificationNode within the
816 canonical AssociationType ClassificationScheme.
 - 817 ○ A server MUST support the canonical association types as defined by the canonical
818 AssociationType ClassificationScheme. Deployments and profiles may extend the canonical
819 AssociationType ClassificationScheme by adding additional ClassificationNodes to it.

820 3.4 Access Control

821 A client MAY create an AssociationType instance between *any* two RegistryObjectType instances
822 assuming the access control policies associated with the source and target object permit the client to
823 create a reference to them. The default access control policy permits any client to create a reference to an
824 object.

4 Classification Information Model

825

826 The ebRIM information model supports classification of RegistryObjectType instances using values
827 defined by a taxonomy or controlled vocabulary. A taxonomy is represented in ebRIM by the
828 ClassificationSchemeType type. Values in a taxonomy are represented by the ClassificationNode type. A
829 classification instance is represented in ebRIM by the ClassificationType type.

830 This specification specifies a set of canonical ClassificationSchemes. Deployments and profiles MAY
831 extend these canonical ClassificationSchemes by adding additional ClassificationNodes to them. They
832 MAY also define new ClassificationSchemes. A RegistryObjectType instance MAY be classified using *any*
833 ClassificationNode in *any* ClassificationScheme supported by the server. A RegistryObjectType instance
834 MAY have any number of classifications defined for it.

835 A general ClassificationScheme can be viewed as a tree structure where the ClassificationScheme is the
836 root and ClassificationNodes are either intermediate or leaf nodes in the tree.

837 Illustration 4 below shows RegistryObjectType instances representing Organizations as grey boxes. Each
838 Organization represents an automobile manufacturer. Organization is classified by the ClassificationNode
839 named "Automotive" under the ClassificationScheme instance with name "IndustryScheme". Furthermore,
840 the US Automobile manufacturers are classified by the "US" ClassificationNode under the
841 ClassificationScheme with name "GeographyScheme". Similarly, a European automobile manufacturer is
842 classified by the "Europe" ClassificationNode under the ClassificationScheme with name
843 "GeographyScheme".

844 The example shows how a RegistryObject may be classified by multiple ClassificationNodeType instances
845 under multiple ClassificationScheme instances (e.g., IndustryScheme, GeographyScheme).

846

847

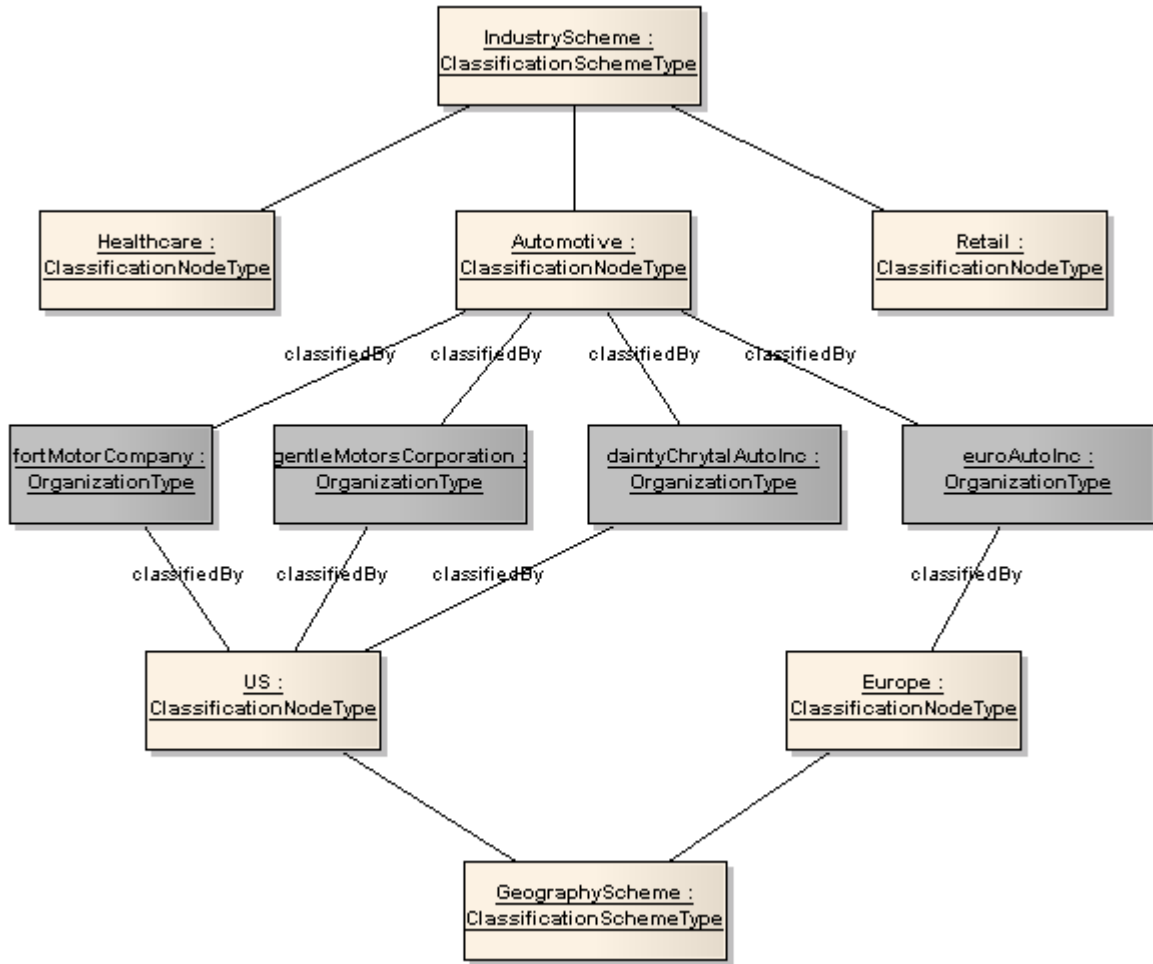


Illustration 4: Classification Example

849 Illustration 5 shows the Classification information model.

850

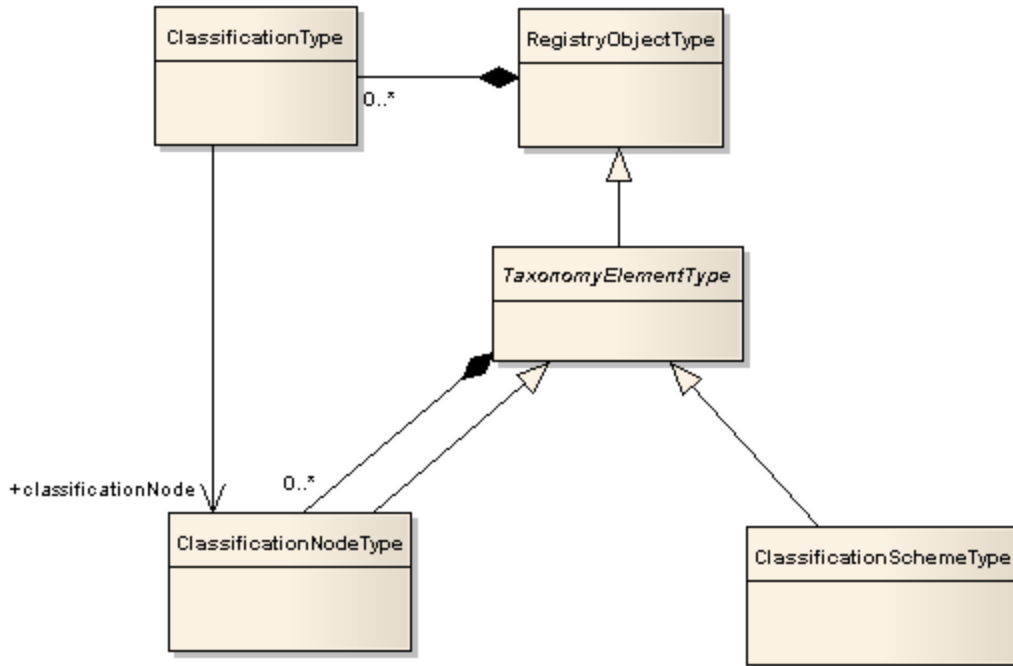


Illustration 5: Classification Information Model

852 4.1 TaxonomyElementType

853 **Base Type:** RegistryObjectType

854 This abstract type is the common base type for ClassificationSchemeType and ClassificationNodeType.

855 4.1.1 Syntax

```

856 <complexType name="TaxonomyElementType" abstract="true">
857   <complexContent>
858     <extension base="tns:RegistryObjectType">
859       <sequence>
860         <element name="ClassificationNode" type="tns:ClassificationNodeType"
861           minOccurs="0" maxOccurs="unbounded"/>
862       </sequence>
863     </extension>
864   </complexContent>
865 </complexType>

```

866 4.1.2 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
Classification Node	ClassificationNodeType	0..*		Client	Yes

867

- 868 ● Element ClassificationNode – This element represents a ClassificationNode child of a parent
869 TaxonomyElementType instance. A TaxonomyElementType instance MAY have any number of
870 ClassificationNode child elements.

871 4.2 ClassificationSchemeType

872 **Base Type:** [TaxonomyElementType](#)

873 A ClassificationScheme instance represents a taxonomy.

874 The taxonomy hierarchy may be defined internally to the server using instances of ClassificationNodeType
875 type, or it may be defined externally to the server, in which case the structure and values of the taxonomy
876 elements are not known to the Registry.

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

879 4.2.1 Syntax

```
880 <complexType name="ClassificationSchemeType">  
881   <complexContent>  
882     <extension base="tns:TaxonomyElementType">  
883       <attribute name="isInternal" type="boolean" use="required"/>  
884       <attribute name="nodeType"  
885         type="tns:objectReferenceType" use="required"/>  
886     </extension>  
887   </complexContent>  
888 </complexType>
```

889 4.2.2 Example

890 The following examples shows a ClassificationScheme representing gender values.

```
891 <rim:RegistryObject xsi:type="rim:ClassificationSchemeType"  
892   id="urn:acme:GenderScheme" isInternal="true"  
893   nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode" ...>  
894   <Name>  
895     <LocalizedString value="GenderScheme"/>  
896   </Name>  
897   <rim:ClassificationNode id="urn:acme:Gender:Male" code="Male" .../>  
898   <rim:ClassificationNode id="urn:acme:Gender:Female" code="Female" .../>  
899   <rim:ClassificationNode id="urn:acme:Gender:Other" code="Other" .../>  
900 </rim:RegistryObject>
```

901 4.2.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
isInternal	xs:boolean	1		Client	No
nodeType	objectReferenceType	1		Client	No

902

- 903 ● Attribute isInternal - When submitting a ClassificationSchemeType instance the client MUST
904 declare whether the ClassificationSchemeType instance represents an internal or an external
905 taxonomy. This allows the server to validate the subsequent submissions of
906 ClassificationNodeType and ClassificationType instances in order to maintain the type of
907 ClassificationScheme consistent throughout its lifecycle.
- 908 ● Attribute nodeType - When submitting a ClassificationScheme instance the client MUST declare
909 the structure of taxonomy nodes within the ClassificationScheme via the nodeType attribute. The
910 value of the nodeType attribute MUST be a reference to a ClassificationNodeType instance within
911 the canonical NodeType ClassificationScheme. A server MUST support the node types as defined
912 by the canonical NodeType ClassificationScheme. The canonical NodeType

913 ClassificationScheme MAY easily be extended by adding additional ClassificationNodes to it.
 914
 915 The following table lists the canonical ClassificationNode defined as values for the NodeType
 916 ClassificationScheme:

917

Name	Description
UniqueCode	Indicates that the code for each ClassificationNode in the ClassificationScheme is unique within the scope of the ClassificationScheme
EmbeddedPath	Indicates that the code assigned to each node of the taxonomy also encodes its path.
NonUniqueCode	Indicates that the code for each ClassificationNode in the ClassificationScheme is not unique within the scope of the ClassificationScheme. For example, in a geography taxonomy Moscow could be under both Russia and the USA, where there are five cities of that name in different states.

918

919 4.3 ClassificationNodeType

920 **Base Type:** [TaxonomyElementType](#)

921 ClassificationNodeType instances are used to define values for a taxonomy represented by
 922 ClassificationSchemeType instance.

923 4.3.1 Syntax

```

924 <complexType name="ClassificationNodeType">
925   <complexContent>
926     <extension base="tns:TaxonomyElementType">
927       <attribute name="parent" type="tns:objectReferenceType" use="optional"/>
928       <attribute name="path" type="string" use="optional"/>
929       <attribute name="code" type="tns:LongText" use="required"/>
930     </extension>
931   </complexContent>
932 </complexType>
  
```

933 4.3.2 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
code	LongText	1		Client	No
parent	objectReferenceType	0..1		Client	No
path	xs:string	0..1		Registry	No

934

- 935 ● Attribute code - A ClassificationNodeType instance MUST have a *code* attribute. The code
 936 attribute contains a code that represents a value within a ClassificationScheme.
- 937 ○ The code attribute of a ClassificationNodeType instance MUST be unique with respect to all
 938 sibling ClassificationNodes that are immediate children of the same parent
 939 TaxonomyElementType instance.

- 940 ● Attribute *parent* - A ClassificationNodeType instance MAY have a *parent* attribute. The parent
941 attribute references the parent TaxonomyElementType instance. This is either another
942 ClassificationNodeType instance or the ClassificationSchemeType instance.
- 943 ● Attribute *path* - A ClassificationNodeType instance MAY have a *path* attribute. The path attribute
944 represents a hierarchical path from the root ClassificationSchemeType to the
945 ClassificationNodeType instance. The [syntax of the path attribute value](#) is defined in 4.3.3.
 - 946 ○ A server MUST set the path attribute for any ClassificationNodeType instance when it is
947 submitted by a client.
 - 948 ○ The path attribute MUST be ignored by the server if it is specified by the client during the
949 submission of the ClassificationNodeType instance.
 - 950 ○ The path attribute of a ClassificationNode MUST be unique within a server.

951 4.3.3 Canonical Path Syntax

952 The path attribute of the ClassificationNodeType instance contains an absolute path in a canonical
953 representation that uniquely identifies the path leading from the root ClassificationSchemeType instance
954 to that ClassificationNodeType instance.

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

956

```
957 canonicalPath ::= '/' rootTaxonomyElementId nodePath
958 nodePath      ::= '/' nodeCode
959              |   '/' nodeCode ( nodePath )?
960
```

961 In the above grammar, *rootTaxonomyElementId* is the *id* attribute of the root ClassificationSchemeType or
962 ClassificationNodeType instance, and *nodeCode* is defined by NCName production as defined by
963 <http://www.w3.org/TR/REC-xml-names/#NT-NCName>.

964 Example of Canonical Path Representation

965 The following canonical path represents the *path* attribute value for the ClassificationNode with code
966 “Male” in the sample Gender ClassificationScheme presented earlier.

967

```
968 /urn:acme:GenderScheme/Male
```

969 4.4 ClassificationType

970 **Base Type:** [RegistryObjectType](#)

971 A ClassificationType instance classifies a RegistryObjectType instance by using a value defined within a
972 particular ClassificationScheme. An internal Classification specifies the value by referencing the
973 ClassificationNodeType instance within a ClassificationSchemeType instance. An external Classification
974 specifies the value using a string value that is defined in some external specification represented by an
975 external ClassificationSchemeType instance.

976 4.4.1 Syntax

```
977 <complexType name="ClassificationType">
978   <complexContent>
979     <extension base="tns:RegistryObjectType">
```

```

980     <attribute name="classificationScheme"
981       type="tns:objectReferenceType" use="optional"/>
982     <attribute name="classifiedObject"
983       type="tns:objectReferenceType" use="optional"/>
984     <attribute name="classificationNode"
985       type="tns:objectReferenceType" use="optional"/>
986     <attribute name="nodeRepresentation"
987       type="tns:LongText" use="optional"/>
988   </extension>
989 </complexContent>
990 </complexType>

```

991 4.4.2 Example

992 The following examples shows how a Person instance is classified using the sample Gender
 993 ClassificationScheme used in earlier examples.

994

```

995 <rim:RegistryObject xsi:type="rim:PersonType"
996   id="urn:acme:person:Danyal" ...>
997   ...
998   <Classification classifiedObject="urn:acme:person:Danyal"
999     classificationNode="urn:acme:Gender:Male"
1000     ...
1001 </rim:RegistryObject>
1002
1003

```

1004 4.4.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
classificationNode	objectReferenceType	0..1		Client	No
classifiedObject	objectReferenceType	0..1		Client	No
classificationScheme	objectReferenceType	0..1		Client	No
nodeRepresentation	LongText	0..1		Client	No

1005

- 1006 ● Attribute *classificationNode* - If the ClassificationType instance represents an internal
 1007 classification, then the *classificationNode* attribute is required.
 - 1008 ○ The *classificationNode* value MUST reference a ClassificationNodeType instance.
- 1009 ● Attribute *classifiedObject* - For both internal and external classifications, the *classifiedObject*
 1010 attribute is required and it references the RegistryObjectType instance that is classified by this
 1011 Classification.
- 1012 ● Attribute *classificationScheme* - If the ClassificationType instance represents an external
 1013 classification, then the *classificationScheme* attribute is required.
 - 1014 ○ The *classificationScheme* value MUST reference a ClassificationScheme instance.
- 1015 ● Attribute *nodeRepresentation* - If the ClassificationType instance represents an external
 1016 classification, then the *nodeRepresentation* attribute is required. It is a representation of a
 1017 taxonomy value from a classification scheme.

1018
1019

- A canonical slot with name “urn:oasis:names:tc:ebxml-regrep:rim:Classification:context” may be optionally specified to provide additional context for a ClassificationType instance

1020

5 Provenance Information Model

1021 The term **provenance** in the English language implies the origin and history of ownership and
1022 custodianship of things of value. When applied to the ebXML RegRep, provenance implies information
1023 about the origin, history of ownership, custodianship, and other relationships between entities such as
1024 people, organizations and information represented by RegistryObjectType instances.

1025 The ebRIM information model supports types and relationships that MAY be used to represent the
1026 provenance of RegistryObjectType instances.

1027 The following figure presents the significant types defined by the provenance information model.

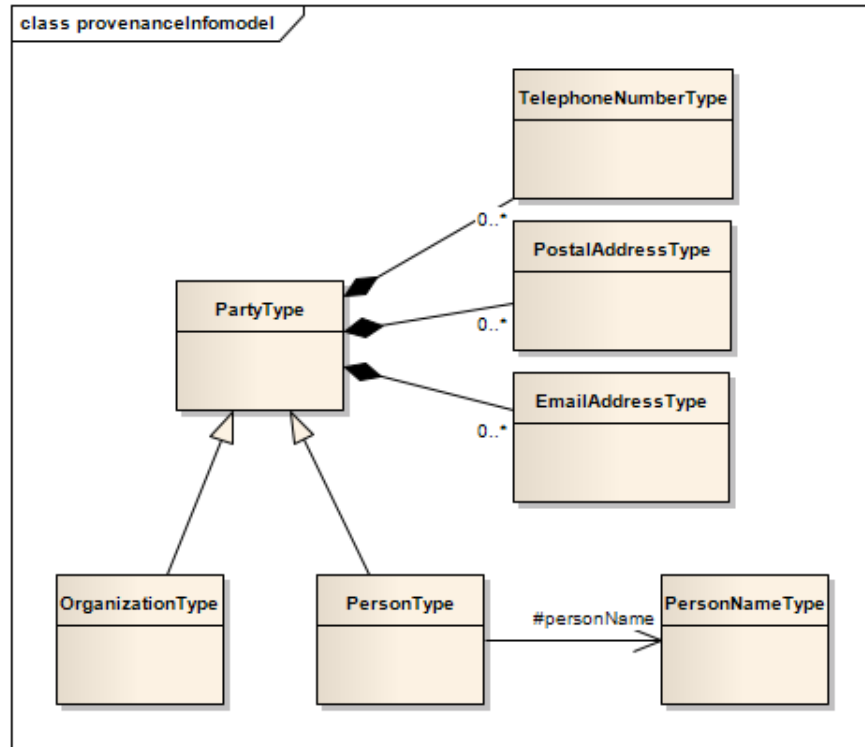


Illustration 6: Provenance Information Model

5.1 PostalAddressType

1031 **Base Type:** [ExtensibleObjectType](#)

1032 This type represents a postal or mailing address.

5.1.1 Syntax

```
1034 <complexType name="PostalAddressType">
1035   <complexContent>
1036     <extension base="tns:ExtensibleObjectType">
1037       <attribute name="city" type="tns:ShortText" use="optional"/>
1038       <attribute name="country" type="tns:ShortText" use="optional"/>
1039       <attribute name="postalCode" type="tns:ShortText" use="optional"/>
1040       <attribute name="stateOrProvince" type="tns:ShortText" use="optional"/>
1041       <attribute name="street" type="tns:ShortText" use="optional"/>

```

```

1042     <attribute name="streetNumber" type="tns:String32" use="optional"/>
1043     <attribute name="type" type="tns:objectReferenceType" use="optional"/>
1044     </extension>
1045     </complexContent>
1046 </complexType>

```

1047 5.1.2 Example

```

1048 <rim:RegistryObject xsi:type="rim:PersonType" id="urn:acme:person:Danyal" ...>
1049     ...
1050     <rim:PostalAddress streetNumber="10" street="Street 1" city="Islamabad"
1051         stateOrProvince="Punjab" country="Pakistan" postalCode="12345"/>
1052     ...
1053 </rim:RegistryObject>

```

1054 5.1.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
city	ShortText	No		Client	Yes
country	ShortText	No		Client	Yes
postalCode	ShortText	No		Client	Yes
stateOrProvince	ShortText	No		Client	Yes
street	ShortText	No		Client	Yes
streetNumber	String32	No		Client	Yes

1055

- 1056 ● Attribute *city* - A *PostalAddressType* instance MAY have a *city* attribute identifying the city for that
1057 address.
- 1058 ● Attribute *country* - A *PostalAddressType* instance MAY have a *country* attribute identifying the
1059 country for that address.
- 1060 ● Attribute *postalCode* - A *PostalAddressType* instance MAY have a *postalCode* attribute identifying
1061 the postal code (e.g., zip code) for that address.
- 1062 ● Attribute *stateOrProvince* - A *PostalAddressType* instance MAY have a *stateOrProvince* attribute
1063 identifying the state, province or region for that address.
- 1064 ● Attribute *street* - A *PostalAddressType* instance MAY have a *street* attribute identifying the street
1065 name for that address.
- 1066 ● Attribute *streetNumber* - A *PostalAddressType* instance MAY have a *streetNumber* attribute
1067 identifying the street number (e.g., 65) for the street address.

1068 5.2 TelephoneNumberType

1069 **Base Type:** [ExtensibleObjectType](#)

1070 This type defines attributes of a telephone number.

1071 5.2.1 Syntax

```

1072 <complexType name="TelephoneNumberType">
1073     <complexContent>
1074         <extension base="tns:ExtensibleObjectType">
1075             <attribute name="areaCode" type="tns:String8" use="optional"/>

```

```

1076     <attribute name="countryCode" type="tns:String8" use="optional"/>
1077     <attribute name="extension" type="tns:String8" use="optional"/>
1078     <attribute name="number" type="tns:String16" use="optional"/>
1079     <attribute name="type" type="tns:objectReferenceType" use="optional"/>
1080   </extension>
1081 </complexContent>
1082 </complexType>

```

1083 5.2.2 Example

```

1084 <rim:RegistryObject xsi:type="rim:PersonType" id="urn:acme:person:Danyal" ...>
1085   ...
1086   <rim:TelephoneNumber countryCode="92" areaCode="51" number="123-4567"
1087     type="urn:oasis:names:tc:ebxml-regrep:PhoneType:MobilePhone"/>
1088   ...
1089 </rim:RegistryObject>

```

1090 5.2.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
areaCode	String8	0..1		Client	Yes
countryCode	String8	0..1		Client	Yes
extension	String8	0..1		Client	Yes
number	String16	0..1		Client	Yes
type	objectReferenceType	0..1		Client	Yes

1091

- 1092 ● Attribute *areaCode* - A *TelephoneNumberType* instance MAY have an *areaCode* attribute that
1093 provides the area code for that telephone number.
- 1094 ● Attribute *countryCode* - A *TelephoneNumberType* instance MAY have a *countryCode* attribute
1095 that provides the country code for that telephone number.
- 1096 ● Attribute *extension* - A *TelephoneNumberType* instance MAY have an *extension* attribute that
1097 provides the extension number, if any, for that telephone number.
- 1098 ● Attribute *number* - A *TelephoneNumberType* instance MAY have a *number* attribute that provides
1099 the local number (without area code, country code and extension) for that telephone number.
- 1100 ● Attribute *type* - A *TelephoneNumberType* instance MAY have a *type* attribute that provides the
1101 type for the *TelephoneNumber*. The value of the *phoneType* attribute MUST be a reference to a
1102 *ClassificationNode* in the canonical *PhoneType ClassificationScheme*.

1103 5.3 EmailAddressType

1104 **Base Type:** [ExtensibleObjectType](#)

1105 This type defines attributes of an email address.

1106 5.3.1 Syntax

```

1107 <complexType name="EmailAddressType">
1108   <complexContent>
1109     <extension base="tns:ExtensibleObjectType">
1110       <attribute name="address" type="tns:ShortText" use="required"/>

```

```

1111     <attribute name="type" type="tns:objectReferenceType" use="optional"/>
1112     </extension>
1113     </complexContent>
1114 </complexType>

```

1115 5.3.2 Example

```

1116 <rim:RegistryObject xsi:type="rim:PersonType" id="urn:acme:person:Danyal" ...>
1117     ...
1118     <rim:EmailAddress address="danyal@play.com"
1119         type="urn:oasis:names:tc:ebxml-regrep:EmailType:HomeEmail"/>
1120     ...
1121 </rim:RegistryObject>
1122
1123

```

1124 5.3.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
address	ShortText	1		Client	Yes
type	objectReferenceType	0..1		Client	Yes

1125

- 1126 ● Attribute address - An EmailAddressType instance MUST have an *address* attribute that provides
1127 the actual email address.
- 1128 ● Attribute type - An EmailAddressType instance MAY have a *type* attribute that provides the type
1129 for that email address. The value of the type attribute MUST be a reference to a
1130 ClassificationNode in the canonical EmailType ClassificationScheme.

1131 5.4 PartyType

1132 **Base Type:** [RegistryObjectType](#)

1133 This abstract type represents a party that has contact information such as PostalAddress, EmailAddress,
1134 TelephoneNumber etc. It is used as a common base type for PersonType and OrganizationType.

1135 5.4.1 Syntax

```

1136 <complexType name="PartyType" abstract="true">
1137     <complexContent>
1138         <extension base="tns:RegistryObjectType">
1139             <sequence>
1140                 <element name="PostalAddress" type="tns:PostalAddressType"
1141                     minOccurs="0" maxOccurs="unbounded"/>
1142                 <element name="TelephoneNumber" type="tns:TelephoneNumberType"
1143                     minOccurs="0" maxOccurs="unbounded"/>
1144                 <element name="EmailAddress" type="tns:EmailAddressType"
1145                     minOccurs="0" maxOccurs="unbounded"/>
1146             </sequence>
1147         </extension>
1148     </complexContent>
1149 </complexType>

```

1150 **5.4.2 Description**

Node	Type	Cardinality	Default Value	Specified By	Mutable
EmailAddress	EmailAddressType	0..*		Client	Yes
PostalAddress	PostalAddressType	0..*		Client	Yes
TelephoneNumber	TelephoneNumberType	0..*		Client	Yes

1151

- 1152 ● Element EmailAddress - A PartyType instance MAY have any number of EmailAddress sub-
1153 elements. Each EmailAddress provides an email address for that PartyType instance. A
1154 PartyType instance SHOULD have at least one EmailAddress.
- 1155 ● Element PostalAddress - A PartyType instance MAY have any number of PostalAddress sub-
1156 elements. Each PostalAddress element provides a postal address for that PartyType instance. A
1157 PartyType instance SHOULD have at least one PostalAddress.
- 1158 ● Element TelephoneNumber - A PartyType instance MAY have any number of TelephoneNumber
1159 sub-elements. Each TelephoneNumber element provides a TelephoneNumber for that PartyType
1160 instance. A PartyType instance SHOULD have at least one TelephoneNumber.

1161 **5.5 PersonType**

1162 **Base Type:** [PartyType](#)

1163 This type represent a person.

1164 **5.5.1 Syntax**

```

1165 <complexType name="PersonType">
1166   <complexContent>
1167     <extension base="tns:PartyType">
1168       <sequence>
1169         <element name="PersonName" type="tns:PersonNameType"
1170           minOccurs="0" maxOccurs="1"/>
1171       </sequence>
1172     </extension>
1173   </complexContent>
1174 </complexType>

```

1175 **5.5.2 Example**

```

1176 <rim:RegistryObject xsi:type="rim:PersonType" id="urn:acme:person:Danyal" ...>
1177   <rim:PersonName firstName="Danyal" middleName="Idris" lastName="Najmi"/>
1178   <rim:PostalAddress streetNumber="10" street="Street 1" city="Islamabad"
1179     stateOrProvince="Punjab" country="Pakistan" postalCode="12345"/>
1180   <rim:TelephoneNumber countryCode="92" areaCode="51" number="123-4567"
1181     type="urn:oasis:names:tc:exml-regrep:PhoneType:MobilePhone"/>
1182   <rim:EmailAddress address="danyal@play.com"
1183     type="urn:oasis:names:tc:exml-regrep:EmailType:HomeEmail"/>
1184 </rim:RegistryObject>

```

1185 **5.5.3 Description**

Node	Type	Cardinality	Default Value	Specified By	Mutable
------	------	-------------	---------------	--------------	---------

PersonName	PersonNameType	0..1		Client	No
------------	----------------	------	--	--------	----

1186

- Element PersonName – A PersonType instance SHOULD have a PersonName sub-element that provides the name for that person.

1189 5.6 PersonNameType

1190 **Base Type:** ExtensibleObjectType

1191 This represents the name for a PersonType instance.

1192 5.6.1 Syntax

```

1193 <complexType name="PersonNameType">
1194   <complexContent>
1195     <extension base="tns:ExtensibleObjectType">
1196       <attribute name="firstName" type="tns:ShortText" use="optional"/>
1197       <attribute name="middleName" type="tns:ShortText" use="optional"/>
1198       <attribute name="lastName" type="tns:ShortText" use="optional"/>
1199     </extension>
1200   </complexContent>
1201 </complexType>

```

1202 5.6.2 Example

```

1203 <rim:RegistryObject xsi:type="rim:PersonType" id="urn:acme:person:Danyal" ...>
1204   ...
1205   <rim:PersonName firstName="Danyal" middleName="Idris" lastName="Najmi"/>
1206   ...
1207 </rim:RegistryObject>

```

1208 5.6.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
firstName	ShortText	0..1		Client	Yes
lastName	ShortText	0..1		Client	Yes
middleName	ShortText	0..1		Client	Yes

1209

- Attribute firstName - A PersonName instance SHOULD have a *firstName* attribute that is the given name of the person.
- Attribute lastName - A PersonName instance SHOULD have a *lastName* attribute that is the family name of the person.
- Attribute middleName - A PersonName instance SHOULD have a *middleName* attribute that is the middle name of the person.

1216 5.7 OrganizationType

1217 **Base Type:** PartyType

1218 This type represents an organization or entity.

1219 5.7.1 Syntax

```
1220 <complexType name="OrganizationType">
1221   <complexContent>
1222     <extension base="tns:PartyType">
1223       <sequence>
1224         <element name="Organization" type="tns:OrganizationType"
1225           minOccurs="0" maxOccurs="unbounded"/>
1226       </sequence>
1227       <attribute name="primaryContact" type="tns:objectReferenceType"
1228         use="optional"/>
1229     </extension>
1230   </complexContent>
1231 </complexType>
```

1232 5.7.2 Example

```
1233 <rim:RegistryObject xsi:type="rim:OrganizationType"
1234   id="urn:acme:Organization:acme"
1235   primaryContact="urn:acme:person:Danyal" ...>
1236   <rim:PostalAddress streetNumber="1" street="Grand Trunk Rd."
1237     city="Hasan Abdal"
1238     stateOrProvince="Punjab" country="Pakistan" postalCode="12345"/>
1239   <rim:TelephoneNumber countryCode="92" areaCode="52" number="123-4567"
1240     type="urn:oasis:names:tc:ebxml-regrep:PhoneType:OfficePhone"/>
1241   <rim:EmailAddress address="info@acme.com"
1242     type="urn:oasis:names:tc:ebxml-regrep:EmailType:OfficeEmail"/>
1243 </rim:RegistryObject>
```

1244 5.7.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
Organization	OrganizationType	0..*		Client	Yes
primaryContact	objectReferenceType	0..1		Client	Yes

1245

- 1246 ● Element Organization – This element allows clients to specify sub-organizations of the
1247 Organization instance using a simpler alternative to specifying “HasMember” AssociationType
1248 instances between the parent and child Organizations.
 - 1249 ○ A server MUST replace any nested Organization elements within an OrganizationType
1250 instance with AssociationType instances such that each nested Organization element is
1251 replaced with an AssociationType instance with type “urn:oasis:names:tc:ebxml-
1252 regrep:AssociationType:HasMember”, with sourceObject specifying the id of the parent
1253 OrganizationType instance and with targetObject specifying the id of the nested Organization
1254 element
- 1255 ● Attribute primaryContact - An OrganizationType instance SHOULD have a *primaryContact*
1256 attribute that references the Person instance for the person that is the primary contact for that
1257 organization.

1258 5.8 Associating Organization With Persons

1259 There are many situation where a person is related to an organization. Such relationship SHOULD be
1260 defined by AssociationType instances between an OrganizationType instance and a PersonType instance
1261 as follows:

- 1262 ● The type attribute of the Association references the canonical ClassificationNode with id
1263 “urn:oasis:names:tc:ebxml-regrep:AssociationType:AffiliatedWith” or one of its descendants.
- 1264 ● The sourceObject references the PersonType instance.
- 1265 ● The targetObject references the OrganizationType instance.

1266 **5.9 Associating Organization With Organizations**

1267 There are many situation where an organization is related to another organization. Such relationship
1268 SHOULD be defined by AssociationType instances between an OrganizationType instance and another
1269 OrganizationType instance.

- 1270 ● To represent parent-child relationship between organizations the type attribute of the Association
1271 SHOULD reference the canonical ClassificationNode with id “urn:oasis:names:tc:ebxml-
1272 regrep:AssociationType:HasMember” or one of its descendants.
- 1273 ● The sourceObject SHOULD reference the parent OrganizationType instance.
- 1274 ● The targetObject SHOULD reference the child OrganizationType instance.

1275 **5.10 Associating Organizations With RegistryObjects**

1276 An organization MAY be associated with zero or more RegistryObjectType instances. Each such
1277 association is modeled in ebRIM using an Association instance between an Organization instance and a
1278 RegistryObjectType instance.

1279 Associations between Organizations and RegistryObjectType instances do not entitle organizations to any
1280 special privileges with respect to those instances. Such privileges are defined by the Access Control
1281 Policies defined for the RegistryObjectType instances as described in the [Access Control Information](#)
1282 [Model chapter](#).

1283

6 Service Information Model

1284 This chapter describes the parts of the information model that support the description of services within an
 1285 ebXML RegRep server. Although service information model aligns with [WSDL2] model, it may be used to
 1286 describe any type of service in addition to web services.

1287

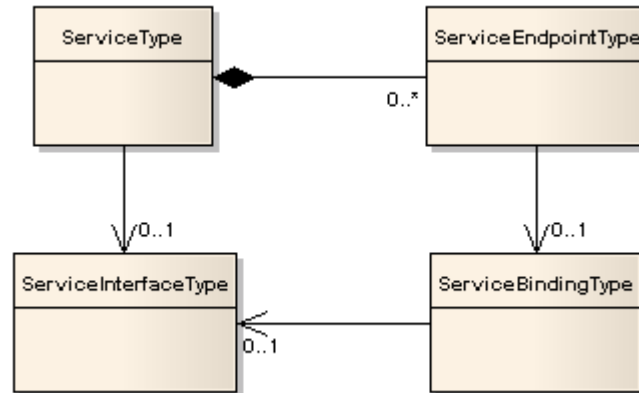


Illustration 7: Service Information Model

1289

6.1 ServiceType

1290

Base Type: RegistryObjectType

1291 This type represent a logical service. Physical service endpoints are represented by the
 1292 [ServiceEndpointType](#) type. A ServiceType instance typically contains ServiceEndpoint sub-elements
 1293 where each ServiceEndpoint sub-element represents an alternate endpoint for a service.

1294

6.1.1 Syntax

1295

```

1296 <complexType name="ServiceType">
1297   <complexContent>
1298     <extension base="tns:RegistryObjectType">
1299       <sequence>
1300         <element name="ServiceEndpoint" type="tns:ServiceEndpointType"
1301           minOccurs="0" maxOccurs="unbounded"/>
1302       </sequence>
1303       <attribute name="serviceInterface"
1304         type="tns:objectReferenceType" use="optional" />
1305     </extension>
1306   </complexContent>
</complexType>

```

1307

6.1.2 Example

1308

```

1309 <rim:RegistryObject xsi:type="rim:ServiceType"
1310   id="urn:acme:Service:StockQuoteService" ...>
1311   ...
1312   <rim:ServiceEndpoint
1313     id="urn:acme:ServiceEndpoint:StockQuoteService:free" .../>
1314   <rim:ServiceEndpoint

```

```

1314     id="urn:acme:ServiceEndpoint:StockQuoteService:premium" .../>
1315 </rim:RegistryObject>

```

1316 6.1.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
ServiceEndpoint	ServiceEndpointType	0..*		Client	Yes
serviceInterface	objectReferenceType	0..1		Client	Yes

1317

- 1318 ● Element ServiceEndpoint – Represents a physical endpoint for the service that MAY be used by
- 1319 clients to access the service
- 1320 ● Attribute serviceInterface – References the abstract interface description for the service
- 1321 ○ MUST reference a [ServiceInterfaceType](#) instance if specified

1322 6.2 ServiceEndpointType

1323 **Base Type:** [RegistryObjectType](#)

1324 This type represents a physical endpoint for the service that MAY be used by clients to access a service.

1325 6.2.1 Syntax

```

1326 <complexType name="ServiceEndpointType">
1327   <complexContent>
1328     <extension base="tns:RegistryObjectType">
1329       <attribute name="address" type="anyURI" use="optional" />
1330       <attribute name="serviceBinding"
1331         type="tns:objectReferenceType" use="optional" />
1332     </extension>
1333   </complexContent>
1334 </complexType>

```

1335 6.2.2 Example

```

1336 <rim:RegistryObject xsi:type="rim:ServiceType"
1337   id="urn:acme:Service:StockQuoteService" ...>
1338   ...
1339   <rim:ServiceEndpoint id="urn:acme:ServiceEndpoint:StockQuoteService:free"
1340     address="http://acme.com/StockQuoteService/free"
1341     serviceBinding="urn:acme:ServiceBinding:soap:StockQuoteService">
1342 </rim:RegistryObject>

```

1343 6.2.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
address	xs:anyURI	0..1		Client	Yes
serviceBinding	objectReferenceType	0..1		Client	Yes

1344

- 1345 ● Attribute address – Represents the endpoint address URI that a client of the service endpoint may
- 1346 use to access the service endpoint

- 1347 ● Attribute `serviceBinding` – References the [ServiceBindingType](#) instance that represents protocol-
1348 specific binding information for the `ServiceEndpointType` instance
- 1349 ○ MUST reference a `ServiceBindingType` instance

1350 6.3 ServiceBindingType

1351 **Base Type:** [RegistryObjectType](#)

1352 This type represents protocol-specific binding information for a `ServiceEndpointType` instance. Example of
1353 a protocol-specific binding is a SOAP binding.

1354 6.3.1 Syntax

```
1355 <complexType name="ServiceBindingType">
1356   <complexContent>
1357     <extension base="tns:RegistryObjectType">
1358       <attribute name="serviceInterface"
1359         type="tns:objectReferenceType" use="optional" />
1360     </extension>
1361   </complexContent>
1362 </complexType>
```

1363 6.3.2 Example

```
1364 <rim:RegistryObject xsi:type="rim:ServiceBindingType"
1365   id="urn:acme:ServiceBinding:soap:StockQuoteService"
1366   serviceInterface="urn:acme:ServiceInterface:StockQuoteService" .../>
1367   ...
1368 </rim:RegistryObject>
```

1369 6.3.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
<code>serviceInterface</code>	<code>objectReferenceType</code>	0..1		Client	Yes

- 1370
- 1371 ● Attribute `serviceInterface` – References a `ServiceInterfaceType` instance which represents the
1372 abstract service interface for the service
- 1373 ○ MUST reference a `ServiceInterfaceType` instance if specified

1374 6.4 ServiceInterfaceType

1375 **Base Type:** [RegistryObjectType](#)

1376 This type represents an abstract service interface for a service.

1377 6.4.1 Syntax

```
1378 <complexType name="ServiceInterfaceType">
1379   <complexContent>
1380     <extension base="tns:RegistryObjectType">
1381     </extension>
1382   </complexContent>
```

1383 </complexType>

1384 6.4.2 Example

```
1385 <rim:RegistryObject xsi:type="rim:ServiceInterfaceType"  
1386   id="urn:acme:ServiceInterface:StockQuoteService" .../>  
1387   ...  
1388 </rim:RegistryObject>
```

1389 6.4.3 Description

1390 No attributes or elements beyond those inherited from [RegistryObjectType](#) are defined for this type.

1391

7 Query Information Model

1392
1393

This chapter describes the information model for defining and invoking parameterized queries in ebXML RegRep. The following significant types are defined by the Query Information Model:

1394
1395

- QueryDefinitionType - Represents the definition of a parameterized query
- QueryType – Represents the invocation of a parameterized query

1396
1397
1398
1399

Several canonical QueryDefinitionType instances are defined by the ebRS specification. Profiles of ebXML RegRep MAY define additional QueryDefinitionType instances as canonical queries for that profile. Deployments MAY also define additional QueryDefinitionType instances. Finally, clients MAY submit additional QueryDefinitionType instances.

1400
1401
1402

A QueryDefinitionType instance MAY be invoked using a QueryType instance. The ebRS Query protocol allows clients to invoke a QueryDefinitionType instance using a QueryType instance within the Query protocol.

1403

The following figure presents the significant types defined by the Query information model.

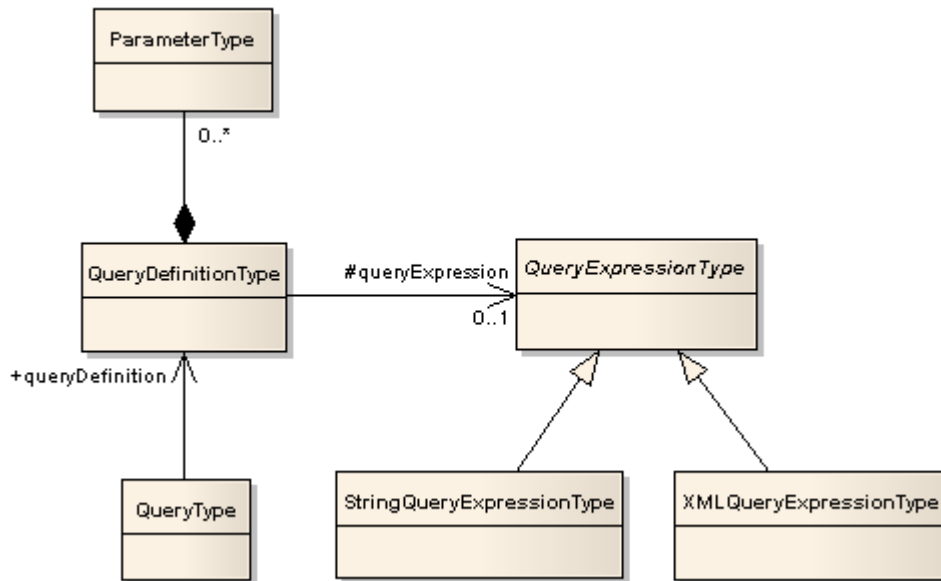


Illustration 8: Query Information Model

1405

7.1 QueryDefinitionType

1406

Base Type: RegistryObjectType

1407
1408

This type represents the definition of a parameterized query. The definition of a query includes the definition of its supported parameters and the definition of a parameterized query expression.

1409

7.1.1 Syntax

1410
1411
1412
1413
1414

```

<complexType name="QueryDefinitionType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="Parameter"
  
```

```

1415         type="tns:ParameterType" minOccurs="0" maxOccurs="unbounded"/>
1416         <element name="QueryExpression"
1417             type="tns:QueryExpressionType" minOccurs="0" maxOccurs="1"/>
1418     </sequence>
1419 </extension>
1420 </complexContent>
1421 </complexType>

```

1422 7.1.2 Example

```

1423 <rim:RegistryObject xsi:type="rim:QueryDefinitionType"
1424     id="urn:oasis:names:tc:ebxml-regrep:query:GetObjectById">
1425     <rim:Parameter parameterName="id"
1426         minOccurs="1" maxOccurs="1" defaultValue="%">
1427     </Parameter>
1428     <rim:QueryExpression xsi:type="rim:StringQueryExpressionType"
1429         queryLanguage="urn:oasis:names:tc:ebxml-regrep:QueryLanguage:EJBQL">
1430     <Value>
1431         SELECT Object(ro) FROM ...
1432     </Value>
1433     </QueryExpression>
1434 </rim:RegistryObject>

```

1435 7.1.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
Parameter	ParameterType	0..*		Client	Yes
QueryExpression	QueryExpressionType	0..1		Client	Yes

1436

- 1437 ● Element Parameter – Represents the definition of a query parameter for the QueryDefinitionType
- 1438 instance. A QueryDefinitionType instance MAY have any number of Parameter sub-elements
- 1439 ● Element QueryExpression – Represents a query expression for the parameterized query.
- 1440 ○ MAY be omitted if the query is implemented as a Query plugin as defined by ebRS

1441 7.2 ParameterType

1442 **Base Type:** [ExtensibleObjectType](#)

1443 This type represents the definition of a parameter within a QueryDefinitionType.

1444 7.2.1 Syntax

```

1445 <complexType name="ParameterType">
1446     <complexContent>
1447         <extension base="tns:ExtensibleObjectType">
1448             <sequence>
1449                 <element name="Name" type="tns:InternationalStringType"
1450                     minOccurs="1" maxOccurs="1"/>
1451                 <element name="Description" type="tns:InternationalStringType"
1452                     minOccurs="0" maxOccurs="1"/>
1453             </sequence>
1454             <attribute name="parameterName" type="string" use="required"/>
1455             <attribute name="dataType" type="string" use="required" />
1456             <attribute name="defaultValue" type="string" use="optional"/>

```

```

1457     <attribute name="minOccurs" type="nonNegativeInteger" default="1"/>
1458     <attribute name="maxOccurs" type="nonNegativeInteger" default="1"/>
1459     </extension>
1460 </complexContent>
1461 </complexType>

```

1462 7.2.2 Example

```

1463 <rim:RegistryObject xsi:type="rim:QueryDefinitionType"
1464   id="urn:oasis:names:tc:ebxml-regrep:query:GetObjectById">
1465   <rim:Parameter parameterName="id" dataType="string" minOccurs="1"
1466     maxOccurs="1" defaultValue="%" />
1467   ...
1468   <rim:QueryExpression .../>
1469 </rim:RegistryObject>

```

1470 7.2.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
dataType	xs:string	1		Client	Yes
defaultValue	xs:string	0..1		Client	Yes
Description	InternationalStringType	0..1		Client	Yes
minOccurs	xs:nonNegativeInteger	0..1	1	Client	Yes
maxOccurs	xs:nonNegativeInteger	0..1	1	Client	Yes
Name	InternationalStringType	1		Client	Yes
parameterName	xs:string	1		Client	Yes

1471

- 1472 ● Attribute dataType – Specifies the data type for the parameter.
 - 1473 ○ The dataType MUST be “string” for parameters whose values are represented by a string
 - 1474 value.
 - 1475 ○ The dataType MUST be “boolean” for parameters whose values are represented by a
 - 1476 boolean value.
 - 1477 ○ The dataType MUST be “taxonomyElement” for parameters whose value is the id of a
 - 1478 TaxonomyElement.
- 1479 ● Attribute defaultValue - Specifies the default value for the parameter. This value MUST be used
- 1480 as parameter value when the query is invoked if the client does not specify a value for this
- 1481 parameter.
- 1482 ● Element Description - Specifies a human-friendly description of the parameter that indicates what
- 1483 the parameter value represents and what kind of value is allowed. The description MAY be
- 1484 provided in multiple local languages and character sets.
- 1485 ● Attribute minOccurs – Specifies the minimum number of values allowed for the parameter.
- 1486 ● Attribute maxOccurs - Specifies the maximum number of values allowed for the parameter.
- 1487 ● Element Name - Specifies a human-friendly name for the parameter. The name MAY be provided
- 1488 in multiple local languages and character sets.
- 1489 ● Attribute parameterName – Specifies the canonical name of the parameter. The canonicalName
- 1490 identifies the parameter in a locale-insensitive manner

- 1491 ○ SHOULD match a declared parameter name within the query expression for the
1492 QueryDefinitionType instance
- 1493 ○ The parameterName MUST be unique across the universe of all sibling ParameterType
1494 instances within a QueryDefinitionType instance

1495 7.3 QueryExpressionType

1496 **Base Type:** ExtensibleObjectType

1497 This type represents a query expression in a specified query language that MAY be used by the server to
1498 invoke a query.

1499 The QueryExpressionType is the abstract root of a type hierarchy for the following more specialized sub-
1500 types:

- 1501 ● StringQueryExpressionType – This type MAY be used to represent non-XML query syntaxes such
1502 as SQL-92 and EJBQL.
- 1503 ● XMLQueryExpressionType - This type MAY be used to represent XML query syntaxes such as
1504 OGC Filter Query.

1505 This specification does not specify a specific query expression syntax that a server must support.

1506 7.3.1 Syntax

```
1507 <complexType name="QueryExpressionType" abstract="true">
1508   <complexContent>
1509     <extension base="tns:ExtensibleObjectType">
1510       <attribute name="queryLanguage"
1511         type="tns:objectReferenceType" use="required"/>
1512     </extension>
1513   </complexContent>
1514 </complexType>
```

1515 7.3.2 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
queryLanguage	objectReferenceType	1		Client	Yes

- 1516
- 1517 ● Attribute queryLanguage – Specifies the query language used by the QueryExpressionType
1518 instance.
- 1519 ○ MUST be a reference to a ClassificationNode in the canonical Query Language
1520 ClassificationScheme whose id is "urn:oasis:names:tc:ebxml-
1521 regrep:classificationScheme:QueryLanguage".

1522 7.4 StringQueryExpressionType

1523 **Base Type:** QueryExpressionType

1524 This type is used to represent non-XML query syntaxes such as SQL-92 and EJBQL.

1525 7.4.1 Syntax

```
1526 <complexType name="StringQueryExpressionType">
1527   <complexContent>
1528     <extension base="tns:QueryExpressionType">
1529       <sequence>
1530         <element name="Value" type="string" minOccurs="1" maxOccurs="1"/>
1531       </sequence>
1532     </extension>
1533   </complexContent>
1534 </complexType>
```

1535 7.4.2 Example

```
1536 <rim:RegistryObject xsi:type="rim:QueryDefinitionType"
1537   id="urn:oasis:names:tc:ebxml-regrep:query:GetObjectById">
1538   <rim:Parameter ... />
1539   ...
1540   <rim:QueryExpression xsi:type="rim:StringQueryExpressionType"
1541     queryLanguage="urn:oasis:names:tc:ebxml-regrep:QueryLanguage:EJBQL">
1542     <Value>
1543       SELECT Object(ro) FROM RegistryObjectType WHERE ...
1544     </Value>
1545   </rim:QueryExpression>
1546 </rim:RegistryObject>
```

1547 7.4.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
Value	xs:string	1		Client	Yes

1548

- 1549 ● Element Value – Specifies the string value representing the actual query expression within the
1550 query language specified by the queryLanguage attribute inherited from base type
1551 QueryExpressionType.

1552 7.5 XMLQueryExpressionType

1553 **Base Type:** [QueryExpressionType](#)

1554 This type is used to represent XML query syntaxes such as OGC Filter Query.

1555 7.5.1 Syntax

```
1556 <complexType name="XMLQueryExpressionType">
1557   <complexContent>
1558     <extension base="tns:QueryExpressionType">
1559       <sequence>
1560         <any namespace="##other"
1561           processContents="lax" minOccurs="1" maxOccurs="1"/>
1562       </sequence>
1563     </extension>
1564   </complexContent>
1565 </complexType>
```

1566 7.5.2 Example

```
1567 <rim:RegistryObject xsi:type="rim:QueryDefinitionType"  
1568 <rim:Parameter ... />  
1569 ...  
1570 <rim:QueryExpression xsi:type="rim:XMLQueryExpressionType"  
1571 queryLanguage="urn:oasis:names:tc:ebxml-regrep:QueryLanguage:EJBQL">  
1572 <ogc:Filter>  
1573 ...  
1574 </ogc:Filter>  
1575 </rim:QueryExpression>  
1576 </rim:RegistryObject>
```

1577 7.5.3 Description

1578 An XMLQueryExpressionType instance MAY contain any XML element from a namespace other than the
1579 name space for rim.xsd. In the example above we use an ogc:Filter element to represent an OGC Filter
1580 query.

1581 7.6 QueryType

1582 **Base Type:** [ExtensibleObjectType](#)

1583 This type represents the invocation of a parameterized query.

1584 7.6.1 Syntax

```
1585 <complexType name="QueryType">  
1586 <complexContent>  
1587 <extension base="tns:ExtensibleObjectType">  
1588 <attribute name="queryDefinition"  
1589 type="tns:objectReferenceType" use="required"/>  
1590 </extension>  
1591 </complexContent>  
1592 </complexType>
```

1593 7.6.2 Example

```
1594 <rim:RegistryObject xsi:type="rim:QueryType"  
1595 queryDefinition="urn:oasis:names:tc:ebxml-regrep:query:GetObjectById">  
1596 <rim:Slot name="id">  
1597 <rim:SlotValue xsi:type="rim:StringValueType">  
1598 <rim:Value>urn:acme:person:Danyal</rim:Value>  
1599 </rim:SlotValue>  
1600 </rim:Slot>  
1601 </rim:RegistryObject>
```

1602 7.6.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
queryDefinition	objectReferenceType	1		Client	Yes

1603

- 1604 ● Attribute queryDefinition – References the parameterized query to be invoked by the server.

- 1605 ○ The value of this attribute MUST be a reference to a QueryDefinitionType instance that is
1606 supported by the server.
- 1607 ● Element Slot (Inherited) - Each Slot element specifies a parameter value for a parameter
1608 supported by the QueryDefinitionType instance.
- 1609 ○ The slot name MUST match a parameterName attribute within a Parameter's definition within
1610 the QueryDefinitionType instance.
- 1611 ○ The slot value's type MUST match the dataType attribute for the Parameter's definition within
1612 the QueryDefinitionType instance.
- 1613 ○ A server MUST NOT treat the order of parameters as significant.

1614

8 Event Information Model

1615 This chapter defines the information model types that supports the Event Notification feature for ebXML
1616 RegRep. These types include the following:

- 1617 ● **AuditableEventType** – Represents a server event that is typically a consequence of a client
1618 request.
- 1619 ● **SubscriptionType** – Represents a client's subscription to receive notification of
1620 AuditableEventType instances based upon a specified selection criteria.
- 1621 ● **QueryType** – Represents a query invocation that is used to select events of interest within a
1622 SubscriptionType instance. This type has been specified previously in the Query Information
1623 Model.
- 1624 ● **NotificationType** – Represents a notification sent by the server to a client regarding an event that
1625 matches the criteria specified by the client within a SubscriptionType instance.

1626

1627 Illustration 9 shows how a Subscription may be defined that uses a QueryType instance as a selector
1628 query to select the AuditableEvents of interest to the subscriber. The Subscription MAY also have zero or
1629 more DeliveryInfoType elements that specify the subscriber's endpoint to deliver the selected events to.
1630 The endpoint may be a REST or SOAP service endpoint or it may be an email address endpoint in case
1631 notification is to be delivered via email.

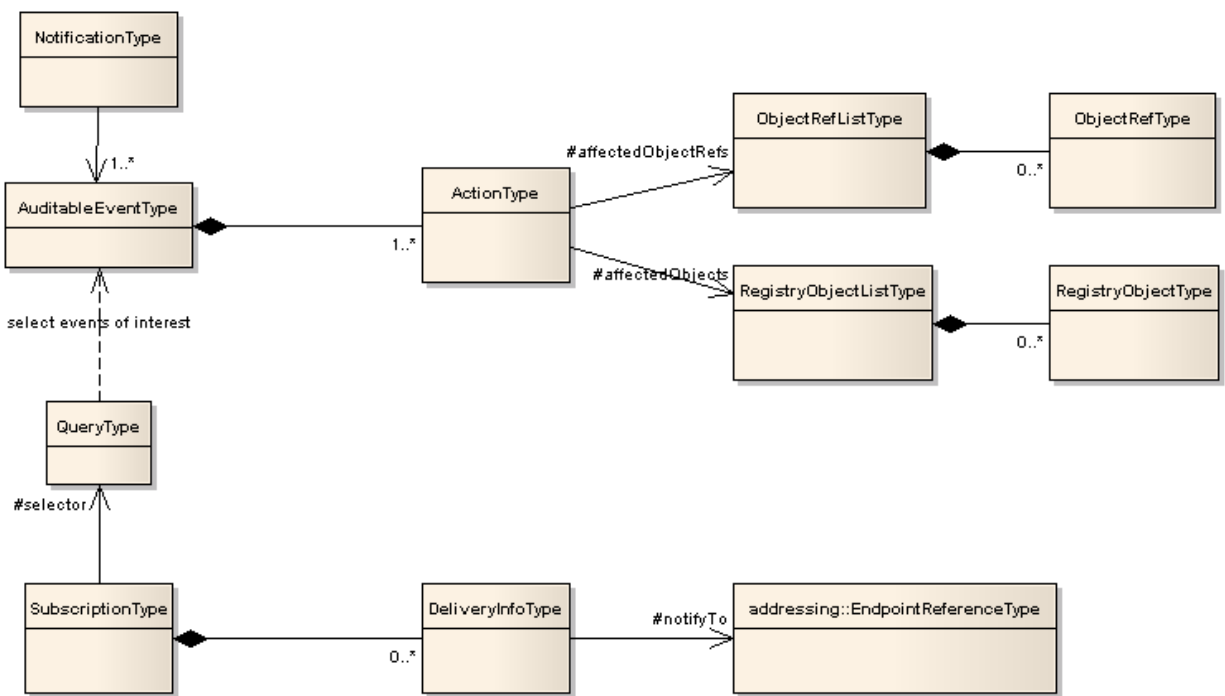


Illustration 9: Event Information Model

1633

8.1 AuditableEventType

1634

Base Type: RegistryObjectType

1635 This type represents a server event. AuditableEventType instances provide a long-term record of events
 1636 that effected changes in the state of a RegistryObjectType instance. AuditableEventType instances MUST
 1637 be generated by the server and MUST NOT be submitted by clients.

1638 AuditableEventType instances represent a change in the state of a RegistryObjectType instance. For
 1639 example a client request could Create, Update, Deprecate or Delete a RegistryObjectType instance. An
 1640 AuditableEventType instance is created when a request creates or alters the state of a
 1641 RegistryObjectType instance. Read-only requests typically do not generate an AuditableEventType
 1642 instance.

1643 8.1.1 Syntax

```
1644 <complexType name="AuditableEventType">
1645   <complexContent>
1646     <extension base="tns:RegistryObjectType">
1647       <sequence>
1648         <element name="Action" type="tns:ActionType"
1649           minOccurs="1" maxOccurs="unbounded"/>
1650       </sequence>
1651       <attribute name="timestamp" type="dateTime" use="required"/>
1652       <attribute name="user" type="string" use="required"/>
1653       <attribute name="requestId"
1654         type="string" use="required"/>
1655     </extension>
1656   </complexContent>
1657 </complexType>
```

1658 8.1.2 Example

1659 The following example shows an AuditableEventType instance that logs the creation of an object within
 1660 the context of a client request.

```
1661 <rim:RegistryObject xsi:type="rim:AuditableEventType"
1662   requestId="urn:uuid:24cee176-9098-4931-894f-fea5dab1732a"
1663   timestamp="2008-01-10T19:20:30+01:00" user="farid"
1664   ...>
1665   <rim:Action eventType="urn:oasis:names:tc:ebxml-regrep:EventType:Created">
1666     <rim:AffectedObjectRefs>
1667       <rim:ObjectRef id="urn:acme:person:Danyal" />
1668     </rim:AffectedObjectRefs>
1669   </rim:Action>
1670 </AuditableEvent>
```

1671 8.1.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
Action	ActionType	1..*		Registry	No
requestId	xs:string	1		Registry	No
timestamp	xs:dateTime	1		Registry	No
user	xs:string	1		Registry	No

1672

- 1673 ● Element Action – Represents an action taken by the server within the context of an
 1674 AuditableEventType instance. An AuditableEventType instance MUST have one or more Action
 1675 instances.

- Attribute requestId – Specifies the id of the request that generated the AuditableEventType instance.
- Attribute timestamp – Specifies the timestamp that represents the date and time the event occurred.
- Attribute user – Specifies the id of the registered user associated with the client that made the request to the server that generated the AuditableEventType instance. Note that the inherited attribute owner SHOULD be set by a server to an internal system user since it is the server and not the user associated with the request that creates an AuditableEventType instance

1684 8.2 ActionType

1685 **Base Type:** ExtensibleObjectType

1686 Represents an action taken by the server within the context of an AuditableEventType instance.

1687 8.2.1 Syntax

```

1688 <complexType name="ActionType">
1689   <sequence>
1690     <element name="AffectedObjects" type="tns:RegistryObjectListType"
1691       minOccurs="0" maxOccurs="1"/>
1692     <element name="AffectedObjectRefs" type="tns:ObjectRefListType"
1693       minOccurs="0" maxOccurs="1"/>
1694   </sequence>
1695   <attribute name="eventType" type="tns:objectReferenceType" use="required"/>
1696 </complexType>

```

1697 8.2.2 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
AffectedObjects	RegistryObjectListType	0..1		Registry	No
AffectedObjectRefs	ObjectRefListType	0..1		Registry	No
eventType	objectReferenceType	1		Registry	No

1698

- Element AffectedObject – Identifies the RegistryObjectType instances that were affected by the event. The AffectedObject element contains any number of elements of type RegistryObjectType, each of which is a RegistryObjectType instance affected by the event. If this element is present then AffectedObjectRefs element MUST NOT be present.
- Element AffectedObjectRefs – Identifies the RegistryObjectType instances that were affected by the event. The AffectedObject element contains any number of ObjectRef elements each of which reference a RegistryObjectType instance that was affected by the event. If this element is present then AffectedObjects element MUST NOT be present.
- Attribute eventType – Specifies the type of event associated with the Action within an AuditableEventType instance.
 - The value of the eventType attribute MUST be a reference to a ClassificationNode in the canonical EventType ClassificationScheme.
 - A Registry MUST support the event types as defined by the EventType ClassificationScheme.
 - The canonical EventType ClassificationScheme MAY easily be extended by adding additional ClassificationNodes to it

1714 8.3 SubscriptionType

1715 **Base Type:** RegistryObjectType

1716 This type represents a subscription on behalf of a client to receive notifications by the server of events that
1717 are of interest to the client.

1718 8.3.1 Syntax

```
1719 <complexType name="SubscriptionType">
1720   <complexContent>
1721     <extension base="tns:RegistryObjectType">
1722       <sequence>
1723         <element name="DeliveryInfo"
1724           type="tns:DeliveryInfoType" minOccurs="0" maxOccurs="unbounded" />
1725         <element name="Selector"
1726           type="tns:QueryType" minOccurs="1" maxOccurs="1" />
1727       </sequence>
1728       <attribute name="startTime" type="dateTime" use="optional"/>
1729       <attribute name="endTime" type="dateTime" use="optional"/>
1730       <attribute name="notificationInterval"
1731         type="duration" use="optional"/>
1732     </extension>
1733   </complexContent>
1734 </complexType>
```

1735 8.3.2 Example

1736 The following example shows a subscription to receive notification of changes to the object whose id value
1737 matches "urn:acme:person:Danyal". The DeliveryInfo specifies the SOAP endpoint where the server
1738 should deliver the Notification.

```
1739 <rim:RegistryObject xsi:type="rim:SubscriptionType"
1740   id="urn:acme:Subscription:subscribeToDanyal"
1741   startTime="2008-01-10T19:20:30+01:00" endTime="2009-01-10T19:20:30+01:00"
1742   ...>
1743   <DeliveryInfo>
1744     <NotifyTo>
1745       <wsa:Address rim:endpointType="urn:oasis:names:tc:ebxml-
1746   regrep:endPointType:soap">http://www.acme.com/notificationListener</wsa:Address>
1747     </NotifyTo>
1748   </DeliveryInfo>
1749   <Selector queryDefinition="urn:oasis:names:tc:ebxml-
1750   regrep:query:GetObjectById">
1751     <Slot name="id">
1752       <SlotValue xsi:type="rim:StringValueType">
1753         <Value>urn:acme:person:Danyal</Value>
1754       </SlotValue>
1755     </Slot>
1756   </Selector>
1757 </rim:RegistryObject>
```

1759 8.3.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
DeliveryInfo	DeliveryInfoType	0..*		Client	Yes
endTime	xs:dateTime	0..1		Client	Yes
notificationInterval	xs:duration	0..1		Client	Yes

Selector	QueryType	1		Client	Yes
startTime	xs:dateTime	0..1	Time of submission	Client	Yes

1760

- 1761 ● Attribute `startTime`, `endTime` – Define the time window within which the subscription is valid.
- 1762 ○ A server MUST use the current time at the time of submission of Subscription as value for the
- 1763 `startTime` attribute if it is unspecified.
- 1764 ○ The Subscription validity window MUST be inclusive of the `startTime` and `endTime`.
- 1765 ○ If `endTime` is unspecified then a server MUST assume the Subscription is valid at any time
- 1766 any time since `startTime` inclusively.
- 1767 ● Element `DeliveryInfo` – Specifies the information needed by the server to deliver notifications for
- 1768 the subscription. It includes the reference to the endpoint where notifications should be delivered.
- 1769 ○ A server MUST deliver notifications that match the Selector query for a valid `SubscriptionType`
- 1770 instance to the endpoint specified by each `DeliveryInfo` element of the `SubscriptionType`
- 1771 instance.
- 1772 ○ If no `DeliveryInfo` element is present then client MUST use the canonical query `GetNotification`
- 1773 via the Query protocol to “pull” the pending notification if any at a time of their choosing as
- 1774 defined in ebRS.
- 1775 ● Attribute `notificationInterval` – Specifies the duration that a server MUST wait between delivering
- 1776 successive notifications to the client. The client specifies this attribute in order to control the
- 1777 frequency of notification communication between server and client.
- 1778 ○ A server MUST deliver any pending notifications within the interval specified by this attribute.
- 1779 ○ A server MUST NOT deliver the same event more than once for the same subscription.
- 1780 ● Element `Selector` – Specifies the query that the server MUST invoke to determine whether an
- 1781 event matches a subscription or not. If the result of the query contains an object that is affected by
- 1782 an event that has not yet been delivered to the subscriber then the event matches the
- 1783 subscription.

1784

1785 8.4 DeliveryInfoType

1786 **Base Type:** [ExtensibleObjectType](#)

1787 This type provides the information needed by the server to *deliver* notifications for the subscription. It

1788 includes the reference to the endpoint where notifications should be delivered. The endpoint reference is

1789 typically one of the following types:

- 1790 ● SOAP service endpoint
- 1791 ● REST service endpoint
- 1792 ● E-mail address endpoint
- 1793 ● Software plugin endpoint that is configured within the same process as the registry server

1794 8.4.1 Syntax

```
1795 <complexType name="DeliveryInfoType">
1796   <complexContent>
1797     <extension base="tns:ExtensibleObjectType">
```

```

1798     <sequence>
1799         <element name="NotifyTo"
1800             type="wsa:EndpointReferenceType" minOccurs="1" maxOccurs="1" />
1801     </sequence>
1802     <attribute name="notificationOption" type="tns:objectReferenceType"
1803 default="urn:oasis:names:tc:ebxml-regrep:NotificationOptionType:ObjectRefs"/>
1804 </extension>
1805 </complexContent>
1806 </complexType>

```

1807 8.4.2 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
notificationOption	objectReferenceType	0..1		Client	Yes
NotifyTo	wsa:EndpointReferenceType	1		Client	Yes

1808

- 1809 ● Attribute notificationOption – Specifies the modality of how notifications are to be delivered to the
 1810 subscriber. Its value MUST reference a ClassificationNode in the canonical
 1811 NotificationOptionType ClassificationScheme.
 - 1812 ○ urn:oasis:names:tc:ebxml-regrep:NotificationOptionType:Objects – Indicates that the server
 1813 MUST provide complete RegistryObjectType instances in notifications delivered to the
 1814 subscriber when this mode is specified.
 - 1815 ○ urn:oasis:names:tc:ebxml-regrep:NotificationOptionType:ObjectRefs – Indicates that the
 1816 server MUST provide ObjectRefType instances rather than complete RegistryObjectType
 1817 instances in notifications delivered to the subscriber when this mode is specified. A client
 1818 MAY pull the complete RegistryObjectType instances using Query protocol after receiving the
 1819 notification.
 - 1820 ● Element NotifyTo – Specifies the endpoint reference for the endpoint where the server should
 1821 deliver notifications for the Subscription.
 - 1822 ○ The type of this element is wsa:EndpointReferenceType as defined by [WSA-Core]
 - 1823 ○ The NotifyTo element has a <wsa:Address> sub-element
 - 1824 ○ The content of the <wsa:Address> element is a string representing the endpoint address
 1825 which SHOULD be a URI
 - 1826 ○ The type of endpoint (SOAP, REST, email, ...) is indicated by an extension attribute
 1827 rim:endpointType defined on the <wsa:Address> element as follows:
 - 1828 ■ If endpoint is a SOAP web service then the rim:endpointType attribute value MUST be
 1829 “urn:oasis:names:tc:ebxml-regrep:endPointType:soap”
 - 1830 ■ If endpoint is a REST web service then the rim:endpointType attribute value MUST be
 1831 “urn:oasis:names:tc:ebxml-regrep:endPointType:rest”
 - 1832 ■ If endpoint is an email address then the rim:endpointType attribute value MUST be
 1833 “urn:oasis:names:tc:ebxml-regrep:endPointType:mail”
 - 1834 ■ If endpoint is a software plugin then the rim:endpointType attribute value MUST be
 1835 “urn:oasis:names:tc:ebxml-regrep:endPointType:plugin”
- 1836

1837 8.5 NotificationType

1838 **Base Type:** RegistryObjectType

1839 This type represents a notification that is sent by the server to a client to notify it of server events that are
1840 of interest to the client.

1841

1842 8.5.1 Syntax

```
1843 <complexType name="NotificationType">
1844   <complexContent>
1845     <extension base="tns:RegistryObjectType">
1846       <sequence>
1847         <element name="Event" type="tns:AuditableEventType"
1848           minOccurs="1" maxOccurs="unbounded"/>
1849       </sequence>
1850       <attribute name="subscription"
1851         type="tns:objectReferenceType" use="required"/>
1852     </extension>
1853   </complexContent>
1854 </complexType>
1855 <element name="Notification" type="tns:NotificationType"/>
```

1856 8.5.2 Example

1857 The following example shows a Notification sent by the server for the subscription in earlier example. It
1858 notifies the subscriber that the object with id "urn:acme:person:Danyal" has changed.

```
1859 <Notification subscription="urn:acme:Subscription:subscribeToDanyal" ...>
1860   <Event user="123456" timestamp="2008-10-17T15:44:29.637" ...>
1861     <Action eventType="urn:oasis:names:tc:ebxml-regrep:EventType:Created">
1862       <AffectedObjectRefs>
1863         <ObjectRef id="urn:acme:person:Danyal"/>
1864       </AffectedObjectRefs>
1865     </Action>
1866   </Event>
1867 </Notification>
```

1868 8.5.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
Event	AuditableEventType	1..*		Server	No
subscription	objectReferenceType	1		Server	No

1869

- 1870 ● Element Event – Represents an Event that is of interest to the subscriber.
 - 1871 ○ Unlike an AuditableEvent element that contains all objects affected by it, the Event element
 - 1872 MUST only contain objects that match the selector query of the SubscriptionType instance. It
 - 1873 has only a subset of affected objects compared to the actual AuditableEvent it represents.
 - 1874 The subset of affected objects MUST be those that match the selector query for the
 - 1875 subscription.
 - 1876 ○ The Action elements within the Event element MUST contain a RegistryObjectList element if
 - 1877 subscription's notificationOption is "Push".

- 1878 ○ The Action elements within the Event element MUST contain a RegistryObjectRefList element
- 1879 if subscription's notificationOption is "Pull".
- 1880 ● Attribute subscription – References the SubscriptionType instance for which this is a Notification.

9 Federation Information Model

1881

1882 This chapter describes the information model that support the definition of registry federations. A registry
1883 federation is a set of ebXML RegRep servers that have voluntarily agreed to form a loosely coupled union.
1884 Such a federation may be based on common business interests or membership in a community-of-
1885 interest. Registry federations enabled clients to query the content of their member servers using federated
1886 queries as if they are a single logical server.

9.1 Federation Configuration

1887

1888 A federation is created by the creation of a FederationType instance. A federation may have any number
1889 of registries as well as other federations as its members.

1890 Membership of a registry or federation within a parent federation is established by creating an Association
1891 between the RegistryType or FederationType instance representing the registry or federation seeking
1892 membership, and the FederationType instance representing the parent federation as follows:

- 1893 ● The Association MUST have its associationType be the id of the canonical ClassificationNode
1894 "HasFederationMember"
- 1895 ● The Association MUST have as its sourceObject the FederationType instance representing the
1896 parent federation
- 1897 ● The Association MUST have as its targetObject the RegistryType or FederationType instance that
1898 is seeking membership within the parent federation as shown in Illustration 10.

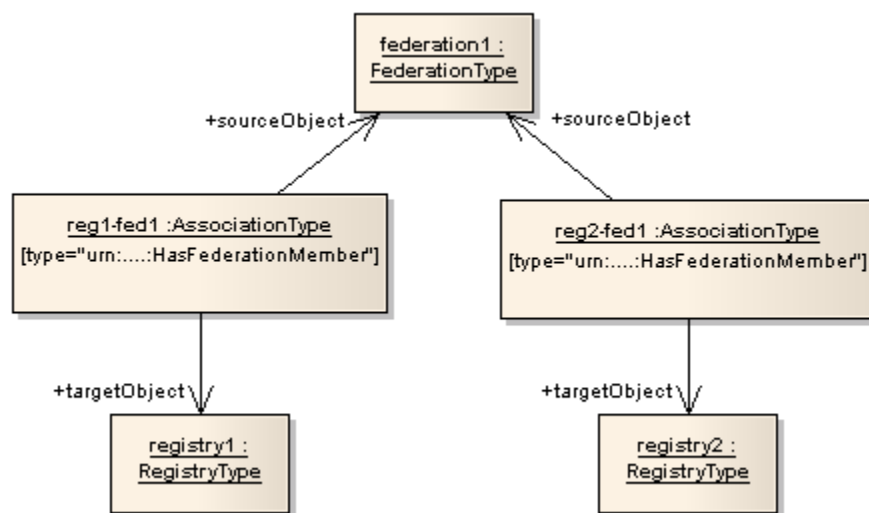


Illustration 10: Federation Information Model

1899

1900

1901 Thus a Federation is defined by a tree where a FederationType instances are root and intermediate n,
1902 RegistryType instances are leaf nodes and HasFederationMember AssociationType instances are the
1903 edges between the nodes. This tree is referred to as the federation membership tree.

9.2 RegistryType

1904

1905 **Base Type:** [RegistryObjectType](#)

1906 RegistryType instances are used to represent an ebXML RegRep server. RegistryType instances are also
 1907 used by a server to advertise the capabilities it supports. A client MAY read the RegistryType instance for
 1908 a server to determine whether it is compatible with a server or not. Profiles of ebXML RegRep
 1909 specifications MAY define canonical slots to represents support for the profile as well as optional features
 1910 defined by the profile.

1911 9.2.1 Syntax

```

1912 <complexType name="RegistryType">
1913   <complexContent>
1914     <extension base="tns:RegistryObjectType">
1915       <attribute name="operator"
1916         type="tns:objectReferenceType" use="required"/>
1917       <attribute name="specificationVersion"
1918         type="string" use="required"/>
1919       <attribute default="P1D" name="replicationSyncLatency"
1920         type="duration" use="optional"/>
1921       <attribute default="PT0S" name="catalogingLatency"
1922         type="duration" use="optional"/>
1923       <attribute name="conformanceProfile"
1924         use="optional" default="RegistryLite">
1925         <simpleType>
1926           <restriction base="NCName">
1927             <enumeration value="RegistryFull"/>
1928             <enumeration value="RegistryLite"/>
1929           </restriction>
1930         </simpleType>
1931       </attribute>
1932     </extension>
1933   </complexContent>
1934 </complexType>
  
```

1935 9.2.2 Example

1936 The following example describes an ebXML RegRep server operated by organization with id
 1937 "urn:acme:Organization:acme-inc", that implements the "RegistryFull" conformance level of version 4.0 of
 1938 the ebXML RegRep specifications. The server performs replication synchronization once a day (P1D) and
 1939 performs cataloging of submitted content immediately when content is submitted.

```

1940 <rim:RegistryObject xsi:type="rim:RegistryType"
1941   id="urn:acme:Registry:serviceRegistry"
1942   operator="urn:acme:Organization:acme-inc"
1943   specificationVersion="4.0"
1944   conformanceProfile="RegistryFull"
1945   replicationSyncLatency="P1D"
1946   catalogingLatency="PT0S"
1947   ...>
1948   ...
1949 </rim:RegistryObject>
  
```

1950 9.2.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
catalogingLatency	xs:duration	0..1	P1D (once a day)	Server	Yes
conformanceProfile	xs:string	0..1	RegistryLite	Server	Yes
operator	objectReferenceType	1		Server	Yes

replicationSyncLatency	xs:duration	0..1	PT0S (immediately)	Server	Yes
specificationversion	objectReferenceType	1		Server	Yes

1951

1952 ● Attribute *catalogingLatency* - A RegistryType instance MAY have an attribute named
 1953 *catalogingLatency* that specifies the maximum latency between the time a submission is made to
 1954 the server and the time it gets cataloged by any cataloging services defined for the objects within
 1955 the submission. The default value of PT0S indicates a duration of 0 seconds which implies that
 1956 cataloging happens immediately when request is submitted.

1957 ● Attribute *conformanceProfile* - A RegistryType instance MAY have an attribute named
 1958 *conformanceProfile* that declares the conformance profile that the server supports. The
 1959 conformance profiles choices are "RegistryLite" and "RegistryFull" as defined by [regrep-rs-v4.0].

1960 ● Attribute *operator* - A RegistryType instance MUST have an attribute named *operator* that is a
 1961 reference to the Organization instance representing the organization for the server's operator.
 1962 Since the same Organization MAY operate multiple registries, it is possible that the home registry
 1963 for the Organization referenced by operator may not be the local registry.

1964 ● Attribute *replicationSyncLatency* - A RegistryType instance MAY have an attribute named
 1965 *replicationSyncLatency* that specifies the maximum latency between the time when an original
 1966 object changes and the time when its replica object within the local server gets updated to
 1967 synchronize with the new state of the original object. The default value of P1D indicates a duration
 1968 of once a day.

1969 ● Attribute *specificationVersion* - A RegistryType instance MUST have an attribute named
 1970 *specificationVersion* that is the version of the ebXML RegReg Specifications it implements.

1971 9.3 FederationType

1972 **Base Type:** RegistryObjectType

1973 Federation instances are used to represent a registry federation. A FederationType instance has a set of
 1974 RegistryType instances as its members. The membership of a RegistryType instance in a federationType
 1975 instance is represented by an AssociationType instance whose type is HasFederationMember.

1976 9.3.1 Syntax

```
1977 <complexType name="FederationType">
1978   <complexContent>
1979     <extension base="tns:RegistryObjectType">
1980       <attribute name="replicationSyncLatency"
1981         type="duration" use="optional" default="P1D" />
1982     </extension>
1983   </complexContent>
1984 </complexType>
```

1985 9.3.2 Example

1986 The following example shows a Federation with two independently-operated ebXML RegRep servers as
 1987 members.

```
1988 <rim:RegistryObject xsi:type="rim:FederationType"
1989   id="urn:acme:Federation:supplierFederation"
1990   replicationSyncLatency="P1D" ...>
1991   ...
```

```

1992 </rim:RegistryObject>
1993
1994 <rim:RegistryObject xsi:type="rim:AssociationType"
1995   sourceObject="urn:acme:Federation:supplierFederation"
1996   targetObject="urn:widgetInc:Registry:widget-inc"
1997   type="urn:oasis:names:tc:ebxml-regrep:AssociationType:HasFederationMember"/>
1998
1999 <rim:RegistryObject xsi:type="rim:AssociationType"
2000   sourceObject="urn:acme:Federation:supplierFederation"
2001   targetObject="urn:supplierInc:Registry:supplier-inc"
2002   type="urn:oasis:names:tc:ebxml-regrep:AssociationType:HasFederationMember"/>

```

2003 9.3.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
replicationSyncLatency	xs:duration	0..1	P1D (1 day)	Client	Yes

2004

- 2005 ● Attribute replicationSyncLatency - A FederationType instance MAY specify a
2006 *replicationSyncLatency* attribute that describes the time duration that is the amount of time within
2007 which a member of this Federation MUST synchronize itself with the current state of the
2008 Federation. Members of the Federation MAY use this parameter to periodically synchronize the
2009 federation metadata they MUST cache locally about the state of the Federation and its members.
2010 Such synchronization MAY be based upon the registry event notification capability.
2011

2012

10 Access Control Information Model

2013 This chapter defines the Information Model used to control access to RegistryObjects and
2014 RepositoryItems managed by it. It also defines a normative profile of [XACML] for ebXML RegRep.

2015 It is assumed that the reader is already familiar with [XACML]. This specification does not provide any
2016 introduction to [XACML].

2017 A server MUST support the roles of both Enforcement Point (PEP) and a Policy Decision Point (PDP) as
2018 defined in [XACML].

2019 The Access Control Model attempts to reuse terms defined by [XACML] wherever possible. The
2020 definitions of some key terms are duplicated here from [XACML] for convenience of the reader:

2021

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"> • A <i>RegistryObject resource</i> • A <i>RepositoryItem resource</i>
Subject	An actor whose attributes may be referenced by within a Policy definition. Examples of subject include: <ul style="list-style-type: none"> • The registered user associated with a client request • An ebXML RegRep server • A software service or agent

2022

2023 10.1 Defining an Access Control Policy

2024 A RegistryObjectType instance is associated with exactly one Access Control Policy that governs “who” is
2025 authorized to perform “what” action on that RegistryObject. This Access Control Policy is expressed as an
2026 [XACML] document which is the repositoryItem for an ExtrinsicObjectType instance. The Access Control
2027 Policy is published to the server as an ExtrinsicObject and repositoryItem pair using the Submit protocol
2028 defined by [regrep-rs-v4.0].

2029 The objectType attribute of this ExtrinsicObject MUST reference a descendent of the “XACML”
2030 ClassificationNode (e.g. “Policy” or PolicySet”) in the canonical ObjectType ClassificationScheme.

2031 10.2 Assigning Access Control Policy to a RegistryObject

2032 An Access Control Policy MAY be assigned to a RegistryObjectType instance using the canonical slot
2033 “urn:oasis:names:tc:ebxml-regrep:rim:RegistryObject:accessControlPolicy”. The value slot references the
2034 ExtrinsicObject representing the Access Control Policy and contains the id of that ExtrinsicObject.

2035 If a RegistryObjectType instance does not have an Access Control Policy explicitly associated with it via
2036 the canonical slot with name “urn:oasis:names:tc:ebxml-regrep:rim:RegistryObject:accessControlPolicy”,
2037 then it is implicitly associated with the [default Access Control Policy](#) defined for the server.

2038

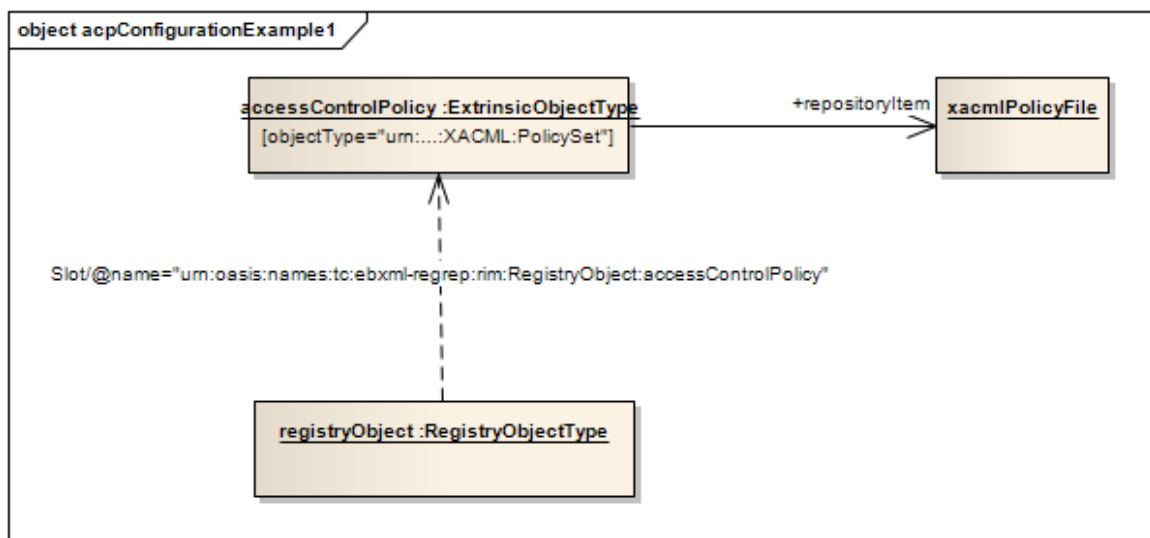


Illustration 11: Assigning Access Control Policy to a RegistryObject

2039

2040

2041 Illustration 11 shows a UML instance diagram where an Organization instance *org* references an
2042 ExtrinsicObject instance *accessControlPolicy* as its Access Control Policy object using the canonical
2043 *accessControlPolicy* slot.

2044 10.2.1 Default Access Control Policy for a RegistryObject

2045 A server MUST support a default Access Control Policy. A server MAY implement any default access
2046 control policy. The default Access Control Policy applies to all RegistryObjectType instances that do not
2047 explicitly have an Access Control Policy assigned.

2048 This following specify the semantics of a suggested default Access Control Policy that a server SHOULD
2049 implement:

- 2050 ● An unauthenticated client is permitted to perform *read* actions (that do not modify the state of
2051 resources) on any resource
- 2052 ● An authenticated client with authentication credentials of a registered user is permitted all actions
2053 on RegistryObjects submitted by the user
- 2054 ● A authenticated client with authentication credentials that are assigned the canonical subject role
2055 of "urn:oasis:names:tc:ebxml-regrep:SubjectRole:RegistryAdministrator" is permitted to perform
2056 any action on any object

2057 **10.2.2 Access Control Policy Inheritance**

2058 A RegistryObjectType instance that does not explicit define an Access Control Policy MAY inherit an
2059 Access Control Policy from its nearest RegistryPackageType ancestor if it is in the membership hierarchy
2060 of a RegistryPackageType instance.

2061 An Access Control Policy for members of a RegistryPackageType instance MAY be assigned to the
2062 RegistryPackageType instance using the canonical slot "urn:oasis:names:tc:ebxml-
2063 regrep:rim:RegistryPackage:memberAccessControlPolicy". The value slot references the ExtrinsicObject
2064 representing the Access Control Policy and contains the id of that ExtrinsicObject. The member Access
2065 Control Policy is implicitly inherited as the applicable Access Control Policy for any member
2066 RegistryObjectType instance that does not have an explicit Access Control Policy assigned to it.

2067 In the event that a RegistryObjectType instance has a member Access Control Policy defined from two
2068 RegistryPackageType ancestors at the same ancestry level a server MAY choose any mechanism to
2069 select one of the two member Access Control Policies.

2070 **Algorithm for Getting Applicable Access Control Policy**

2071 A server MUST implement the following algorithm for determining the applicable Access Control Policy for
2072 a RegistryObjectType instance:

- 2073 ● If an Access Control Policy is explicitly assigned to the object then use it
- 2074 ● If no Access Control Policy is explicitly assigned to the object then get the member Access
2075 Control Policy from a nearest RegistryPackageType ancestor of the object
- 2076 ● If no Access Control Policy is explicitly assigned to the object or inherited from a
2077 RegistryPackageType ancestor of the object then use the system-wide default Access Control
2078 Policy

2079 **10.2.3 Performance Implications**

2080 Excessive use of custom Access Control Policies MAY result in slower processing of registry requests in
2081 some registry implementations. It is therefor suggested that, whenever possible, a submitter SHOULD
2082 reuse an existing Access Control Policy. Submitters SHOULD use good judgment on when to reuse or
2083 extend an existing Access Control Policy and when to create a new one.

2084

2085 **10.3 Defining a Contextual Role**

2086 A contextual role may be define by a RoleType instance within a server as defined next.

2087 10.3.1 RoleType

2088 **Base Type:** RegistryObjectType

2089 Syntax

```
2090 <complexType name="RoleType">
2091   <complexContent>
2092     <extension base="tns:RegistryObjectType">
2093       <attribute name="type" type="tns:objectReferenceType" use="required"/>
2094     </extension>
2095   </complexContent>
2096 </complexType>
```

2097 10.3.2 Example

2098 The following examples shows a RoleType instances representing a contextual role of "ProjectLead"
2099 within the organizational context of an Organization TestSubmittingOrg1.

```
2100 <rim:RegistryObject xsi:type="rim:RoleType"
2101   type="urn:oasis:names:tc:ebxml-regrep:SubjectRole:ProjectLead" ...>
2102   <rim:Slot name="urn:oasis:names:tc:ebxml-
2103   regrep:RoleAssociation:organizationContext">
2104     <rim:SlotValue xsi:type="rim:StringValueType">
2105       <rim:Value>urn:test:Organization:TestSubmittingOrg1</rim:Value>
2106     </rim:SlotValue>
2107   </rim:Slot>
2108 </rim:RegistryObject>
```

2109 10.3.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
type	objectReferenceType	1		Client	Yes

2110

- 2111 ● Attribute type - Each RoleType instance MUST have a type attribute that identifies the type of role.
 - 2112 ○ The value of the type attribute SHOULD be a reference to a ClassificationNode within the
 - 2113 canonical SubjectRole ClassificationScheme.
 - 2114 ○ A server MUST support the canonical subject role types as defined by the canonical
 - 2115 SubjectRole ClassificationScheme. Deployments and profiles may extend the canonical
 - 2116 SubjectRole ClassificationScheme by adding additional ClassificationNodes to it.
- 2117 ● The RoleType instance may have any number of Slots that provide context for the role as a
- 2118 name/value pair. The name attribute of each such context slot provides the context key while the
- 2119 value of the Slot (typically a string) provides the context value

2120 10.4 Assigning a Contextual Role to a Subject

2121 A subject such as a registered user MAY be assigned a contextual role by associating the id attribute
2122 value of the RoleType instance representing the contextual role with the id of the subject. This
2123 specification does not define how such an association is made. Implementations may provide this
2124 association in an implementation specific manner. For example, in an LDAP based identity management
2125 system this may be done by making the node representing the subject to be a member of a groupOfName
2126 node as illustrated below:

```

2127 #person ProjectLead1 definition
2128 dn: uid=ProjectLead1,...
2129 objectclass: top
2130 objectclass: person
2131 objectclass: organizationalPerson
2132 objectclass: inetOrgPerson
2133 cn: Project Lead 1
2134 sn: ProjectLead1
2135 uid: ProjectLead1
2136
2137 #Role TestSubmittingOrg1ProjectLead definition
2138 dn: cn=urn:test:Role:TestSubmittingOrg1ProjectLead,...
2139 objectclass: top
2140 objectclass: groupOfNames
2141 cn: urn:test:Role:TestSubmittingOrg1ProjectLead
2142 member: uid=ProjectLead1,ou=people,...

```

2143 10.5 Action Matching

2144 An XACML Access Control Policy MAY use an action identifier associated with the action as an action
 2145 attribute within <xacml:ActionMatch> elements to match the action that is authorized for a subject on a
 2146 resource.

2147 The following requirements are defined for a server for action matching:

- 2148 ● A server MUST specify the action identifier in the request context for an XACML decision
 2149 request using a <xacmlc:Request>/<xacmlc:Action>/<xacmlc:Attribute> element
 - 2150 ○ The <xacmlc:Attribute> element MUST have an AttributeId attribute with content
 2151 "urn:oasis:names:tc:xacml:1.0:action:action-id"
 - 2152 ○ The <xacmlc:Attribute> element MUST have a DataType attribute with value
 2153 "http://www.w3.org/2001/XMLSchema#string"
 - 2154 ○ The <xacmlc:Attribute> MUST have a <xacmlc:AttributeValue> element whose content
 2155 MUST be the id attribute of the ClassificationNodeType instance within the canonical
 2156 ActionTypeScheme that represent the requested action
- 2157 ● The following requirements are defined for an Access Control Policy for action matching:
- 2158 ● The policy MAY match an action using an <xacmlp:ActionMatch> element
- 2159 ● The <xacmlp:ActionMatch> element MUST have a <xacmlp:AttributeValue
 2160 DataType="http://www.w3.org/2001/XMLSchema#string"> element whose content MUST be the id
 2161 attribute of a ClassificationNodeType instance within the canonical ActionTypeScheme
- 2162 ● The <xacmlp:ActionMatch> element MUST have a <xacmlp:ActionAttributeDesignator
 2163 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
 2164 DataType="http://www.w3.org/2001/XMLSchema#string"/> element

2165

2166 The following example shows an Action that matches the "Read" action.

```

2167 <Target>
2168 <Actions>
2169 <Action>
2170 <ActionMatch
2171 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
2172 <AttributeValue
2173 DataType="http://www.w3.org/2001/XMLSchema#string">
2174 urn:oasis:names:tc:ebxml-regrep:ActionType:read

```

```

2175     </AttributeValue>
2176     <ActionAttributeDesignator
2177         AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
2178         DataType="http://www.w3.org/2001/XMLSchema#string"/>
2179     </ActionMatch>
2180 </Action>
2181 </Actions>
2182 </Target>

```

2183

2184 **10.5.1 Action Attribute: *reference-source***

2185 This attribute is only relevant to the “Reference” action. This attribute MAY be used to specify the object
 2186 from which the reference is being made to the resource being protected. The AttributeId of this attribute
 2187 MUST be “urn:oasis:names:tc:ebxml-regrep:rim:acp:subject:reference-source”. The value of this attribute
 2188 MUST be the value of the id attribute for the object that is the source of the reference. A server MUST
 2189 specify this attribute for a reference action.

2190 **10.5.2 Action Attribute: *reference-source-attribute***

2191 This attribute is only relevant to the “Reference” action. This attribute MAY be used to specify the attribute
 2192 name within the RegistryObjectType that the reference-source object is an instance of. A server MUST
 2193 specify this attribute for a reference action. The AttributeId of this attribute MUST be
 2194 “urn:oasis:names:tc:ebxml-regrep:rim:acp:subject:reference-source-attribute”. The value of this attribute
 2195 MUST be the name of an attribute within the RIM type that is the type for the reference source object.

2196 For example, if the reference source object is an Association instance then the reference-source-attribute
 2197 MAY be used to specify the values “sourceObject” or “targetObject” to restrict the references to be allowed
 2198 from only specific attributes of the source object. This enables, for example, a policy to only allow
 2199 reference to objects under its protection only from the sourceObject attribute of an Association instance.

2200 **10.6 Subject Matching**

2201 An XACML Access Control Policy MAY use the identity and roles associated with the subject as subject
 2202 attributes within <xacml:SubjectMatch> elements to match the subject that is authorized for an action on a
 2203 resource.

2204 The following requirements are defined for a server for subject matching:

- 2205 ● A server MUST specify the subject identifier in the request context for an XACML decision request
 2206 using a <xacmlc:Request>/<xacmlc:Subject>/<xacmlc:Attribute> element
 - 2207 ○ The <xacmlc:Attribute> element MUST have an AttributeId attribute with value
 2208 “urn:oasis:names:tc:xacml:1.0:subject:subject-id”
 - 2209 ○ The <xacmlc:Attribute> element MUST have a DataType attribute with value
 2210 “http://www.w3.org/2001/XMLSchema#string”
 - 2211 ○ The <xacmlc:Attribute> MUST have a <xacmlc:AttributeValue> element whose content MUST
 2212 be the unique id associated with the requestor
- 2213 ● A server MUST specify any subject roles in the request context for an XACML decision request
 2214 using a <xacmlc:Request>/<xacmlc:Subject>/<xacmlc:Attribute> element. This specification does
 2215 not define how roles are assigned to a subject. Implementations SHOULD provide that
 2216 functionality in an implementation-specific manner.

- 2217 ○ The <xacmlc:Attribute> element MUST have an AttributeId attribute with value
2218 "urn:oasis:names:tc:ebxml-regrep:3.0:rim:acp:subject:role"
- 2219 ○ The <xacmlc:Attribute> element MUST have a DataType attribute with value
2220 "<http://www.w3.org/2001/XMLSchema#string>"
- 2221 ○ The <xacmlc:Attribute> MUST have a <xacmlc:AttributeValue> element whose content
2222 MUST be the id attribute value of a RoleType instance associated with the requestor

2223 10.6.1 Matching Subjects By Id

2224 The following requirements are defined for an Access Control Policy for subject matching by the subject
2225 id:

- 2226 ● The policy MAY match a subject by id using an <xacmlp:SubjectMatch> element
 - 2227 ○ The <xacmlp:SubjectMatch> element MUST have a <xacmlp:AttributeValue
2228 DataType="http://www.w3.org/2001/XMLSchema#string"> element whose content MUST be
2229 the unique id of the subject
 - 2230 ○ The <xacmlp:SubjectMatch> element MUST have a <xacmlp:SubjectAttributeDesignator
2231 AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
2232 DataType="http://www.w3.org/2001/XMLSchema#string"/> element

2233

2234 The following example shows a Subject that matches a registered user with id "urn:acme:person:Danyal":

```
2235 <Target>
2236   <Subjects>
2237     <Subject>
2238       <SubjectMatch
2239         MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
2240         <AttributeValue
2241           DataType="http://www.w3.org/2001/XMLSchema#string">
2242           urn:acme:person:Danyal
2243         </AttributeValue>
2244         <SubjectAttributeDesignator
2245           AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
2246           DataType="http://www.w3.org/2001/XMLSchema#string"/>
2247         </SubjectMatch>
2248       </Subject>
2249     </Subjects>
2250   </Target>
```

2251

2252 10.6.2 Matching Subject By Role

2253 The following requirements are defined for an Access Control Policy for subject matching by a subject
2254 role:

- 2255 ● The policy MAY match a subject by a contextual role using an <xacmlp:Condition> element
 - 2256 ○ The <xacmlp:Condition> element MUST use an
2257 <xacmlp:Apply/@FunctionId="urn:oasis:names:tc:ebxml-regrep:4.0:rim:acp:function:matches-
2258 role"> element to invoke the "matches-role" XACML function as defined by this specification

2259

2260 The following example shows a Subject that matches a subject role "ProjectLead" within the organizational
2261 context of Organization TestSubmittingOrg1 and register context of Register TestRegister1:

```
2262 <Rule Effect="Permit" RuleId="urn:test:customACP1:rule:restricted-delete">
2263   <Target/>
2264
2265   <Condition>
2266     <Apply FunctionId="urn:oasis:names:tc:ebxml-
2267     regrep:4.0:rim:acp:function:matches-role">
2268       <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:ebxml-
2269       regrep:3.0:rim:acp:subject:role"
2270       DataType="http://www.w3.org/2001/XMLSchema#string"/>
2271       <AttributeValue
2272       DataType="http://www.w3.org/2001/XMLSchema#string">urn:oasis:names:tc:ebxml-
2273       regrep:SubjectRole:ProjectLead</AttributeValue>
2274       <AttributeValue
2275       DataType="http://www.w3.org/2001/XMLSchema#string">urn:oasis:names:tc:ebxml-
2276       regrep:RoleAssociation:organizationContext</AttributeValue>
2277       <AttributeValue
2278       DataType="http://www.w3.org/2001/XMLSchema#string">urn:test:Organization:TestS
2279       ubmittingOrg1</AttributeValue>
2280       <AttributeValue
2281       DataType="http://www.w3.org/2001/XMLSchema#string">urn:oasis:names:tc:ebxml-
2282       regrep:RoleAssociation:registerContext</AttributeValue>
2283       <AttributeValue
2284       DataType="http://www.w3.org/2001/XMLSchema#string">urn:test:Register:TestRegis
2285       ter1</AttributeValue>
2286     </Apply>
2287   </Condition>
2288
2289 </Rule>
```

2290

2291 10.7 Resource Matching

2292 An XACML Access Control Policy MAY use the id associated with the resource as resource attributes
2293 within <xacml:ResourceMatch> elements to match a resource within an authorization decision for an
2294 action on the resource.

2295 The following requirements are defined for a server for resource matching:

- 2296 ● A server MUST specify the resource identifier in the request context for an XACML decision
2297 request using a <xacmlc:Request>/<xacmlc:Resource>/<xacmlc:Attribute> elementThe
2298 <xacmlc:Attribute> element MUST have an AttributeId attribute with value
2299 "urn:oasis:names:tc:xacml:1.0:resource:resource-id"
 - 2300 ○ The <xacmlc:Attribute> element MUST have a DataType attribute with value
2301 "http://www.w3.org/2001/XMLSchema#string"
 - 2302 ○ The <xacmlc:Attribute> MUST have a <xacmlc:AttributeValue> element whose content
2303 MUST be the unique id associated with the resource
- 2304 ● A server MUST specify the owner attribute value of the RegistryObjectType resource in the
2305 request context for an XACML decision request using a
2306 <xacmlc:Request>/<xacmlc:Resource>/<xacmlc:Attribute> element
 - 2307 ○ The <xacmlc:Attribute> element MUST have an AttributeId attribute with value
2308 "urn:oasis:names:tc:ebxml-regrep:3.0:rim:acp:resource:owner"
 - 2309 ○ The <xacmlc:Attribute> element MUST have a DataType attribute with value
2310 "http://www.w3.org/2001/XMLSchema#string"

- 2311 ○ The <xacmlc:Attribute> MUST have a <xacmlc:AttributeValue> element whose content MUST
2312 be the owner attribute value of the RegistryObjectType resource

2313

2314 10.7.1 Matching a Resource By Id

2315 The following requirements are defined for an Access Control Policy for resource matching by the
2316 resource's id:

- 2317 ● The policy MAY match a resource by id using an <xacmlp:ResourceMatch> element
 - 2318 ○ The <xacmlp:ResourceMatch> element MUST have a <xacmlp:AttributeValue
2319 DataType="http://www.w3.org/2001/XMLSchema#string"> element whose content MUST be
2320 the unique id of the resource
 - 2321 ○ The <xacmlp:ResourceMatch> element MUST have a <xacmlp:ResourceAttributeDesignator
2322 AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
2323 DataType="http://www.w3.org/2001/XMLSchema#string"/> element

2324

2325 The following example shows a ResourceMatch that matches a resource with id
2326 "urn:acme:person:Danyal":

```
2327 <Target>  
2328 <Resources>  
2329 <Resource>  
2330 <ResourceMatch  
2331 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">  
2332 <AttributeValue  
2333 DataType="http://www.w3.org/2001/XMLSchema#string">  
2334 urn:acme:person:Danyal  
2335 </AttributeValue>  
2336 <ResourceAttributeDesignator  
2337 AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"  
2338 DataType="http://www.w3.org/2001/XMLSchema#string"/>  
2339 </ResourceMatch>  
2340 </Resource>  
2341 </Resources>  
2342 </Target>
```

2343 10.7.2 Matching a Resource Using XPATH Expression

2344 An XACML Access Control Policy MAY use any node in the XML document representing a
2345 RegistryObjectType instance within an <xacml:ResourceMatch> element. In this case, the
2346 <xacml:ResourceMatch> element SHOULD use an XPATH expression to match any part of the XML
2347 element representing the RegistryObjectType instance.

2348 The following example uses XPATH expression to match resource if it has a Slot with name
2349 "someSlotName".

```
2350 <Resource>  
2351 <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">  
2352 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">  
2353 urn:oasis:names:tc:ebxml-regrep:xsd:rim:4.0  
2354 </AttributeValue>  
2355 <ResourceAttributeDesignator  
2356 AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"  
2357 DataType="http://www.w3.org/2001/XMLSchema#string"/>  
2358 </ResourceMatch>
```

```

2359 <ResourceMatch
2360   MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
2361   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
2362     //<rim:Slot>/@name="someSlotName"
2363   </AttributeValue>
2364   <ResourceAttributeDesignator
2365     AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
2366     DataType="http://www.w3.org/2001/XMLSchema#string"/>
2367   </ResourceMatch>
2368 </Resource>

```

2369

2370 10.8 Canonical XACML Functions

2371 Section A.3 of [XACML] defines a set of standard functions. This section defines addition XACML
 2372 functions that MUST be supported by an ebXML RegRep server that supports XACML based custom
 2373 access control policies. XACML specifies the following functions. If an argument of one of these functions
 2374 were to evaluate to "Indeterminate", then the function MUST be set to "Indeterminate".

2375 10.8.1 Function AssociationExists

2376 **Function ID:** urn:oasis:names:tc:ebxml-regrep:rim:acp:function:AssociationExists

2377

Parameter / Return	Name	Description	Data Type
Parameter 1	sourceObject	Specifies a value for the sourceObject attribute of AssociationType. MAY use '%' and '_' as wildcard to match multiple or single characters.	http://www.w3.org/2001/XMLSchema#string
Parameter 2	targetObject	Specifies a value for the targetObject attribute of AssociationType. MAY use '%' and '_' as wildcard to match multiple or single characters.	http://www.w3.org/2001/XMLSchema#string
Parameter 3	type	Specifies the path attribute value for a ClassificationNode in the AssociationType ClassificationScheme. MAY use '%' and '_' as wildcard to match multiple or single characters. This attribute is used to match the type attribute of AssociationType. The type parameter MUST also match ClassificationNodes that are descendants of ClassificationNode specified by the type parameter. This parameter is optional and MAY be omitted.	http://www.w3.org/2001/XMLSchema#string
Returns		MUST return "True" if and only if an AssociationType instance exists that matches the specified sourceObjectId, targetObjectId and type. MUST return "False" otherwise.	http://www.w3.org/2001/XMLSchema#boolean

2378

2379 10.8.2 Function ClassificationNodeCompare

2380 **Function ID:** urn:oasis:names:tc:ebxml-regrep:rim:acp:function:ClassificationNodeCompare

2381 A client MAY use this XACML function to test whether a resource's objectType attribute matches a
 2382 specific objectType or its sub-types.

2383

Parameter / Return	Name	Description	Data Type
Parameter 1	node1	Specifies the id of a ClassificationNode.	http://www.w3.org/2001/XMLSchema#string
Parameter 2	node2	Specifies the id of a ClassificationNode.	http://www.w3.org/2001/XMLSchema#string
Returns		MUST return "True" if and only if ClassificationNode with id matching node2 value is same as or descendent of if ClassificationNode with id matching node1. MUST return "False" otherwise.	http://www.w3.org/2001/XMLSchema#boolean

2384

2385 10.8.3 Function matches-role

2386 **Function ID:** urn:oasis:names:tc:ebxml-regrep:4.0:rim:acp:function:matches-role

2387

Parameter / Return	Name	Description	Data Type
Parameter 1	roles	Specifies a bag containing ids of RoleType instances representing the contextual roles that a subject is expected to have	Bag of attributes of type http://www.w3.org/2001/XMLSchema#string
Parameter 2	roleType	Specifies the id of a ClassificationNode within the canonical SubjectRole ClassificationScheme	http://www.w3.org/2001/XMLSchema#string
Subsequent parameters MUST come in pairs where each pair specifies a context key/value pair. Any number of such pairs MUST be supported by server implementing this function			
Parameter 3+N	contextKey	Specifies a context identifier	http://www.w3.org/2001/XMLSchema#string
4+N	contextValue	Specifies a context value associated with the context identifier specified by previous parameter	http://www.w3.org/2001/XMLSchema#string
Returns		MUST return "True" if and only if at least one RoleType instance assigned to the subject meets the following conditions: <ul style="list-style-type: none"> •If roleType is specified, then the type attribute of the RoleType instance MUST match the role type ClassificationNode (or a descendant of it) specified by the roleType parameter •If any context key/value pairs are specified then the RoleType instance MUST have a Slot whose name matches the context key and whose value matches the context value 	http://www.w3.org/2001/XMLSchema#boolean

	MUST return "False" otherwise.	
--	--------------------------------	--

2388

2389 **10.9 Constraints on XACML Binding**

2390 This specification normatively defines the following constraints on the binding of the Access Control Model
2391 to [XACML]. These constraints MAY be relaxed in future versions of this specification.

- 2392 ● All Policy and PolicySet definitions MUST reside within an ebXML Registry as RepositoryItems.

2393

2394 **10.10 Resolving Policy References**

2395 An XACML PolicySet MAY reference XACML Policy objects defined outside the repository item containing
2396 the XACML PolicySet. A server implementation MUST be able to resolve such references. To resolve
2397 such references efficiently a server SHOULD be able to find the repository item containing the referenced
2398 Policy without having to load and search all Access Control Policies in the repository. This section
2399 describes the normative behavior that enables a server to resolve policy references efficiently.

2400 A server SHOULD define a Content Cataloging Service for the canonical XACML PolicySet objectType.
2401 The PolicySet cataloging service MUST automatically catalog every PolicySet upon submission to contain
2402 a special Slot with name ComposedPolicies. The value of this Slot MUST be a Set where each element in
2403 the Set is the id for a Policy object that is composed within the PolicySet.

2404 Thus a server is able to use an ad hoc query to find the repositoryItem representing an XACML PolicySet
2405 that contains the Policy that is being referenced by another PolicySet.

2406