



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

Committee Specification 01

28 July 2011

Specification URIs:

This version:

<http://docs.oasis-open.org/regrep/regrep-core/v4.0/cs01/regrep-core-rim-v4.0-cs01.odt>

(Authoritative)

<http://docs.oasis-open.org/regrep/regrep-core/v4.0/cs01/regrep-core-rim-v4.0-cs01.html>

<http://docs.oasis-open.org/regrep/regrep-core/v4.0/cs01/regrep-core-rim-v4.0-cs01.pdf>

Previous version

<http://docs.oasis-open.org/regrep/regrep-core/v4.0/csprd01/regrep-core-rim-v4.0-csprd01.odt>

(Authoritative)

<http://docs.oasis-open.org/regrep/regrep-core/v4.0/csprd01/regrep-core-rim-v4.0-csprd01.pdf>

<http://docs.oasis-open.org/regrep/regrep-core/v4.0/csprd01/regrep-core-rim-v4.0-csprd01.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.html>

<http://docs.oasis-open.org/regrep/regrep-core/v4.0/regrep-core-rim-v4.0.pdf>

Technical Committee:

OASIS ebXML Registry TC

Chairs:

Kathryn Breininger (Kathryn.r.Breininger@boeing.com), Boeing

Farrukh Najmi (farrukh@wellfleetsoftware.com), Wellfleet Software

Editors:

Farrukh Najmi, (farrukh@wellfleetsoftware.com), Wellfleet Software

Nikola Stojanovic (nikola.stojanovic@acm.org), 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). 28 July 2011. OASIS Committee Specification 01. <http://docs.oasis-open.org/regrep/regrep-core/v4.0/cs01/regrep-core-rim-v4.0-cs01.html>.

Notices

Copyright © OASIS Open 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.....	11
1.1	Terminology.....	11
1.2	XML Schema.....	11
1.3	Information Model Types: Inheritance View.....	11
1.4	Extending ebRIM.....	12
1.5	Canonical Classification Schemes.....	13
1.6	Document Organization.....	13
2	Core Information Model.....	14
2.1	InternationalStringType.....	14
2.1.1	Syntax.....	14
2.1.2	Example.....	15
2.1.3	Description.....	15
2.2	LocalizedStringType.....	15
2.2.1	Syntax.....	15
2.2.2	Example.....	15
2.2.3	Description.....	15
2.3	ExtensibleObjectType.....	16
2.3.1	Syntax.....	16
2.3.2	Example.....	16
2.3.3	Description.....	16
2.4	SlotType.....	16
2.4.1	Syntax.....	16
2.4.2	Example.....	17
2.4.3	Description.....	17
2.5	ValueType.....	17
2.5.1	Syntax.....	17
2.5.2	Description.....	17
2.6	IdentifiableObjectType.....	18
2.6.1	Syntax.....	18
2.6.2	Example.....	19
2.6.3	Description.....	19
2.7	RegistryObjectType.....	19
2.7.1	Syntax.....	19
2.7.2	Description.....	19
2.8	VersionInfoType.....	21
2.8.1	Syntax.....	21

2.8.2	Example.....	22
2.8.3	Description.....	22
2.9	objectReferenceType.....	22
2.9.1	Syntax.....	22
2.9.2	Example.....	23
2.9.3	Description.....	23
2.9.3.1	Local and Remote References	23
2.9.3.2	Static and Dynamic References.....	23
2.9.3.3	Encoding of objectReferenceType.....	23
2.10	ObjectRefType.....	24
2.10.1	Syntax.....	24
2.10.2	Description.....	25
2.11	DynamicObjectRefType.....	25
2.11.1	Syntax.....	25
2.11.2	Description.....	25
2.12	ExtrinsicObjectType.....	26
2.12.1	Syntax.....	26
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.....	28
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
4.3.3.1	Example of Canonical Path Representation.....	39
4.4	ClassificationType.....	39
4.4.1	Syntax.....	39
4.4.2	Example.....	40
4.4.3	Description.....	40
5	Provenance Information Model.....	41
5.1	PostalAddressType.....	41
5.1.1	Syntax.....	41
5.1.2	Example.....	42
5.1.3	Description.....	42
5.2	TelephoneNumberType.....	42
5.2.1	Syntax.....	42
5.2.2	Example.....	43
5.2.3	Description.....	43
5.3	EmailAddressType.....	43
5.3.1	Syntax.....	43
5.3.2	Example.....	44
5.3.3	Description.....	44
5.4	PartyType.....	44
5.4.1	Syntax.....	44
5.4.2	Description.....	44

5.5	PersonType	45
5.5.1	Syntax	45
5.5.2	Example	45
5.5.3	Description	45
5.6	PersonNameType	46
5.6.1	Syntax	46
5.6.2	Example	46
5.6.3	Description	46
5.7	OrganizationType	46
5.7.1	Syntax	46
5.7.2	Example	47
5.7.3	Description	47
5.8	Associating Organization With Persons	47
5.9	Associating Organization With Organizations	47
5.10	Associating Organizations With RegistryObjects	48
6	Service Information Model	49
6.1	ServiceType	49
6.1.1	Syntax	49
6.1.2	Example	49
6.1.3	Description	50
6.2	ServiceEndpointType	50
6.2.1	Syntax	50
6.2.2	Example	50
6.2.3	Description	50
6.3	ServiceBindingType	51
6.3.1	Syntax	51
6.3.2	Example	51
6.3.3	Description	51
6.4	ServiceInterfaceType	51
6.4.1	Syntax	51
6.4.2	Example	52
6.4.3	Description	52
7	Query Information Model	53
7.1	QueryDefinitionType	53
7.1.1	Syntax	53
7.1.2	Example	54
7.1.3	Description	54
7.2	ParameterType	54

7.2.1	Syntax.....	54
7.2.2	Example.....	55
7.2.3	Description.....	55
7.3	QueryExpressionType.....	56
7.3.1	Syntax.....	56
7.3.2	Description.....	56
7.4	StringQueryExpressionType.....	56
7.4.1	Syntax.....	56
7.4.2	Example.....	57
7.4.3	Description.....	57
7.5	XMLQueryExpressionType.....	57
7.5.1	Syntax.....	57
7.5.2	Example.....	57
7.5.3	Description.....	58
7.6	QueryType.....	58
7.6.1	Syntax.....	58
7.6.2	Example.....	58
7.6.3	Description.....	58
8	Event Information Model.....	59
8.1	AuditableEventType.....	59
8.1.1	Syntax.....	60
8.1.2	Example.....	60
8.1.3	Description.....	60
8.2	ActionType.....	61
8.2.1	Syntax.....	61
8.2.2	Description.....	61
8.3	SubscriptionType.....	61
8.3.1	Syntax.....	62
8.3.2	Example.....	62
8.3.3	Description.....	62
8.4	DeliveryInfoType.....	63
8.4.1	Syntax.....	63
8.4.2	Description.....	64
8.5	NotificationType.....	64
8.5.1	Syntax.....	64
8.5.2	Example.....	65
8.5.3	Description.....	65
9	Federation Information Model.....	66

9.1	Federation Configuration	66
9.2	RegistryType	66
9.2.1	Syntax	67
9.2.2	Example	67
9.2.3	Description	67
9.3	FederationType	68
9.3.1	Syntax	68
9.3.2	Example	68
9.3.3	Description	69
10	Access Control Information Model	70
10.1	Defining an Access Control Policy	71
10.2	Assigning Access Control Policy to a RegistryObject	71
10.2.1	Default Access Control Policy for a RegistryObject	71
10.2.2	Access Control Policy Inheritance	72
10.2.2.1	Algorithm for Getting Applicable Access Control Policy	72
10.2.3	Performance Implications	72
10.3	Defining a Contextual Role	72
10.3.1	RoleType	72
10.3.1.1	Syntax	72
10.3.2	Example	73
10.3.3	Description	73
10.4	Assigning a Contextual Role to a Subject	73
10.5	Action Matching	74
10.5.1	Action Attribute: reference-source	75
10.5.2	Action Attribute: reference-source-attribute	75
10.6	Subject Matching	75
10.6.1	Matching Subjects By Id	75
10.6.2	Matching Subject By Role	76
10.7	Resource Matching	77
10.7.1	Matching a Resource By Id	77
10.7.2	Matching a Resource Using XPATH Expression	78
10.8	Canonical XACML Functions	78
10.8.1	Function AssociationExists	78
10.8.2	Function ClassificationNodeCompare	79
10.8.3	Function matches-role	79
10.9	Constraints on XACML Binding	80
10.10	Resolving Policy References	80

Illustration Index

Illustration 1: Information Model Inheritance View.....	12
Illustration 2: Core Information Model.....	14
Illustration 3: Association Example.....	32
Illustration 4: Classification Example.....	35
Illustration 5: Classification Information Model.....	36
Illustration 6: Provenance Information Model.....	41
Illustration 7: Service Information Model.....	49
Illustration 8: Query Information Model.....	53
Illustration 9: Event Information Model.....	59
Illustration 10: Federation Information Model.....	66
Illustration 11: Assigning Access Control Policy to a RegistryObject.....	71

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.

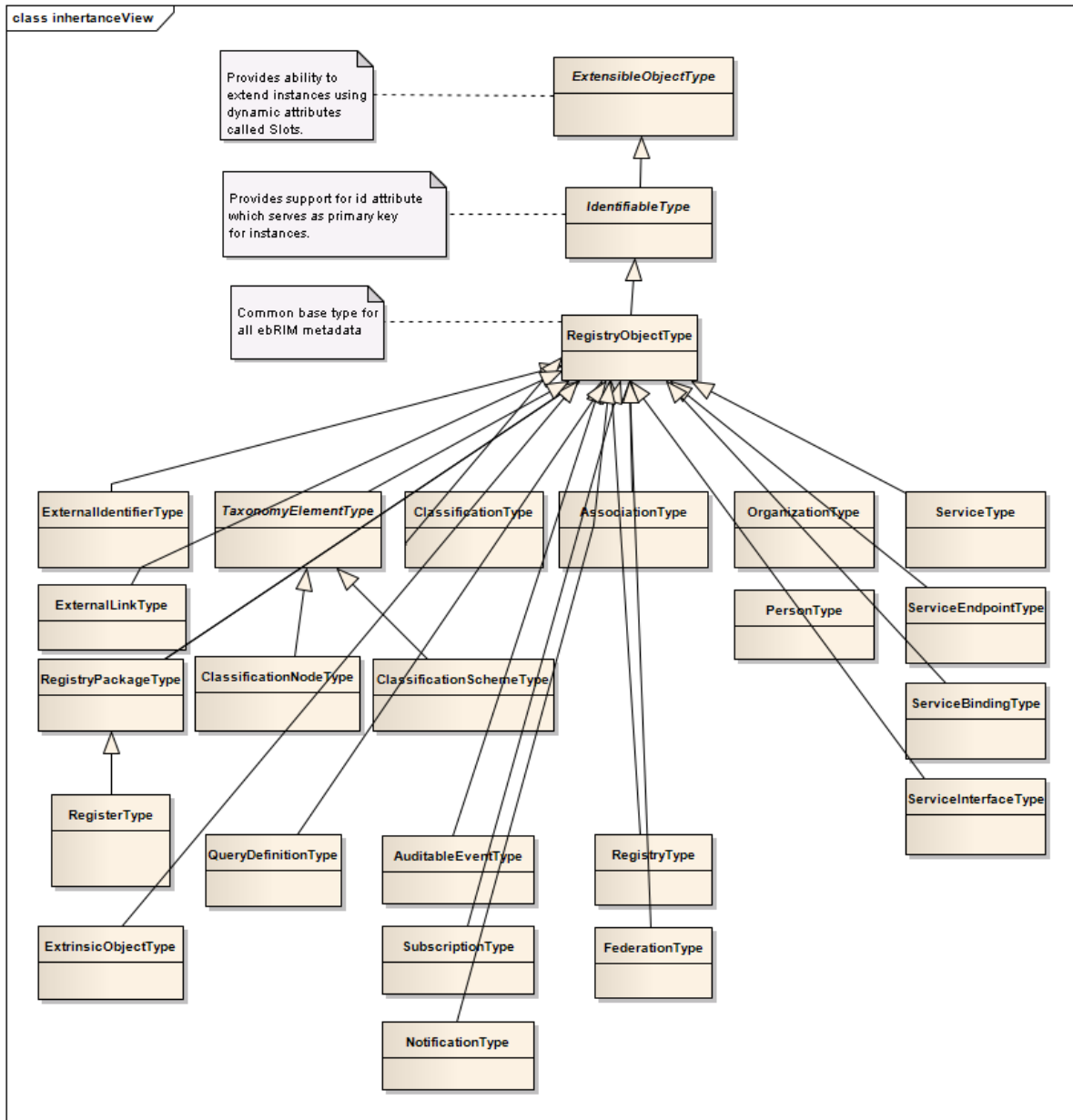


Illustration 1: Information Model Inheritance View

22 1.4 Extending ebRIM

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

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

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

- 28 ● The server protocols as defined by [regrep-rs-v4.0] MUST support extended types in a manner equivalent
29 to pre-defined types. Specifically they MUST support submit, update, versioning and removal of extended
30 types derived directly or indirectly from RegistryObjectType

- 31 ● The server MUST be able to faithfully persist instances of extended types including all extension attributes
32 and elements without any information loss
- 33 ● The server MUST be able to faithfully return instances of extension types including extension attributes
34 and elements within a query response without any information loss
- 35 ● This specification does not prescribe how a server may support the addition of new extension types to the
36 server

37 1.5 Canonical ClassificationSchemes

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

40 This specification uses several standard ClassificationSchemes referred to as *canonical ClassificationS-*
41 *schemes*. The values defined within canonical ClassificationSchemes are defined using standard Classific-
42 ationNodes that are referred to as *canonical ClassificationNodes*.

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

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

51 1.6 Document Organization

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

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

66 The remainder of this document will describe each of the above related group of information model types in a dedic-
67 ated chapter named accordingly.

68 2 Core Information Model

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

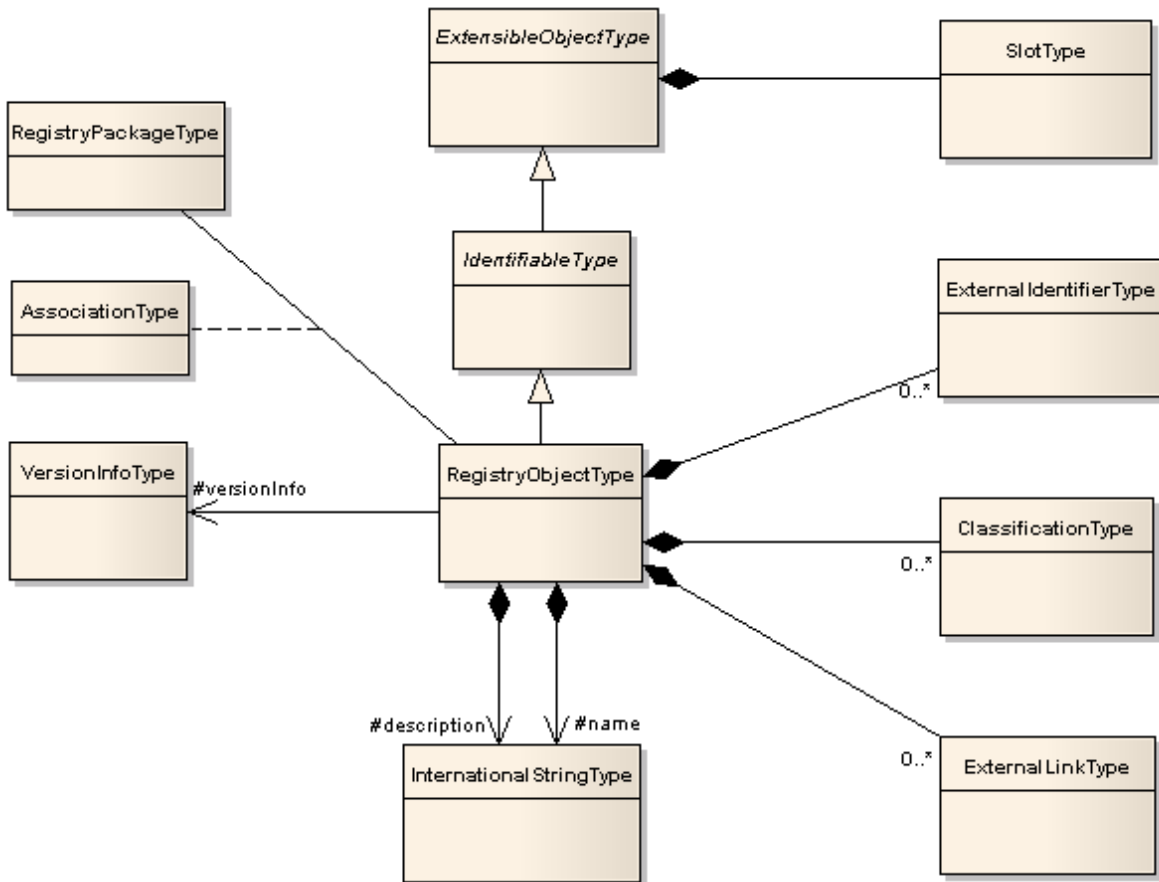


Illustration 2: Core Information Model

72 2.1 InternationalStringType

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

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

77 2.1.1 Syntax

```
<complexType name="InternationalStringType">
  <sequence>
    <element name="LocalizedString" type="tns:LocalizedString"
      minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>
```

78 2.1.2 Example

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

79 2.1.3 Description

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

80

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

83 2.2 LocalizedStringType

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

88

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

90

91 2.2.1 Syntax

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

92 2.2.2 Example

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

93 2.2.3 Description

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

94

- 95 ● Attribute xml:lang - Each LocalizedStringType instance MAY have a *xml:lang* attribute that specifies the
96 language used by that LocalizedStringType instance. The xml:lang attribute and legal values for it are
97 defined by [XML].
- 98 ● Attribute value - Each LocalizedStringType instance MUST have a *value* attribute that specifies the string
99 value used by that LocalizedStringType instance

100 2.3 ExtensibleObjectType

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

103 2.3.1 Syntax

```
<complexType name="ExtensibleObjectType" abstract="true">
  <sequence>
    <element name="Slot" type="tns:SlotType" minOccurs="0"
      maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

104 2.3.2 Example

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

107

```
<rim:RegistryObject xsi:type="rim:OrganizationType"
  id="urn:freebxml:registry:Organization:freebXMLRegistry" ...>
  <rim:Slot name="urn:foo:slot:taxPayerId">
    <rim:SlotValue xsi:type="rim:StringValueType">
      <rim:Value>1234567890</rim:Value>
    </rim:SlotValue>
  </rim:Slot>
  ...
</rim:RegistryObject>
```

108 2.3.3 Description

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

109

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

111 2.4 SlotType

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

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

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

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

119 2.4.1 Syntax

```
<complexType name="SlotType">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <sequence>
        <element name="SlotValue" type="tns:ValueType"

```



```

        minOccurs="0" maxOccurs="1"/>
    </sequence>
    <attribute name="name" type="tns:LongText" use="required"/>
    <attribute name="type" type="tns:LongText" use="optional"/>
</extension>
</complexContent>
</complexType>

```

120 2.4.2 Example

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

```

<rim:Slot
  name="geographicBoundingBox"
  type="urn:ogc:def:dataType:ISO-19107:GM_Geometry">
  <rim:SlotValue xsi:type="rim:AnyValueType">
    <gml:Envelope srsName="urn:ogc:def:crs:OGC:2:WGS84">
      <!--BB: POLYGON((0 0, 30 0, 30 30, 0 30, 0 0))-->
      <gml:lowerCorner>0 0</gml:lowerCorner>
      <gml:upperCorner>30 30</gml:upperCorner>
    </gml:Envelope>
  </rim:SlotValue>
</rim:Slot>

```

122 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

123

- 124 ● Attribute name – The name of this SlotType instance. The name of the slot MUST be unique within the uni-
125 verse of slot names for all sibling slots within its parent object
- 126 ● Element SlotValue – This element is the container for the actual value for the SlotType instance
- 127 ● Attribute type – A string that specifies the type for the SlotType instance. The type may be used to assign a
128 category for the SlotType instance

129 2.5 ValueType

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

131 2.5.1 Syntax

```

<complexType name="ValueType" abstract="true">
</complexType>

```

132 2.5.2 Description

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

- 135 ● AnyValueType – This concrete sub-type of ValueType is used as a container for any well-formed XML ele-
136 ment value in any namespace

- 137 ● BooleanValueType - This concrete sub-type of ValueType is used as a container for a boolean value
- 138 ● CollectionValueType - This concrete sub-type of ValueType is used as a container for a collection of values.
- 139 It may be used to represent a SlotValue that is a collection of values where each value is represented by a
- 140 ValueType instance
- 141 ○ Attribute collectionType – Defines the type of collection for the CollectionValueType. Must be an ob-
- 142 jectReferenceType that references a ClassificationNode in the canonical ClassificationScheme Collec-
- 143 tionTypeScheme. A server MUST enforce the following semantics associated with the following ca-
- 144 nonical collection types:
 - 145 ■ List – Server MUST maintain the order of the values in the collection
 - 146 ■ Set – Server MUST NOT allow duplicate values in the collection
 - 147 ■ Sorted Set – Server MUST NOT allow duplicate values in the collection and MUST maintain a
 - 148 sort order according to the alphanumeric ordering of its elements according to the default locale
 - 149 associated with the server
 - 150 ■ Bag – Server MUST allow duplicate values and MAY not maintain order of values
- 151 ● DateTimeValueType - This concrete sub-type of ValueType is used as a container for a dateTime value
- 152 ● DurationValueType - This concrete sub-type of ValueType is used as a container for a duration value
- 153 ● FloatValueType - This concrete sub-type of ValueType is used as a container for a float value
- 154 ● IntegerValueType - This concrete sub-type of ValueType is used as a container for an integer value
- 155 ● InternationalStringValueType - This concrete sub-type of ValueType is used as a container for an Interna-
- 156 tionalString value capable of holding strings in multiple locales
- 157 ● MapValueType - This concrete sub-type of ValueType is used as a container for a map value. A map con-
- 158 sists of Entry sub-elements where each Entry consists of an EntryKey and EntryValue both of which are of
- 159 type ValueType
- 160 ● SlotValueType – This concrete sub-type of ValueType is used as a container for a SlotType value
- 161 ● StringValueType – This concrete sub-type of ValueType is used as a container for a string value
- 162 ● VocabularyTermValueType - This concrete sub-type of ValueType is used as a container for a Vocabula-
- 163 ryTermType value. It is used to reference a term in some externally defined coded vocabulary (e.g. Dublin
- 164 Core)

165 2.6 IdentifiableObjectType

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

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

168 2.6.1 Syntax

```
<complexType name="IdentifiableType" abstract="true">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <attribute name="id" type="string" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

169 **2.6.2 Example**

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

170 **2.6.3 Description**

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

171

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

173 **2.7 RegistryObjectType**

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

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

177 **2.7.1 Syntax**

```
<complexType name="RegistryObjectType">
  <complexContent>
    <extension base="tns:IdentifiableType">
      <sequence>
        <element name="Name" type="tns:InternationalStringType"
          minOccurs="0" maxOccurs="1"/>
        <element name="Description" type="tns:InternationalStringType"
          minOccurs="0" maxOccurs="1"/>
        <element name="VersionInfo" type="tns:VersionInfoType" minOccurs="0"
          maxOccurs="1"/>
        <element name="Classification" type="tns:ClassificationType"
          minOccurs="0" maxOccurs="unbounded"/>
        <element name="ExternalIdentifier" type="tns:ExternalIdentifierType"
          minOccurs="0" maxOccurs="unbounded" />
        <element name="ExternalLink" type="tns:ExternalLinkType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="lid" type="string" use="optional"/>
      <attribute name="objectType" type="tns:objectReferenceType"
        use="optional"/>
      <attribute name="owner" type="string" use="optional"/>
      <attribute name="status" type="tns:objectReferenceType" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

178 **2.7.2 Description**

Node	Type	Cardinality	Default Value	Specified By	Mutable
Classification	Classifica- tionType	0..*		Client	Yes
Description	International-	0..1		Client	Yes

	StringType				
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

179

180

181

182

- Element Classification - A RegistryObjectType instance MAY have zero or more ClassificationType instances that are composed within the RegistryObject. A ClassificationType instance classify the Registry-Object using a value within a ClassificationScheme

183

184

185

- Element Description - A RegistryObjectType instance MAY have textual description in a human readable and user-friendly form. This element is of type InternationalStringType and therefor capable of containing **textual values in multiple local languages** and character sets.

186

187

188

- Element ExternalIdentifier - A RegistryObjectType instance MAY have zero or more ExternalIdentifier instances that are composed within the RegistryObject. An ExternalIdentifier instance represents an alternate identifier for the RegistryObject in addition to the identifier specified by its id attribute value.

189

190

- Attribute lid - A RegistryObjectType instance MUST have a lid (Logical Id) attribute. The lid is used to refer to a logical RegistryObject in a version independent manner.

191

192

- All versions of a RegistryObject MUST have the same value for the lid attribute. Note that this is in contrast with the id attribute that MUST be unique for each version of the same logical RegistryObject.

193

194

- The lid attribute MUST be specified by the client when creating the original version of a RegistryObject.

195

196

- The lid attribute specified when submitting the original version of a RegistryObject MUST be globally unique and MUST NOT be already in use as lid by another object.

197

198

199

200

- Element Name - A RegistryObjectType instance MAY have a human readable name. The name does not need to be unique with respect to other RegistryObjectType instances. This element is of type InternationalStringType and therefor capable of containing **textual values in multiple local languages** and character sets.

201

- Attribute objectType - A RegistryObjectType instance has an *objectType* attribute.

202

203

- The value of the objectType attribute MUST be a reference to a ClassificationNode in the canonical ObjectType ClassificationScheme.

- 204 ○ A server MUST support the object types as defined by the canonical ObjectType ClassificationScheme.
 205 The canonical ObjectType ClassificationScheme may easily be extended by adding additional Classi-
 206 ficationNodes to the canonical ObjectType ClassificationScheme.
- 207 ○ The *objectType* attribute MUST be assigned by the server for all RegistryObjectType instances that are
 208 not instances of ExtrinsicObjectType.
- 209 ○ The *objectType* attribute MAY be assigned by the client for all RegistryObjectType instances that are
 210 instances of ExtrinsicObjectType
- 211 ○ If the client does not specify an objectType for an ExtrinsicObject then the server MUST set its value
 212 to the id of the ClassificationNode representing ExtrinsicObject within the canonical ObjectType Clas-
 213 sificationScheme.
- 214 ○ A server MUST set the correct objectType on a RegistryObject when returning it as a response to a cli-
 215 ent request.
- 216 ● Attribute owner – Specifies the identifier associated with the registered user that owns the RegistryObject-
 217 Type instance. It is used for **access control** and may be referenced within custom access control policies.
- 218 ● Attribute status - A RegistryObjectType instance MUST have a life cycle status indicator. The status is as-
 219 signed by the server. Profiles MAY define additional status values if needed as slots on the RegistryObject-
 220 Type instance. Such slots SHOULD have a type attribute with value “urn:oasis:names:tc:ebxml-
 221 regrep:rim:Slot:type:status”.
- 222 ○ A server MUST set the correct status on a RegistryObject when returning it as a response to a client re-
 223 quest.
- 224 ○ A client SHOULD NOT set the status on a RegistryObject when submitting the object as this is the re-
 225 sponsibility of the server.
- 226 ○ A server MUST ignore the status on a RegistryObject when it is set by the client during submission or
 227 update of the object.
- 228 ○ The value of the status attribute SHOULD be a reference to a ClassificationNode in the canonical
 229 StatusType ClassificationScheme.
- 230 ○ A Registry MUST support the status types as defined by the StatusType ClassificationScheme. The ca-
 231 nonical StatusType ClassificationScheme MAY easily be extended by adding additional Classification-
 232 Nodes to the canonical StatusType ClassificationScheme.
- 233 ● Element VersionInfo - Provides information about the specific version of a RegistryObjectType instance.
 234 The VersionInfo element is set by the server.
- 235 ○ A server MUST set a VersionInfo element for a RegistryObjectType instance. The VersionInfo element
 236 MUST contain a versionName attribute whose value MUST be unique for all versions of that logical
 237 RegistryObjectType.

238 2.8 VersionInfoType

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

242 2.8.1 Syntax

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

```
</complexType>
```

243 2.8.2 Example

```
<rim:RegistryObject xsi:type="rim:OrganizationType" ...>
  ...
  <rim:VersionInfo versionName="1.1" userVersionName="1.1"/>
  ...
</rim:RegistryObject>
```

244 2.8.3 Description

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

245

246 ● Attribute userVersionName - Represents a client-specified version name associated with the VersionInfo for
247 a specific RegistryObject version

248 ○ A client MAY directly provide a value for the userVersionName attribute when submitting or updating
249 an object

250 ○ A server MUST persist any client specified userVersionName for an object without altering it in any
251 form

252 ● Attribute versionName - Represents the registry assigned version name identifying the VersionInfo for a
253 specific RegistryObject version.

254 ○ The value for this attribute SHOULD NOT be specified by the client

255 ○ A server MAY silently ignore the value for this attribute if specified by the client

256 ○ The value for this attribute MUST be automatically generated by the server and MUST be defined for
257 RegistryObjectType instances returned by server responses. The server is free to choose any scheme
258 for generating the value for this attribute as long as the value is uniquely identifies a version for objects
259 that have the same lid attribute value.

260 2.9 objectReferenceType

261 **Base Type:** xs:string

262 A RegistryObjectType instance typically has several references to other RegistryObjectType instances. These refer-
263 ences are represented by attributes of type rim:objectReferenceType within the XML Schema for ebXML RegRep.

264 The RegistryObjectType instance that has a reference to another RegistryObjectType instance is referred to as the
265 *reference source* object. The RegistryObjectType instance that is being referenced is referred to as the *reference tar-*
266 *get* object.

267 2.9.1 Syntax

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

268 2.9.2 Example

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

269

270 2.9.3 Description

271 2.9.3.1 Local and Remote References

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

274 2.9.3.2 Static and Dynamic References

275 When a reference is fixed to a specific reference target it is referred to as a *static reference*. This specification also
276 supports a *dynamic reference* where the reference target is determined dynamically by a query at the time the refer-
277 ence is resolved. Such a reference is referred to as a *dynamic reference*.

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

280 2.9.3.3 Encoding of objectReferenceType

281 A client MUST specify values for reference attributes of type objectReferenceType to be encoded as described be-
282 low:

- 283 ● A static reference to a local reference target SHOULD be encoded as the value of the id attribute of the ref-
284 erence target.
285 The following example shows the reference attribute named primaryContact within Organization element.
286 Its value is the value of the id attribute of a Person element.

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

<rim:RegistryObject xsi:type="rim:PersonType" id="urn:acme:person:Danyal" ...>
  ...
</rim:RegistryObject>
```

288

- 289 ● A dynamic reference to a local reference target SHOULD be encoded to contain the id of a DynamicOb-
290 jectRefType instance. The reference target is determined by the singleton result returned by the Query
291 within the DynamicObjectRef instance.

292
293 The following example shows the reference attribute named primaryContact within Organization element.
294 Its value is the value of the *id* attribute of a DynamicObjectRefType instance. The DynamicObjectRefType
295 instance has a *Query* that gets the latest version of the object identified by the *lid* parameter of the Query.
296 The query when invoked matches the latest version of the Person object representing Danyal.

```
<rim:RegistryObject xsi:type="rim:OrganizationType"
  primaryContact="urn:acme:dynamicRef:LatestVersionOfDanyal" ...>
  ...
  ...
```

```

</rim:RegistryObject>

<rim:ObjectRef xsi:type="rim:ObjectRefType"
  id="urn:acme:dynamicRef:LatestVersionOfDanyal">
  <rim:Query queryDefinition="urn:acme:QueryDefinition:FindLatestVersion">
    <rim:Slot name="lid">
      <rim:SlotValue xsi:type="rim:StringValueType">
        <rim:Value>urn:acme:person:Danyal</rim:Value>
      </rim:SlotValue>
    </rim:Slot>
  </rim:Query>
</rim:ObjectRef>

<rim:RegistryObject xsi:type="rim:PersonType"
  lid="urn:acme:person:Danyal" id="urn:acme:person:Danyal:1.8"...>
  <!-- latest version of object with lid "urn:acme:person:Danyal" -->
  ...
</rim:RegistryObject>

```

298

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

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

303

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

```

<!-- Following object is in local server -->
<rim:RegistryObject xsi:type="rim:OrganizationType"
  primaryContact="http://www.remoteRegistry.com/query?
id=urn:remoteServer:person:Danyal" ...>
  ...
  ...
</rim:RegistryObject>

<!-- Following object is in a remote server -->
<rim:RegistryObject xsi:type="rim:PersonType"
  id="urn:remoteServer:person:Danyal" ...>
  ...
</rim:RegistryObject>

```

308

309 2.10 ObjectRefType

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

311 This type represents an object reference as does the objectReferenceType. However, the two are used in differ-
312 ent situations. The objectReferenceType is used as the type for all reference attributes in ebRIM. The ObjectRefType
313 is used as type for elements rather than attributes. This type is used when there is a need to have multiple object ref-
314 erences within a schema type. An example of this is the ObjectRefList element which is used in several places in the
315 schema where a list of references to RegistryObjectType instances are needed.

316 2.10.1 Syntax

```

<complexType name="ObjectRefType">
  <complexContent>

```



```

<extension base="tns:ExtensibleObjectType">
  <attribute name="id" type="tns:objectReferenceType" use="required"/>
</extension>
</complexContent>
</complexType>

<complexType name="ObjectRefListType">
  <sequence>
    <element name="ObjectRef"
      type="tns:ObjectRefType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>
<element name="ObjectRefList" type="tns:ObjectRefListType"/>

```

317 2.10.2 Description

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

318

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

321 2.11 DynamicObjectRefType

322 **Base Type:** ObjectRefType

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

325 2.11.1 Syntax

```

<complexType name="DynamicObjectRefType">
  <complexContent>
    <extension base="tns:ObjectRefType">
      <sequence>
        <element name="Query" type="tns:QueryType"
          minOccurs="1" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

```

326 2.11.2 Description

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

327

- 328 ● Element Query – Specifies the query that MUST be invoked in order to determine the reference target.
- 329 ○ This query MUST match zero or one RegistryObjectType instances.
- 330 ○ When the query matches zero RegistryObjectType instances, the dynamic object reference is con-
331 sidered to be unresolved.

- 332 ○ A server MUST return a ConfigurationException fault message if the query matches more than 1 Re-
 333 gistryObjectType instances.

334 2.12 ExtrinsicObjectType

335 **Base Type:** RegistryObjectType

336 This type is a common base type for new extended types defined by profiles of ebRIM or by clients. The Extrinsic-
 337 icObjectType also allows arbitrary content to be associated with it. Such arbitrary content is referred to as a Reposit-
 338 ory Item.

339 2.12.1 Syntax

```
<complexType name="ExtrinsicObjectType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="ContentVersionInfo" type="tns:VersionInfoType"
          minOccurs="0" maxOccurs="1"/>
        <choice minOccurs="0" maxOccurs="1">
          <element name="RepositoryItemRef" type="tns:SimpleLinkType"/>
          <element name="RepositoryItem"
            xmlns:xmime="expectedContentTypes=*>*/*" type="base64Binary">
        </choice>
      </sequence>
      <attribute name="mimeType" type="tns:LongText" use="optional" />
    </extension>
  </complexContent>
</complexType>
```

340 2.12.2 Example

```
<rim:RegistryObject xsi:type="rim:ExtrinsicObjectType" mimeType="text/xml"
  objectType="urn:freebxml:registry:sample:profile:cpp:objectType:coppa:CPP"
  lid="urn:freebxml:registry:sample:profile:cpp:instance:cppl"
  id="urn:freebxml:registry:sample:profile:cpp:instance:cppl" >
  <ContentVersionInfo versionName="311" userVersionName="1.1"/>
  <RepositoryItem>...binary encoding of repository item</RepositoryItem>
</rim:RegistryObject>
```

341 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

342

- 343 ● Element ContentVersionInfo - Provides information about the specific version of a RepositoryItem that is
 344 associated with an ExtrinsicObjectType instance. The ContentVersionInfo element is set by the server.

- 345 ○ A server MUST NOT set a ContentVersionInfo element for an ExtrinsicObjectType instance that does
 346 not have a RepositoryItem.

- 347 ○ A server MUST set a ContentVersionInfo element for an ExtrinsicObjectType instance that has a Re-
 348 positoryItem. The ContentVersionInfo element MUST contain a versionName attribute whose value
 349 MUST be unique for all versions of that RepositoryItem.
- 350 ● Attribute mimeType - An ExtrinsicObjectType instance MAY have a mimeType attribute defined. The mi-
 351 meType provides information on the type of repository item cataloged by the ExtrinsicObject instance. The
 352 value of this attribute SHOULD be a registered MIME media type at
 353 <http://www.iana.org/assignments/media-types>.
- 354 ● Element repositoryItem – Provides a base64 binary encoded representation of the repository item associ-
 355 ated with the ExtrinsicObjectType instance (if any).
- 356 ● Element repositoryItemRef – This element MAY be specified as an alternative to the repositoryItem ele-
 357 ment. Its type is SimpleLinkType. It uses xlink:simpleAttrs to specify a reference to a file on the client's
 358 local file system. This provides client libraries an alternative way to specify local files as repository item.
 359 The client library MUST convert a repositoryItemRef element to a repositoryItem element prior to submit-
 360 ting it to the server.

361 2.13 CommentType

362 Extends: [ExtrinsicObjectType](#)

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

366 2.13.1 Syntax

```
<complexType name="CommentType">
  <complexContent>
    <extension base="tns:ExtrinsicObjectType">
    </extension>
  </complexContent>
</complexType>
```

367 2.13.2 Example

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

368 2.13.3 Description

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

- 370 ● An authorized client MAY attach one or more comments to any RegistryObjectType instance using an As-
 371 sociation between the RegistryObjectType instance and the CommentType instance
- 372 ○ Since a CommentType is itself a RegistryObjectType, a client MAY attach one or more comments to
 373 any CommentType instance
- 374 ● The type of the Association MUST reference the canonical HasComment ClassificationNode within the Ca-
 375 nonical AssociationType ClassificationScheme
- 376 ● The sourceObject of the Association MUST be the RegistryObjectType instance

377 ● The targetObject of the Association MUST be the CommentType instance

378 2.14 RegistryPackageType

379 Extends: RegistryObjectType

380 This type allows for grouping of related RegistryObjectType instances. It serves a similar role as a folder in the fa-
381 miliar file-folder metaphor available in most operating systems.

- 382 ● A RegistryObjectType instance MAY be a member of multiple RegistryPackageType instances.
- 383 ● A RegistryPackageType instance MAY have multiple RegistryObjectType instances as its members.
- 384 ● Membership of a RegistryObjectType instance in a RegistryPackageType instance is established via an
385 AssociationType instance where the type attribute references the canonical “HasMember” AssociationType
386 within the canonical AssociationTypeScheme ClassificationScheme.
- 387 ● As a convenience, the RegistryPackageType allows a RegistryObjectList to be specified by the client as a
388 sub-element during submission of a RegistryPackage. The RegistryObjectList contains the set of Registry-
389 ObjectType instances that are members of the RegistryPackageType instance.

390 2.14.1 Syntax

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

391 2.14.2 Example

392 The following example shows the use of a RegistryObjectList to specify the members of a RegistryPackageType in-
393 stance during submission.

```
<rim:RegistryObject xsi:type="rim:RegistryPackageType"
  id="urn:acme:RegistryPackage:photos" ...>
  ...
  <rim:RegistryObjectList>
    <rim:RegistryObject xsi:type="rim:ExtrinsicObjectType"
      mimeType="image/jpeg" id="urn:acme:RegistryPackage:photos:summer-
      2008:wellfleet-beach.jpg"
      <repositoryItem>
        ..binary encoding of photo repository item
      </repositoryItem>
    </rim:RegistryObject>
  </rim:RegistryObjectList>
</rim:RegistryObject>
```

394

395 The following example shows the equivalent syntax for representing the membership relationship between a Re-
396 gistryPackage and its members. This representation uses “HasMember” AssociationType instances to establish the
397 membership relationship.

398

```
<rim:RegistryObject xsi:type="rim:RegistryPackageType"
```

```

id="urn:acme:RegistryPackage:photos" .../>

<rim:RegistryObject xsi:type="rim:ExtrinsicObjectType" mimeType="image/jpeg"
id="urn:acme:RegistryPackage:photos:summer-2008:wellfleet-beach.jpg"
  <repositoryItem>
    ...binary encoding of photo repository item
  </repositoryItem>
</rim:RegistryObject>

<Association
  sourceObject="urn:acme:RegistryPackage:photos"
  targetObject="urn:acme:RegistryPackage:photos:summer-2008:wellfleet-
beach.jpg"
  type="urn:oasis:names:tc:ebxml-regrep:AssociationType:HasMember"/>

```

399

400 2.14.3 Description

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

- 401 ● Element RegistryObjectList – This element allows clients to specify members of the RegistryPackage in-
402 stance using a simpler alternative to “HasMember” AssociationType instances.
- 403 ○ A server MUST replace the RegistryObjectList with AssociationType instances such that each Re-
404 gistryObjectType instance is replaced with an AssociationType instance with type
405 “urn:oasis:names:tc:ebxml-regrep:AssociationType:HasMember”, with sourceObject specifying the id
406 of the RegistryPackage instance and with targetObject specifying the id of the RegistryObjectType in-
407 stance

408 2.15 ExternalIdentifierType

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

410 This type allows any number of additional identifiers to be specified for a RegistryObjectType instance. The identi-
411 fier value is defined using the *value* attribute within the context of a ClassificationScheme referenced via the *identi-*
412 *ficationScheme* attribute.

413 2.15.1 Syntax

```

<complexType name="ExternalIdentifierType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <attribute name="registryObject"
        type="tns:objectReferenceType" use="optional"/>
      <attribute name="identificationScheme"
        type="tns:objectReferenceType" use="required"/>
      <attribute name="value" type="tns:LongText" use="required"/>
    </extension>
  </complexContent>
</complexType>

```

414 2.15.2 Example

415 The following examples shows an Organization instance with its tax payer id specified using an ExternalIdentifier-
416 Type instance.

```

<rim:RegistryObject xsi:type="rim:OrganizationType" ...>
  ...
  <rim:ExternalIdentifier ...
    identificationScheme="urn:acme:ClassificationScheme:TaxPayerId"
    value="1234567890"/>
  </rim:ExternalIdentifier>
  ...
</rim:RegistryObject>

```

417 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

418

- 419 ● Attribute identificationScheme - Each ExternalIdentifier instance MUST have an identificationScheme at-
420 tribute that references a ClassificationScheme. This ClassificationScheme defines the namespace within
421 which an identifier is defined using the value attribute for the RegistryObjectType instance referenced by
422 the RegistryObject attribute.
- 423 ● Attribute registryObject - Each ExternalIdentifier instance MAY have a *registryObject* attribute specified.
424 This attribute references the parent RegistryObjectType instance for which this is an ExternalIdentifier.
 - 425 ○ This attribute MUST be specified when a client submits an ExternalIdentifier separately from its parent
426 RegistryObjectType instance
 - 427 ○ This attribute MAY be unspecified when a client submits an ExternalIdentifier as a sub-element of its
428 parent RegistryObjectType instance. In such cases the server MUST set this attributes value to the
429 value of the id attribute of the parent RegistryObjectType instance.
 - 430 ○ Attribute value - Each ExternalIdentifier instance MUST have a *value* attribute that provides the identi-
431 fier value for this ExternalIdentifier (e.g., the tax payer id in example above).

432 2.16 ExternalLinkType

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

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

435 2.16.1 Syntax

```

<complexType name="ExternalLinkType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="ExternalRef"
          type="tns:SimpleLinkType" minOccurs="1" maxOccurs="1"/>
      </sequence>
      <attribute name="registryObject"
        type="tns:objectReferenceType" use="optional"/>
    </extension>
  </complexContent>
</complexType>

```

436 2.16.2 Example

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

```
<rim:RegistryObject xsi:type="rim:OrganizationType" ...>
  ...
  <rim:ExternalLink ...
    objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:XML:WSDL"
    mimeType="text/xml"/>
    <ExternalRef xlink:href="http://www.acme.com"/>
  </rim:ExternalLink>
  ...
</rim:RegistryObject>
```

439 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

440

- 441 ● Element ExternalRef - Each ExternalLink instance MUST have an ExternalRef sub-element defined. This
442 element provides a URI to the external resource pointed to by this ExternalLink instance.
- 443 ● Attribute registryObject – references the parent RegistryObjectType instance within which the ExternalLink-
444 Type instance is composed. The value MUST be provided by client when an ExternalLink is submitted sep-
445 arate from its parent object. The value MUST be set by the server if the ExternalLink is submitted as part of
446 the submission of its parent object.

447 3 Association Information Model

448 A RegistryObjectType instance MAY be associated or related with zero or more RegistryObjectType instances. The
449 information model defines the AssociationType type, an instance of which MAY be used to associate any two Re-
450 gistryObjectType instances. It also defines an Association element for that type.

451 In the example below, an AssociationType instance with type "...Supersedes" is used to indicate that the NAICS2001
452 ClassificationScheme supersedes the NAICS1997 ClassificationScheme.

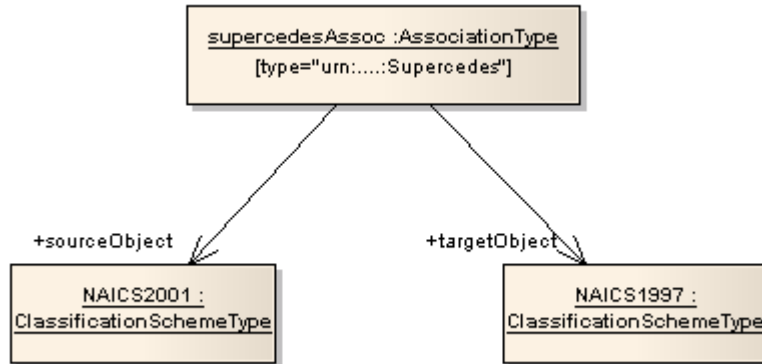


Illustration 3: Association Example

454 3.1 Source and Target Objects

455 An AssociationType instance represents an association between a source RegistryObjectType instance and a target
456 RegistryObjectType instance. These are referred to as *sourceObject* and *targetObject* for the AssociationType in-
457 stance. It is important which object is the sourceObject and which is the targetObject as it determines the directional
458 semantics of an Association.

459 3.2 Type of an Association

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

462 3.3 AssociationType

463 **Base Type:** RegistryObjectType

464 3.3.1 Syntax

```
<complexType name="AssociationType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <attribute name="type"
        type="tns:objectReferenceType" use="required"/>
      <attribute name="sourceObject"
        type="tns:objectReferenceType" use="required"/>
      <attribute name="targetObject"
        type="tns:objectReferenceType" use="required"/>
    </extension>
  </complexContent>
</complexType>
```


465 3.3.2 Example

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

```
<rim:RegistryObject xsi:type="rim:OrganizationType"
  id="urn:acme:Organization:acme-inc" ... />
<rim:RegistryObject xsi:type="rim:ServiceType"
  id="urn:acme:Service:stock-quote" ... />
<rim:RegistryObject xsi:type="rim:AssociationType"
  id="urn:acme:Association:acme-example-relationship"
  sourceObject="urn:acme:Organization:acme-inc"
  targetObject="urn:acme:Service:stock-quote"
  type="urn:oasis:names:tc:ebxml-regrep:AssociationType:OffersService" ... />
```

468 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

469

- 470 ● Attribute sourceObject - Each Association MUST have a *sourceObject* attribute that references the Re-
471 gistryObjectType instance that is the source of that Association.
- 472 ● Attribute targetObject - Each Association MUST have a *targetObject* attribute that references the Registry-
473 Objectype instance that is the target of that Association.
- 474 ● Attribute type - Each Association MUST have a *type* attribute that identifies the type of that association.
 - 475 ○ The value of the type attribute MUST be a reference to a ClassificationNode within the canonical As-
476 sociationType ClassificationScheme.
 - 477 ○ A server MUST support the canonical association types as defined by the canonical AssociationType
478 ClassificationScheme. Deployments and profiles may extend the canonical AssociationType Classifica-
479 tionScheme by adding additional ClassificationNodes to it.

480 3.4 Access Control

481 A client MAY create an AssociationType instance between *any* two RegistryObjectType instances assuming the ac-
482 cess control policies associated with the source and target object permit the client to create a reference to them. The
483 default access control policy permits any client to create a reference to an object.

484 4 Classification Information Model

485 The ebRIM information model supports classification of RegistryObjectType instances using values defined by a
486 taxonomy or controlled vocabulary. A taxonomy is represented in ebRIM by the ClassificationSchemeType type.
487 Values in a taxonomy are represented by the ClassificationNode type. A classification instance is represented in
488 ebRIM by the ClassificationType type.

489 This specification specifies a set of canonical ClassificationSchemes. Deployments and profiles MAY extend these
490 canonical ClassificationSchemes by adding additional ClassificationNodes to them. They MAY also define new
491 ClassificationSchemes. A RegistryObjectType instance MAY be classified using *any* ClassificationNode in *any* Clas-
492 sificationScheme supported by the server. A RegistryObjectType instance MAY have any number of classifications
493 defined for it.

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

496 Illustration 4 below shows RegistryObjectType instances representing Organizations as grey boxes. Each Organiza-
497 tion represents an automobile manufacturer. Organization is classified by the ClassificationNode named “Automot-
498 iva” under the ClassificationScheme instance with name “IndustryScheme”. Furthermore, the US Automobile manu-
499 facturers are classified by the “US” ClassificationNode under the ClassificationScheme with name “GeographyS-
500 cheme”. Similarly, a European automobile manufacturer is classified by the “Europe” ClassificationNode under the
501 ClassificationScheme with name “GeographyScheme”.

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

504

505

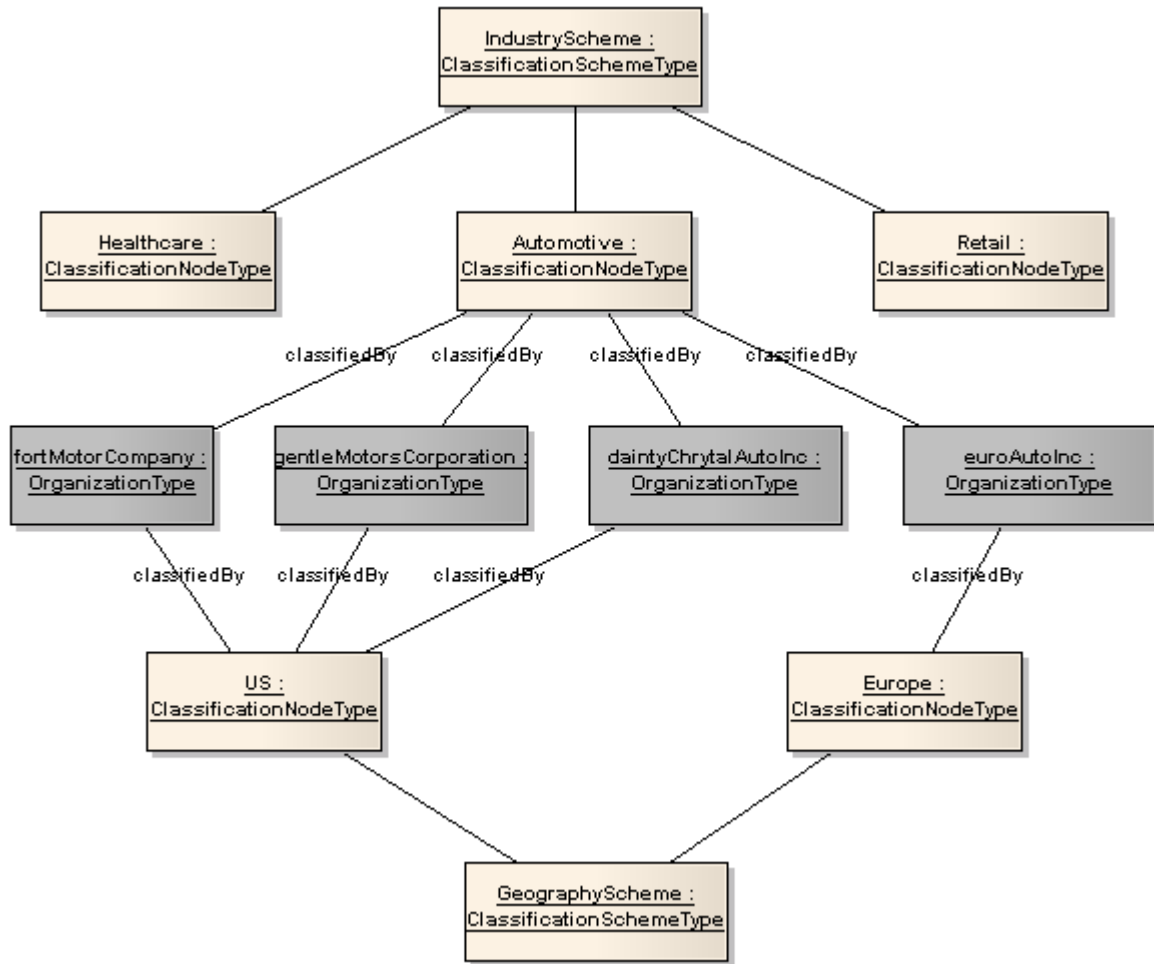


Illustration 4: Classification Example

507 Illustration 5 shows the Classification information model.

508

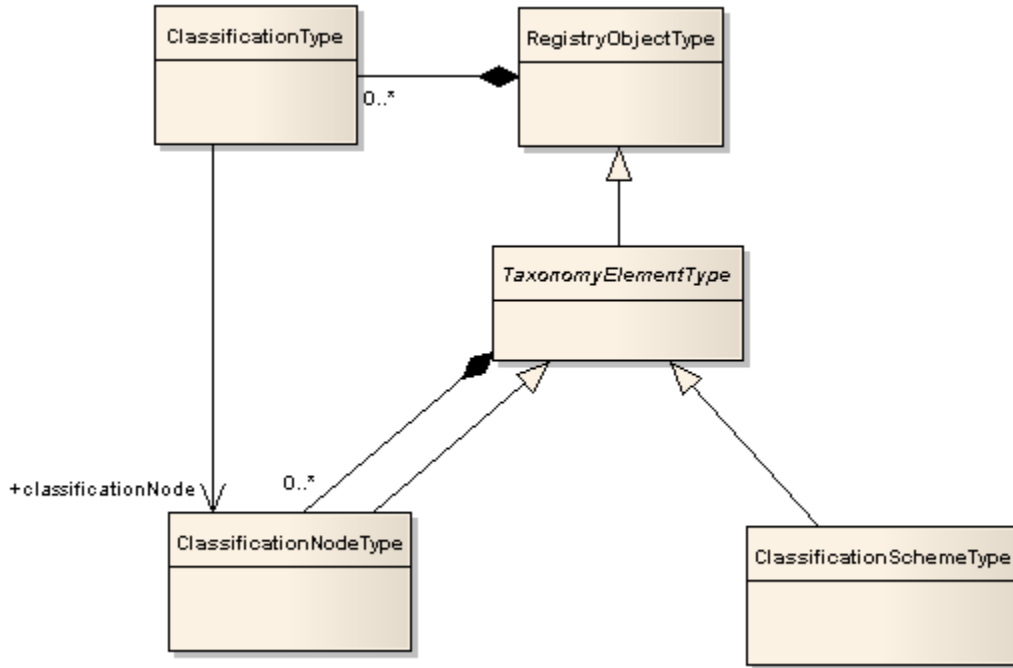


Illustration 5: Classification Information Model

510 4.1 TaxonomyElementType

511 **Base Type:** RegistryObjectType

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

513 4.1.1 Syntax

```

<complexType name="TaxonomyElementType" abstract="true">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="ClassificationNode" type="tns:ClassificationNodeType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

```

514 4.1.2 Description

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

515

- 516 ● Element ClassificationNode – This element represents a ClassificationNode child of a parent Taxonomy-
517 ElementType instance. A TaxonomyElementType instance MAY have any number of ClassificationNode
518 child elements.

519 4.2 ClassificationSchemeType

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

521 A ClassificationScheme instance represents a taxonomy.

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

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

527 4.2.1 Syntax

```
<complexType name="ClassificationSchemeType">
  <complexContent>
    <extension base="tns:TaxonomyElementType">
      <attribute name="isInternal" type="boolean" use="required"/>
      <attribute name="nodeType"
        type="tns:objectReferenceType" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

528 4.2.2 Example

529 The following examples shows a ClassificationScheme representing gender values.

```
<rim:RegistryObject xsi:type="rim:ClassificationSchemeType"
  id="urn:acme:GenderScheme" isInternal="true"
  nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode" ...>
  <Name>
    <LocalizedString value="GenderScheme"/>
  </Name>
  <rim:ClassificationNode id="urn:acme:Gender:Male" code="Male" .../>
  <rim:ClassificationNode id="urn:acme:Gender:Female" code="Female" .../>
  <rim:ClassificationNode id="urn:acme:Gender:Other" code="Other" .../>
</rim:RegistryObject>
```

530 4.2.3 Description

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

531

532 ● Attribute isInternal - When submitting a ClassificationSchemeType instance the client MUST declare
533 whether the ClassificationSchemeType instance represents an internal or an external taxonomy. This allows
534 the server to validate the subsequent submissions of ClassificationNodeType and ClassificationType in-
535 stances in order to maintain the type of ClassificationScheme consistent throughout its lifecycle.

536 ● Attribute nodeType - When submitting a ClassificationScheme instance the client MUST declare the struc-
537 ture of taxonomy nodes within the ClassificationScheme via the nodeType attribute. The value of the node-
538 Type attribute MUST be a reference to a ClassificationNodeType instance within the canonical NodeType
539 ClassificationScheme. A server MUST support the node types as defined by the canonical NodeType Clas-
540 sificationScheme. The canonical NodeType ClassificationScheme MAY easily be extended by adding addi-
541 tional ClassificationNodes to it.
542

543 The following table lists the canonical ClassificationNode defined as values for the NodeType Classifica-
 544 tionScheme:
 545

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.

546

547 4.3 ClassificationNodeType

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

549 ClassificationNodeType instances are used to define values for a taxonomy represented by ClassificationS-
 550 chemeType instance.

551 4.3.1 Syntax

```
<complexType name="ClassificationNodeType">
  <complexContent>
    <extension base="tns:TaxonomyElementType">
      <attribute name="parent" type="tns:objectReferenceType" use="optional"/>
      <attribute name="path" type="string" use="optional"/>
      <attribute name="code" type="tns:LongText" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

552 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

553

554 ● Attribute code - A ClassificationNodeType instance MUST have a *code* attribute. The code attribute con-
 555 tains a code that represents a value within a ClassificationScheme.

556 ○ The code attribute of a ClassificationNodeType instance MUST be unique with respect to all sibling
 557 ClassificationNodes that are immediate children of the same parent TaxonomyElementType instance.

- 558 ● Attribute parent - A ClassificationNodeType instance MAY have a *parent* attribute. The parent attribute
559 references the parent TaxonomyElementType instance. This is either another ClassificationNode-
560 Type instance or the ClassificationSchemeType instance.
- 561 ● Attribute path - A ClassificationNodeType instance MAY have a *path* attribute. The path attribute repres-
562 ents a hierarchical path from the root ClassificationSchemeType to the ClassificationNodeType instance.
563 The [syntax of the path attribute value](#) is defined in 4.3.3.
- 564 ○ A server MUST set the path attribute for any ClassificationNodeType instance when it is submitted by
565 a client.
- 566 ○ The path attribute MUST be ignored by the server if it is specified by the client during the submission
567 of the ClassificationNodeType instance.
- 568 ○ The path attribute of a ClassificationNode MUST be unique within a server.

569 4.3.3 Canonical Path Syntax

570 The path attribute of the ClassificationNodeType instance contains an absolute path in a canonical representation
571 that uniquely identifies the path leading from the root ClassificationSchemeType instance to that ClassificationNo-
572 deType instance.

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

574

```
canonicalPath ::= '/' rootTaxonomyElementId nodePath
nodePath      ::= '/' nodeCode
               | '/' nodeCode ( nodePath )?
```

575 In the above grammar, rootTaxonomyElementId is the id attribute of the root ClassificationSchemeType or Classific-
576 ationNodeType instance, and nodeCode is defined by NCName production as defined by
577 <http://www.w3.org/TR/REC-xml-names/#NT-NCName>.

578 4.3.3.1 Example of Canonical Path Representation

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

581

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

582 4.4 ClassificationType

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

584 A ClassificationType instance classifies a RegistryObjectType instance by using a value defined within a particular
585 ClassificationScheme. An internal Classification specifies the value by referencing the ClassificationNodeType in-
586 stance within a ClassificationSchemeType instance. An external Classification specifies the value using a string
587 value that is defined in some external specification represented by an external ClassificationSchemeType instance.

588 4.4.1 Syntax

```
<complexType name="ClassificationType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <attribute name="classificationScheme"
        type="tns:objectReferenceType" use="optional"/>
      <attribute name="classifiedObject" />
    </extension>
  </complexContent>
</complexType>
```

```

    type="tns:objectReferenceType" use="optional"/>
  <attribute name="classificationNode"
    type="tns:objectReferenceType" use="optional"/>
  <attribute name="nodeRepresentation"
    type="tns:LongText" use="optional"/>
</extension>
</complexContent>
</complexType>

```

589 4.4.2 Example

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

592

```

<rim:RegistryObject xsi:type="rim:PersonType"
  id="urn:acme:person:Danyal" ...>
  ...
  <Classification classifiedObject="urn:acme:person:Danyal"
    classificationNode="urn:acme:Gender:Male"
    ...
  </rim:RegistryObject>

```

593 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

594

- 595 ● Attribute classificationNode - If the ClassificationType instance represents an internal classification, then
596 the *classificationNode* attribute is required.
 - 597 ○ The *classificationNode* value MUST reference a ClassificationNodeType instance.
- 598 ● Attribute classifiedObject - For both internal and external classifications, the *classifiedObject* attribute is
599 required and it references the RegistryObjectType instance that is classified by this Classification.
- 600 ● Attribute classificationScheme - If the ClassificationType instance represents an external classification,
601 then the *classificationScheme* attribute is required.
 - 602 ○ The *classificationScheme* value MUST reference a ClassificationScheme instance.
- 603 ● Attribute nodeRepresentation - If the ClassificationType instance represents an external classification, then
604 the *nodeRepresentation* attribute is required. It is a representation of a taxonomy value from a classification
605 scheme.
- 606 ● A canonical slot with name “urn:oasis:names:tc:ebxml-regrep:rim:Classification:context” may be option-
607 ally specified to provide additional context for a ClassificationType instance

608 5 Provenance Information Model

609 The term [provenance](#) in the English language implies the origin and history of ownership and custodianship of
610 things of value. When applied to the ebXML RegRep, provenance implies information about the origin, history of
611 ownership, custodianship, and other relationships between entities such as people, organizations and information
612 represented by RegistryObjectType instances.

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

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

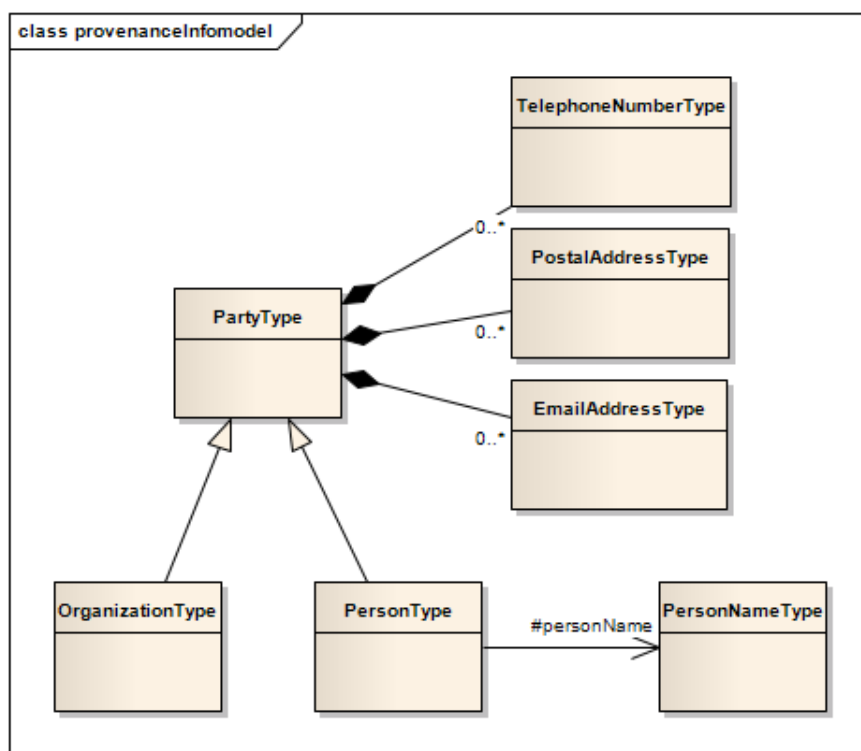


Illustration 6: Provenance Information Model

618 5.1 PostalAddressType

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

620 This type represents a postal or mailing address.

621 5.1.1 Syntax

```
<complexType name="PostalAddressType">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <attribute name="city" type="tns:ShortText" use="optional"/>
      <attribute name="country" type="tns:ShortText" use="optional"/>
      <attribute name="postalCode" type="tns:ShortText" use="optional"/>
      <attribute name="stateOrProvince" type="tns:ShortText" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

```

    <attribute name="street" type="tns:ShortText" use="optional"/>
    <attribute name="streetNumber" type="tns:String32" use="optional"/>
    <attribute name="type" type="tns:objectReferenceType" use="optional"/>
  </extension>
</complexContent>
</complexType>

```

622 5.1.2 Example

```

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

```

623 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

624

- 625 ● Attribute *city* - A *PostalAddressType* instance MAY have a *city* attribute identifying the city for that ad-
626 dress.
- 627 ● Attribute *country* - A *PostalAddressType* instance MAY have a *country* attribute identifying the country for
628 that address.
- 629 ● Attribute *postalCode* - A *PostalAddressType* instance MAY have a *postalCode* attribute identifying the
630 postal code (e.g., zip code) for that address.
- 631 ● Attribute *stateOrProvince* - A *PostalAddressType* instance MAY have a *stateOrProvince* attribute identify-
632 ing the state, province or region for that address.
- 633 ● Attribute *street* - A *PostalAddressType* instance MAY have a *street* attribute identifying the street name for
634 that address.
- 635 ● Attribute *streetNumber* - A *PostalAddressType* instance MAY have a *streetNumber* attribute identifying the
636 street number (e.g., 65) for the street address.

637 5.2 TelephoneNumberType

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

639 This type defines attributes of a telephone number.

640 5.2.1 Syntax

```

<complexType name="TelephoneNumberType">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">

```

```

<attribute name="areaCode" type="tns:String8" use="optional"/>
<attribute name="countryCode" type="tns:String8" use="optional"/>
<attribute name="extension" type="tns:String8" use="optional"/>
<attribute name="number" type="tns:String16" use="optional"/>
<attribute name="type" type="tns:objectReferenceType" use="optional"/>
</extension>
</complexContent>
</complexType>

```

641 5.2.2 Example

```

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

```

642 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

643

- 644 ● Attribute *areaCode* - A *TelephoneNumberType* instance MAY have an *areaCode* attribute that provides the
645 area code for that telephone number.
- 646 ● Attribute *countryCode* - A *TelephoneNumberType* instance MAY have a *countryCode* attribute that
647 provides the country code for that telephone number.
- 648 ● Attribute *extension* - A *TelephoneNumberType* instance MAY have an *extension* attribute that provides the
649 extension number, if any, for that telephone number.
- 650 ● Attribute *number* - A *TelephoneNumberType* instance MAY have a *number* attribute that provides the local
651 number (without area code, country code and extension) for that telephone number.
- 652 ● Attribute *type* - A *TelephoneNumberType* instance MAY have a *type* attribute that provides the type for the
653 *TelephoneNumber*. The value of the *phoneType* attribute MUST be a reference to a *ClassificationNode* in
654 the canonical *PhoneType ClassificationScheme*.

655 5.3 EmailAddressType

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

657 This type defines attributes of an email address.

658 5.3.1 Syntax

```

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

```

```

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

```

659 5.3.2 Example

```

<rim:RegistryObject xsi:type="rim:PersonType" id="urn:acme:person:Danyal" ...>
  ...
  <rim:EmailAddress address="danyal@play.com"
    type="urn:oasis:names:tc:ebxml-regrep:EmailType:HomeEmail"/>
  ...
</rim:RegistryObject>

```

660 5.3.3 Description

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

661

- 662 ● Attribute address - An EmailAddressType instance MUST have an *address* attribute that provides the actual
- 663 email address.
- 664 ● Attribute type - An EmailAddressType instance MAY have a *type* attribute that provides the type for that
- 665 email address. The value of the type attribute MUST be a reference to a ClassificationNode in the canonical
- 666 EmailType ClassificationScheme.

667 5.4 PartyType

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

669 This abstract type represents a party that has contact information such as PostalAddress, EmailAddress, Tele-
 670 phoneNumber etc. It is used as a common base type for PersonType and OrganizationType.

671 5.4.1 Syntax

```

<complexType name="PartyType" abstract="true">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="PostalAddress" type="tns:PostalAddressType"
          minOccurs="0" maxOccurs="unbounded"/>
        <element name="TelephoneNumber" type="tns:TelephoneNumberType"
          minOccurs="0" maxOccurs="unbounded"/>
        <element name="EmailAddress" type="tns:EmailAddressType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

```

672 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

673

674 ● Element EmailAddress - A **PartyType** instance MAY have any number of EmailAddress sub-elements.
675 Each EmailAddress provides an email address for that PartyType instance. A **PartyType** instance
676 SHOULD have at least one EmailAddress.

677 ● Element PostalAddress - A **PartyType** instance MAY have any number of PostalAddress sub-elements.
678 Each PostalAddress element provides a postal address for that PartyType instance. A **PartyType** in-
679 stance SHOULD have at least one PostalAddress.

680 ● Element TelephoneNumber - A **PartyType** instance MAY have any number of *TelephoneNumber* sub-ele-
681 ments. Each TelephoneNumber element provides a TelephoneNumber for that **PartyType** instance. A
682 **PartyType** instance SHOULD have at least one TelephoneNumber.

683 5.5 PersonType

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

685 This type represent a person.

686 5.5.1 Syntax

```
<complexType name="PersonType">
  <complexContent>
    <extension base="tns:PartyType">
      <sequence>
        <element name="PersonName" type="tns:PersonNameType"
          minOccurs="0" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

687 5.5.2 Example

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

688 5.5.3 Description

Node	Type	Cardinality	Default Value	Specified By	Mutable
PersonName	PersonNameType	0..1		Client	No

689

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

692 5.6 PersonNameType

693 Base Type: [ExtensibleObjectType](#)

694 This represents the name for a PersonType instance.

695 5.6.1 Syntax

```
<complexType name="PersonNameType">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <attribute name="firstName" type="tns:ShortText" use="optional"/>
      <attribute name="middleName" type="tns:ShortText" use="optional"/>
      <attribute name="lastName" type="tns:ShortText" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

696 5.6.2 Example

```
<rim:RegistryObject xsi:type="rim:PersonType" id="urn:acme:person:Danyal" ...>
  ...
  <rim:PersonName firstName="Danyal" middleName="Idris" lastName="Najmi"/>
  ...
</rim:RegistryObject>
```

697 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

698

- 699 ● Attribute `firstName` - A `PersonName` instance SHOULD have a *firstName* attribute that is the given name
700 of the person.
- 701 ● Attribute `lastName` - A `PersonName` instance SHOULD have a *lastName* attribute that is the family name
702 of the person.
- 703 ● Attribute `middleName` - A `PersonName` instance SHOULD have a *middleName* attribute that is the middle
704 name of the person.

705 5.7 OrganizationType

706 Base Type: [PartyType](#)

707 This type represents an organization or entity.

708 5.7.1 Syntax

```
<complexType name="OrganizationType">
  <complexContent>
    <extension base="tns:PartyType">
      <sequence>
        <element name="Organization" type="tns:OrganizationType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="primaryContact" type="tns:objectReferenceType"

```

```

        use="optional"/>
    </extension>
</complexContent>
</complexType>

```

709 5.7.2 Example

```

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

```

710 5.7.3 Description

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

711

- 712 ● Element Organization – This element allows clients to specify sub-organizations of the Organization in-
713 stance using a simpler alternative to specifying “HasMember” AssociationType instances between the par-
714 ent and child Organizations.
 - 715 ○ A server MUST replace any nested Organization elements within an OrganizationType instance with
716 AssociationType instances such that each nested Organization element is replaced with an Association-
717 Type instance with type “urn:oasis:names:tc:ebxml-regrep:AssociationType:HasMember”, with
718 sourceObject specifying the id of the parent OrganizationType instance and with targetObject specify-
719 ing the id of the nested Organization element
- 720 ● Attribute primaryContact - An OrganizationType instance SHOULD have a *primaryContact* attribute that
721 references the Person instance for the person that is the primary contact for that organization.

722 5.8 Associating Organization With Persons

723 There are many situation where a person is related to an organization. Such relationship SHOULD be defined by As-
724 sociationType instances between an OrganizationType instance and a PersonType instance as follows:

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

729 5.9 Associating Organization With Organizations

730 There are many situation where an organization is related to another organization. Such relationship SHOULD be
731 defined by AssociationType instances between an OrganizationType instance and another OrganizationType in-
732 stance.

- 733 ● To represent parent-child relationship between organizations the type attribute of the Association SHOULD
734 reference the canonical ClassificationNode with id “urn:oasis:names:tc:ebxml-regrep:AssociationType:Has-
735 Member” or one of its descendants.
- 736 ● The sourceObject SHOULD reference the parent OrganizationType instance.
- 737 ● The targetObject SHOULD reference the child OrganizationType instance.

738 **5.10 Associating Organizations With RegistryObjects**

739 An organization MAY be associated with zero or more RegistryObjectType instances. Each such association is
740 modeled in ebRIM using an Association instance between an Organization instance and a RegistryObjectType in-
741 stance.

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

745 6 Service Information Model

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

749

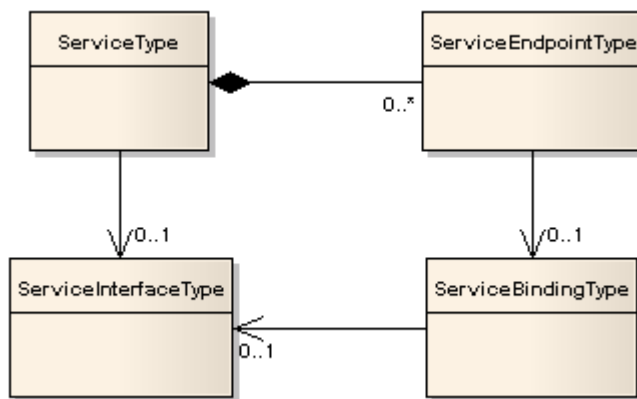


Illustration 7: Service Information Model

751 6.1 ServiceType

752 **Base Type:** RegistryObjectType

753 This type represent a logical service. Physical service endpoints are represented by the ServiceEndpointType type.
754 A ServiceType instance typically contains ServiceEndpoint sub-elements where each ServiceEndpoint sub-element
755 represents an alternate endpoint for a service.

756 6.1.1 Syntax

```
<complexType name="ServiceType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="ServiceEndpoint" type="tns:ServiceEndpointType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="serviceInterface"
        type="tns:objectReferenceType" use="optional" />
    </extension>
  </complexContent>
</complexType>
```

757 6.1.2 Example

```
<rim:RegistryObject xsi:type="rim:ServiceType"
  id="urn:acme:Service:StockQuoteService" ...>
  ...
  <rim:ServiceEndpoint
    id="urn:acme:ServiceEndpoint:StockQuoteService:free" .../>
  <rim:ServiceEndpoint
    id="urn:acme:ServiceEndpoint:StockQuoteService:premium" .../>
```

```
</rim:RegistryObject>
```

758 6.1.3 Description

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

759

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

764 6.2 ServiceEndpointType

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

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

767 6.2.1 Syntax

```
<complexType name="ServiceEndpointType">  
  <complexContent>  
    <extension base="tns:RegistryObjectType">  
      <attribute name="address" type="anyURI" use="optional" />  
      <attribute name="serviceBinding"  
        type="tns:objectReferenceType" use="optional" />  
    </extension>  
  </complexContent>  
</complexType>
```

768 6.2.2 Example

```
<rim:RegistryObject xsi:type="rim:ServiceType"  
  id="urn:acme:Service:StockQuoteService" ...>  
  ...  
  <rim:ServiceEndpoint id="urn:acme:ServiceEndpoint:StockQuoteService:free"  
    address="http://acme.com/StockQuoteService/free"  
    serviceBinding="urn:acme:ServiceBinding:soap:StockQuoteService">  
</rim:RegistryObject>
```

769 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

770

- 771 ● Attribute address – Represents the endpoint address URI that a client of the service endpoint may use to ac-
772 cess the service endpoint

- 773 ● Attribute serviceBinding – References the [ServiceBindingType](#) instance that represents protocol-specific
 774 binding information for the ServiceEndpointType instance
- 775 ○ MUST reference a ServiceBindingType instance

776 6.3 ServiceBindingType

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

778 This type represents protocol-specific binding information for a ServiceEndpointType instance. Example of a proto-
 779 col-specific binding is a SOAP binding.

780 6.3.1 Syntax

```
<complexType name="ServiceBindingType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <attribute name="serviceInterface"
        type="tns:objectReferenceType" use="optional" />
    </extension>
  </complexContent>
</complexType>
```

781 6.3.2 Example

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

782 6.3.3 Description

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

- 783
- 784 ● Attribute serviceInterface – References a ServiceInterfaceType instance which represents the abstract ser-
 785 vice interface for the service
- 786 ○ MUST reference a ServiceInterfaceType instance if specified

787 6.4 ServiceInterfaceType

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

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

790 6.4.1 Syntax

```
<complexType name="ServiceInterfaceType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
    </extension>
  </complexContent>
</complexType>
```

791 6.4.2 Example

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

792 6.4.3 Description

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

794 7 Query Information Model

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

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

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

803 A QueryDefinitionType instance MAY be invoked using a QueryType instance. The ebRS Query protocol allows cli-
804 ents to invoke a QueryDefinitionType instance using a QueryType instance within the Query protocol.

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

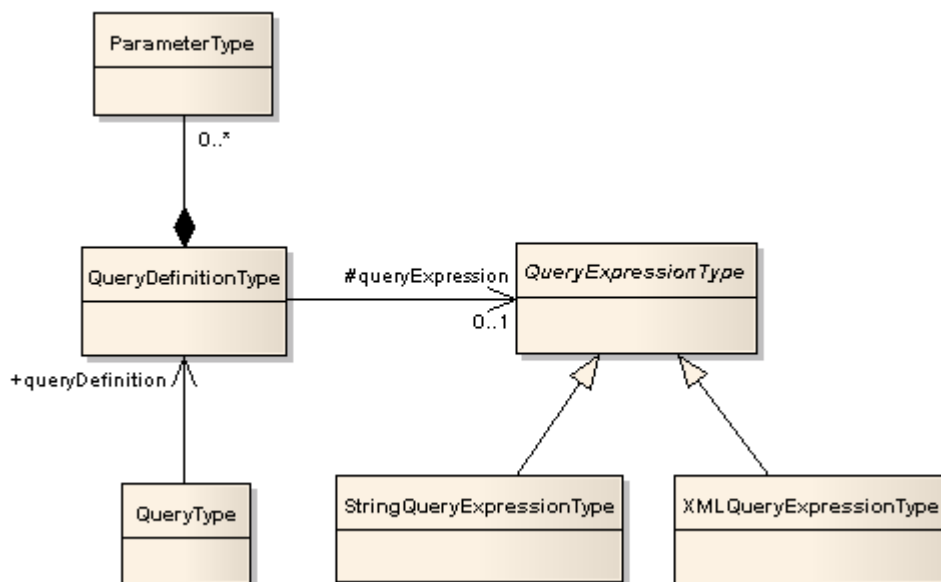


Illustration 8: Query Information Model

807 7.1 QueryDefinitionType

808 **Base Type:** RegistryObjectType

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

811 7.1.1 Syntax

```
<complexType name="QueryDefinitionType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="Parameter"
          type="tns:ParameterType" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

```

    <element name="QueryExpression"
      type="tns:QueryExpressionType" minOccurs="0" maxOccurs="1"/>
  </sequence>
</extension>
</complexContent>
</complexType>

```

812 7.1.2 Example

```

<rim:RegistryObject xsi:type="rim:QueryDefinitionType"
  id="urn:oasis:names:tc:ebxml-regrep:query:GetObjectById">
  <rim:Parameter parameterName="id"
    minOccurs="1" maxOccurs="1" defaultValue="%">
  </Parameter>
  <rim:QueryExpression xsi:type="rim:StringQueryExpressionType"
    queryLanguage="urn:oasis:names:tc:ebxml-regrep:QueryLanguage:EJBQL">
    <Value>
      SELECT Object(ro) FROM ...
    </Value>
  </QueryExpression>
</rim:RegistryObject>

```

813 7.1.3 Description

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

814

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

819 7.2 ParameterType

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

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

822 7.2.1 Syntax

```

<complexType name="ParameterType">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <sequence>
        <element name="Name" type="tns:InternationalStringType"
          minOccurs="1" maxOccurs="1"/>
        <element name="Description" type="tns:InternationalStringType"
          minOccurs="0" maxOccurs="1"/>
      </sequence>
      <attribute name="parameterName" type="string" use="required"/>
      <attribute name="dataType" type="string" use="required" />
      <attribute name="defaultValue" type="string" use="optional"/>
      <attribute name="minOccurs" type="nonNegativeInteger" default="1"/>
      <attribute name="maxOccurs" type="nonNegativeInteger" default="1"/>
    </extension>
  </complexContent>
</complexType>

```

```

</extension>
</complexContent>
</complexType>

```

823 7.2.2 Example

```

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

```

824 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

825

826 ● Attribute dataType – Specifies the data type for the parameter.

827 ○ The dataType MUST be “string” for parameters whose values are represented by a string value.

828 ○ The dataType MUST be “boolean” for parameters whose values are represented by a boolean value.

829 ○ The dataType MUST be “taxonomyElement” for parameters whose value is the id of a TaxonomyElement.
830

831 ● Attribute defaultValue - Specifies the default value for the parameter. This value MUST be used as parameter value when the query is invoked if the client does not specify a value for this parameter.
832

833 ● Element Description - Specifies a human-friendly description of the parameter that indicates what the parameter value represents and what kind of value is allowed. The description MAY be provided in multiple local languages and character sets.
834
835

836 ● Attribute minOccurs – Specifies the minimum number of values allowed for the parameter.

837 ● Attribute maxOccurs - Specifies the maximum number of values allowed for the parameter.

838 ● Element Name - Specifies a human-friendly name for the parameter. The name MAY be provided in multiple local languages and character sets.
839

840 ● Attribute parameterName – Specifies the canonical name of the parameter. The canonicalName identifies the parameter in a locale-insensitive manner
841

842 ○ SHOULD match a declared parameter name within the query expression for the QueryDefinitionType
843 instance

- 844 ○ The parameterName MUST be unique across the universe of all sibling ParameterType instances
845 within a QueryDefinitionType instance

846 **7.3 QueryExpressionType**

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

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

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

- 851 ● StringQueryExpressionType – This type MAY be used to represent non-XML query syntaxes such as SQL-
852 92 and EJBQL.
- 853 ● XMLQueryExpressionType - This type MAY be used to represent XML query syntaxes such as OGC Filter
854 Query.

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

856 **7.3.1 Syntax**

```
<complexType name="QueryExpressionType" abstract="true">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <attribute name="queryLanguage"
        type="tns:objectReferenceType" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

857 **7.3.2 Description**

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

858

- 859 ● Attribute queryLanguage – Specifies the query language used by the QueryExpressionType instance.
- 860 ○ MUST be a reference to a ClassificationNode in the canonical Query Language ClassificationScheme
861 whose id is “urn:oasis:names:tc:ebxml-regrep:classificationScheme:QueryLanguage”.

862 **7.4 StringQueryExpressionType**

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

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

865 **7.4.1 Syntax**

```
<complexType name="StringQueryExpressionType">
  <complexContent>
    <extension base="tns:QueryExpressionType">
      <sequence>
        <element name="Value" type="string" minOccurs="1" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```



```
</complexType>
```

866 7.4.2 Example

```
<rim:RegistryObject xsi:type="rim:QueryDefinitionType"
  id="urn:oasis:names:tc:ebxml-regrep:query:GetObjectById">
  <rim:Parameter ... />
  ...
  <rim:QueryExpression xsi:type="rim:StringQueryExpressionType"
    queryLanguage="urn:oasis:names:tc:ebxml-regrep:QueryLanguage:EJBQL">
    <Value>
      SELECT Object(ro) FROM RegistryObjectType WHERE ...
    </Value>
  </rim:QueryExpression>
</rim:RegistryObject>
```

867 7.4.3 Description

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

868

- 869 ● Element Value – Specifies the string value representing the actual query expression within the query lan-
870 guage specified by the queryLanguage attribute inherited from base type QueryExpressionType.

871 7.5 XMLQueryExpressionType

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

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

874 7.5.1 Syntax

```
<complexType name="XMLQueryExpressionType">
  <complexContent>
    <extension base="tns:QueryExpressionType">
      <sequence>
        <any namespace="##other"
          processContents="lax" minOccurs="1" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

875 7.5.2 Example

```
<rim:RegistryObject xsi:type="rim:QueryDefinitionType"
  <rim:Parameter ... />
  ...
  <rim:QueryExpression xsi:type="rim:XMLQueryExpressionType"
    queryLanguage="urn:oasis:names:tc:ebxml-regrep:QueryLanguage:EJBQL">
    <ogc:Filter>
      ...
    </ogc:Filter>
  </rim:QueryExpression>
</rim:RegistryObject>
```

876 7.5.3 Description

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

879 7.6 QueryType

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

881 This type represents the invocation of a parameterized query.

882 7.6.1 Syntax

```
<complexType name="QueryType">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <attribute name="queryDefinition"
        type="tns:objectReferenceType" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

883 7.6.2 Example

```
<rim:RegistryObject xsi:type="rim:QueryType"
  queryDefinition="urn:oasis:names:tc:ebxml-regrep:query:GetObjectById">
  <rim:Slot name="id">
    <rim:SlotValue xsi:type="rim:StringValueType">
      <rim:Value>urn:acme:person:Danyal</rim:Value>
    </rim:SlotValue>
  </rim:Slot>
</rim:RegistryObject>
```

884 7.6.3 Description

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

885

- 886 ● Attribute queryDefinition – References the parameterized query to be invoked by the server.
- 887 ○ The value of this attribute MUST be a reference to a QueryDefinitionType instance that is supported by
888 the server.
- 889 ● Element Slot (Inherited) - Each Slot element specifies a parameter value for a parameter supported by the
890 QueryDefinitionType instance.
- 891 ○ The slot name MUST match a parameterName attribute within a Parameter's definition within the
892 QueryDefinitionType instance.
- 893 ○ The slot value's type MUST match the dataType attribute for the Parameter's definition within the
894 QueryDefinitionType instance.
- 895 ○ A server MUST NOT treat the order of parameters as significant.

896 8 Event Information Model

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

- 899 ● **AuditableEventType** – Represents a server event that is typically a consequence of a client request.
- 900 ● **SubscriptionType** – Represents a client's subscription to receive notification of AuditableEventType instances based upon a specified selection criteria.
- 901
- 902 ● **QueryType** – Represents a query invocation that is used to select events of interest within a SubscriptionType instance. This type has been specified previously in the Query Information Model.
- 903
- 904 ● **NotificationType** – Represents a notification sent by the server to a client regarding an event that matches the criteria specified by the client within a SubscriptionType instance.
- 905

906

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

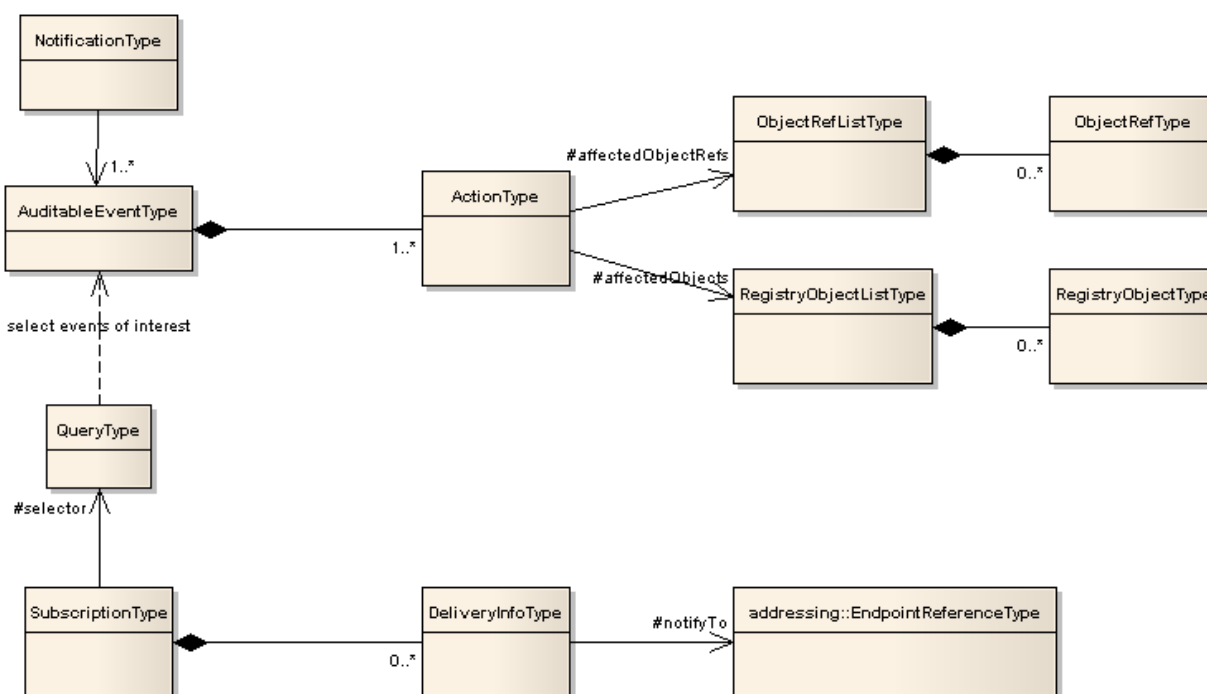


Illustration 9: Event Information Model

912 8.1 AuditableEventType

913 **Base Type:** RegistryObjectType

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

917 AuditableEventType instances represent a change in the state of a RegistryObjectType instance. For example a cli-
 918 ent request could Create, Update, Deprecate or Delete a RegistryObjectType instance. An AuditableEventType in-
 919 stance is created when a request creates or alters the state of a RegistryObjectType instance. Read-only requests typ-
 920 ically do not generate an AuditableEventType instance.

921 8.1.1 Syntax

```
<complexType name="AuditableEventType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="Action" type="tns:ActionType"
          minOccurs="1" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="timestamp" type="dateTime" use="required"/>
      <attribute name="user" type="string" use="required"/>
      <attribute name="requestId"
        type="string" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

922 8.1.2 Example

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

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

925 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

926

- 927 ● Element Action – Represents an action taken by the server within the context of an AuditableEventType in-
 928 stance. An AuditableEventType instance MUST have one or more Action instances.
- 929 ● Attribute requestId – Specifies the id of the request that generated the AuditableEventType instance.
- 930 ● Attribute timestamp – Specifies the timestamp that represents the date and time the event occurred.

- 931 ● Attribute user – Specifies the id of the registered user associated with the client that made the request to the
 932 server that generated the AuditableEventType instance. Note that the inherited attribute owner SHOULD be
 933 set by a server to an internal system user since it is the server and not the user associated with the request
 934 that creates an AuditableEventType instance

935 8.2 ActionType

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

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

938 8.2.1 Syntax

```
<complexType name="ActionType">
  <sequence>
    <element name="AffectedObjects" type="tns:RegistryObjectListType"
      minOccurs="0" maxOccurs="1"/>
    <element name="AffectedObjectRefs" type="tns:ObjectRefListType"
      minOccurs="0" maxOccurs="1"/>
  </sequence>
  <attribute name="eventType" type="tns:objectReferenceType" use="required"/>
</complexType>
```

939 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

940

- 941 ● Element AffectedObject – Identifies the RegistryObjectType instances that were affected by the event. The
 942 AffectedObject element contains any number of elements of type RegistryObjectType, each of which is a
 943 RegistryObjectType instance affected by the event. If this element is present then AffectedObjectRefs ele-
 944 ment MUST NOT be present.
- 945 ● Element AffectedObjectRefs – Identifies the RegistryObjectType instances that were affected by the event.
 946 The AffectedObject element contains any number of ObjectRef elements each of which reference a Re-
 947 gistryObjectType instance that was affected by the event. If this element is present then AffectedObjects
 948 element MUST NOT be present.
- 949 ● Attribute eventType – Specifies the type of event associated with the Action within an AuditableEventType
 950 instance.
- 951 ○ The value of the eventType attribute MUST be a reference to a ClassificationNode in the canonical
 952 EventType ClassificationScheme.
- 953 ○ A Registry MUST support the event types as defined by the EventType ClassificationScheme.
- 954 ○ The canonical EventType ClassificationScheme MAY easily be extended by adding additional Classi-
 955 ficationNodes to it

956 8.3 SubscriptionType

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

958 This type represents a subscription on behalf of a client to receive notifications by the server of events that are of in-
 959 terest to the client.

960 8.3.1 Syntax

```
<complexType name="SubscriptionType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <sequence>
        <element name="DeliveryInfo"
          type="tns:DeliveryInfoType" minOccurs="0" maxOccurs="unbounded" />
        <element name="Selector"
          type="tns:QueryType" minOccurs="1" maxOccurs="1" />
      </sequence>
      <attribute name="startTime" type="dateTime" use="optional"/>
      <attribute name="endTime" type="dateTime" use="optional"/>
      <attribute name="notificationInterval"
        type="duration" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

961 8.3.2 Example

962 The following example shows a subscription to receive notification of changes to the object whose id value matches
 963 “urn:acme:person:Danyal”. The DeliveryInfo specifies the SOAP endpoint where the server should deliver the Noti-
 964 fication.

```
<rim:RegistryObject xsi:type="rim:SubscriptionType"
  id="urn:acme:Subscription:subscribeToDanyal"
  startTime="2008-01-10T19:20:30+01:00" endTime="2009-01-10T19:20:30+01:00"
  ...>
  <DeliveryInfo>
    <NotifyTo>
      <wsa:Address rim:endpointType="urn:oasis:names:tc:ebxml-
regrep:endPointType:soap">http://www.acme.com/notificationListener</wsa:Address>
    </NotifyTo>
  </DeliveryInfo>
  <Selector queryDefinition="urn:oasis:names:tc:ebxml-
  regrep:query:GetObjectById">
    <Slot name="id">
      <SlotValue xsi:type="rim:StringValueType">
        <Value>urn:acme:person:Danyal</Value>
      </SlotValue>
    </Slot>
  </Selector>
</rim:RegistryObject>
```

965 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

966

- 967 ● Attribute `startTime`, `endTime` – Define the time window within which the subscription is valid.
 - 968 ○ A server MUST use the current time at the time of submission of Subscription as value for the start-
 - 969 Time attribute if it is unspecified.
 - 970 ○ The Subscription validity window MUST be inclusive of the `startTime` and `endTime`.
 - 971 ○ If `endTime` is unspecified then a server MUST assume the Subscription is valid at any time any time
 - 972 since `startTime` inclusively.
- 973 ● Element `DeliveryInfo` – Specifies the information needed by the server to deliver notifications for the sub-
- 974 scription. It includes the reference to the endpoint where notifications should be delivered.
 - 975 ○ A server MUST deliver notifications that match the Selector query for a valid `SubscriptionType` in-
 - 976 stance to the endpoint specified by each `DeliveryInfo` element of the `SubscriptionType` instance.
 - 977 ○ If no `DeliveryInfo` element is present then client MUST use the canonical query `GetNotification` via the
 - 978 Query protocol to “pull” the pending notification if any at a time of their choosing as defined in ebRS.
- 979 ● Attribute `notificationInterval` – Specifies the duration that a server MUST wait between delivering success-
- 980 ively notifications to the client. The client specifies this attribute in order to control the frequency of notifica-
- 981 tion communication between server and client.
 - 982 ○ A server MUST deliver any pending notifications within the interval specified by this attribute.
 - 983 ○ A server MUST NOT deliver the same event more than once for the same subscription.
- 984 ● Element `Selector` – Specifies the query that the server MUST invoke to determine whether an event
- 985 matches a subscription or not. If the result of the query contains an object that is affected by an event that
- 986 has not yet been delivered to the subscriber then the event matches the subscription.

987

988 8.4 DeliveryInfoType

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

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

991 reference to the endpoint where notifications should be delivered. The endpoint reference is typically one of the fol-

992 lowing types:

- 993 ● SOAP service endpoint
- 994 ● REST service endpoint
- 995 ● E-mail address endpoint
- 996 ● Software plugin endpoint that is configured within the same process as the registry server

997 8.4.1 Syntax

```
<complexType name="DeliveryInfoType">
  <complexContent>
    <extension base="tns:ExtensibleObjectType">
      <sequence>
        <element name="NotifyTo"
          type="wsa:EndpointReferenceType" minOccurs="1" maxOccurs="1" />
      </sequence>
      <attribute name="notificationOption" type="tns:objectReferenceType"
default="urn:oasis:names:tc:ebxml-regrep:NotificationOptionType:ObjectRefs"/>
    </extension>
  </complexContent>
</complexType>
```

998 **8.4.2 Description**

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

999

- 1000 ● Attribute notificationOption – Specifies the modality of how notifications are to be delivered to the sub-
1001 scriber. Its value MUST reference a ClassificationNode in the canonical NotificationOptionType Classifica-
1002 tionScheme.
 - 1003 ○ urn:oasis:names:tc:ebxml-regrep:NotificationOptionType:Objects – Indicates that the server MUST
1004 provide complete RegistryObjectType instances in notifications delivered to the subscriber when this
1005 mode is specified.
 - 1006 ○ urn:oasis:names:tc:ebxml-regrep:NotificationOptionType:ObjectRefs – Indicates that the server MUST
1007 provide ObjectRefType instances rather than complete RegistryObjectType instances in notifications
1008 delivered to the subscriber when this mode is specified. A client MAY pull the complete RegistryOb-
1009 jectType instances using Query protocol after receiving the notification.
- 1010 ● Element NotifyTo – Specifies the endpoint reference for the endpoint where the server should deliver noti-
1011 fications for the Subscription.
 - 1012 ○ The type of this element is wsa:EndpointReferenceType as defined by [WSA-Core]
 - 1013 ○ The NotifyTo element has a <wsa:Address> sub-element
 - 1014 ○ The content of the <wsa:Address> element is a string representing the endpoint address which
1015 SHOULD be a URI
 - 1016 ○ The type of endpoint (SOAP, REST, email, ...) is indicated by an extension attribute rim:endpointType
1017 defined on the <wsa:Address> element as follows:
 - 1018 ■ If endpoint is a SOAP web service then the rim:endpointType attribute value MUST be
1019 “urn:oasis:names:tc:ebxml-regrep:endPointType:soap”
 - 1020 ■ If endpoint is a REST web service then the rim:endpointType attribute value MUST be
1021 “urn:oasis:names:tc:ebxml-regrep:endPointType:rest”
 - 1022 ■ If endpoint is an email address then the rim:endpointType attribute value MUST be
1023 “urn:oasis:names:tc:ebxml-regrep:endPointType:mail”
 - 1024 ■ If endpoint is a software plugin then the rim:endpointType attribute value MUST be
1025 “urn:oasis:names:tc:ebxml-regrep:endPointType:plugin”

1027 **8.5 NotificationType**

1028 **Base Type:** RegistryObjectType

1029 This type represents a notification that is sent by the server to a client to notify it of server events that are of interest
1030 to the client.

1031

1032 **8.5.1 Syntax**

```
<complexType name="NotificationType">
```



```

<complexContent>
  <extension base="tns:RegistryObjectType">
    <sequence>
      <element name="Event" type="tns:AuditableEventType"
        minOccurs="1" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="subscription"
      type="tns:objectReferenceType" use="required"/>
    </extension>
  </complexContent>
</complexType>
<element name="Notification" type="tns:NotificationType"/>

```

1033 8.5.2 Example

1034 The following example shows a Notification sent by the server for the subscription in earlier example. It notifies the
 1035 subscriber that the object with id “urn:acme:person:Danyal” has changed.

```

<Notification subscription="urn:acme:Subscription:subscribeToDanyal" ...>
  <Event user="123456" timestamp="2008-10-17T15:44:29.637" ...>
    <Action eventType="urn:oasis:names:tc:ebxml-regrep:EventTypes:Created">
      <AffectedObjectRefs>
        <ObjectRef id="urn:acme:person:Danyal"/>
      </AffectedObjectRefs>
    </Action>
  </Event>
</Notification>

```

1036 8.5.3 Description

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

1037

- 1038 ● Element Event – Represents an Event that is of interest to the subscriber.
- 1039 ○ Unlike an AuditableEvent element that contains all objects affected by it, the Event element MUST
- 1040 only contain objects that match the selector query of the SubscriptionType instance. It has only a subset
- 1041 of affected objects compared to the actual AuditableEvent it represents. The subset of affected objects
- 1042 MUST be those that match the selector query for the subscription.
- 1043 ○ The Action elements within the Event element MUST contain a RegistryObjectList element if sub-
- 1044 scription's notificationOption is “Push”.
- 1045 ○ The Action elements within the Event element MUST contain a RegistryObjectRefList element if sub-
- 1046 scription's notificationOption is “Pull”.
- 1047 ● Attribute subscription – References the SubscriptionType instance for which this is a Notification.

1048 9 Federation Information Model

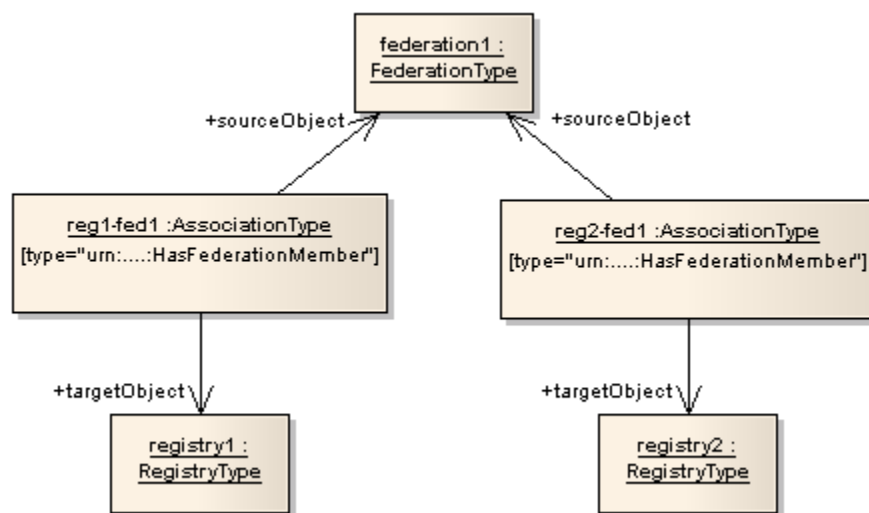
1049 This chapter describes the information model that support the definition of registry federations. A registry federation
1050 is a set of ebXML RegRep servers that have voluntarily agreed to form a loosely coupled union. Such a federation
1051 may be based on common business interests or membership in a community-of-interest. Registry federations en-
1052 abled clients to query the content of their member servers using federated queries as if they are a single logical
1053 server.

1054 9.1 Federation Configuration

1055 A federation is created by the creation of a FederationType instance. A federation may have any number of registries
1056 as well as other federations as its members.

1057 Membership of a registry or federation within a parent federation is established by creating an Association between
1058 the RegistryType or FederationType instance representing the registry or federation seeking membership, and the
1059 FederationType instance representing the parent federation as follows:

- 1060 ● The Association MUST have its associationType be the id of the canonical ClassificationNode “HasFedera-
1061 tionMember”
- 1062 ● The Association MUST have as its sourceObject the FederationType instance representing the parent feder-
1063 ation
- 1064 ● The Association MUST have as its targetObject the RegistryType or FederationType instance that is seek-
1065 ing membership within the parent federation as shown in Illustration 10.



1066 *Illustration 10: Federation Information Model*

1067 Thus a Federation is defined by a tree where a FederationType instances are root and intermediate n, RegistryType
1068 instances are leaf nodes and HasFederationMember AssociationType instances are the edges between the nodes.
1069 This tree is referred to as the federation membership tree.

1071 9.2 RegistryType

1072 **Base Type:** RegistryObjectType

1073 RegistryType instances are used to represent an ebXML RegRep server. RegistryType instances are also used by a
 1074 server to advertise the capabilities it supports. A client MAY read the RegistryType instance for a server to determine
 1075 whether it is compatible with a server or not. Profiles of ebXML RegRep specifications MAY define canonical slots
 1076 to represents support for the profile as well as optional features defined by the profile.

1077 9.2.1 Syntax

```
<complexType name="RegistryType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <attribute name="operator"
        type="tns:objectReferenceType" use="required"/>
      <attribute name="specificationVersion"
        type="string" use="required"/>
      <attribute default="P1D" name="replicationSyncLatency"
        type="duration" use="optional"/>
      <attribute default="PT0S" name="catalogingLatency"
        type="duration" use="optional"/>
      <attribute name="conformanceProfile"
        use="optional" default="RegistryLite">
        <simpleType>
          <restriction base="NCName">
            <enumeration value="RegistryFull"/>
            <enumeration value="RegistryLite"/>
          </restriction>
        </simpleType>
      </attribute>
    </extension>
  </complexContent>
</complexType>
```

1078 9.2.2 Example

1079 The following example describes an ebXML RegRep server operated by organization with id “urn:acme:Organiza-
 1080 tion:acme-inc”, that implements the “RegistryFull” conformance level of version 4.0 of the ebXML RegRep spe-
 1081 cifications. The server performs replication synchronization once a day (P1D) and performs cataloging of submitted
 1082 content immediately when content is submitted.

```
<rim:RegistryObject xsi:type="rim:RegistryType"
  id="urn:acme:Registry:serviceRegistry"
  operator="urn:acme:Organization:acme-inc"
  specificationVersion="4.0"
  conformanceProfile="RegistryFull"
  replicationSyncLatency="P1D"
  catalogingLatency="PT0S"
  ...>
  ...
</rim:RegistryObject>
```

1083 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 (imme-	Server	Yes

			diately)		
specificationversion	objectReferenceType	1		Server	Yes

1084

- 1085 ● Attribute *catalogingLatency* - A RegistryType instance MAY have an attribute named *catalogingLatency*
1086 that specifies the maximum latency between the time a submission is made to the server and the time it gets
1087 cataloged by any cataloging services defined for the objects within the submission. The default value of
1088 PTOS indicates a duration of 0 seconds which implies that cataloging happens immediately when request is
1089 submitted.
- 1090 ● Attribute *conformanceProfile* - A RegistryType instance MAY have an attribute named *conformanceProfile*
1091 that declares the conformance profile that the server supports. The conformance profiles choices are “Re-
1092 gistryLite” and “RegistryFull” as defined by [regrep-rs-v4.0].
- 1093 ● Attribute *operator* - A RegistryType instance MUST have an attribute named *operator* that is a reference to
1094 the Organization instance representing the organization for the server’s operator. Since the same Organiza-
1095 tion MAY operate multiple registries, it is possible that the home registry for the Organization referenced
1096 by operator may not be the local registry.
- 1097 ● Attribute *replicationSyncLatency* - A RegistryType instance MAY have an attribute named *replicationSyn-*
1098 *cLatency* that specifies the maximum latency between the time when an original object changes and the
1099 time when its replica object within the local server gets updated to synchronize with the new state of the
1100 original object. The default value of P1D indicates a duration of once a day.
- 1101 ● Attribute *specificationVersion* - A RegistryType instance MUST have an attribute named *specificationVer-*
1102 *sion* that is the version of the ebXML RegReg Specifications it implements.

1103 9.3 FederationType

1104 **Base Type:** RegistryObjectType

1105 Federation instances are used to represent a registry federation. A FederationType instance has a set of RegistryType
1106 instances as its members. The membership of a RegistryType instance in a federationType instance is represented by
1107 an AssociationType instance whose type is HasFederationMember.

1108 9.3.1 Syntax

```
<complexType name="FederationType">
  <complexContent>
    <extension base="tns:RegistryObjectType">
      <attribute name="replicationSyncLatency"
        type="duration" use="optional" default="P1D" />
    </extension>
  </complexContent>
</complexType>
```

1109 9.3.2 Example

1110 The following example shows a Federation with two independently-operated ebXML RegRep servers as members.

```
<rim:RegistryObject xsi:type="rim:FederationType"
  id="urn:acme:Federation:supplierFederation"
  replicationSyncLatency="P1D" ...>
  ...
</rim:RegistryObject>

<rim:RegistryObject xsi:type="rim:AssociationType"
  sourceObject="urn:acme:Federation:supplierFederation"
  targetObject="urn:widgetInc:Registry:widget-inc"
```

```

type="urn:oasis:names:tc:ebxml-regrep:AssociationType:HasFederationMember"/>
<rim:RegistryObject xsi:type="rim:AssociationType"
sourceObject="urn:acme:Federation:supplierFederation"
targetObject="urn:supplierInc:Registry:supplier-inc"
type="urn:oasis:names:tc:ebxml-regrep:AssociationType:HasFederationMember"/>

```

1111 **9.3.3 Description**

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

1112

- 1113 ● Attribute replicationSyncLatency - A FederationType instance MAY specify a *replicationSyncLatency* at-
- 1114 tribute that describes the time duration that is the amount of time within which a member of this Federation
- 1115 MUST synchronize itself with the current state of the Federation. Members of the Federation MAY use this
- 1116 parameter to periodically synchronize the federation metadata they MUST cache locally about the state of
- 1117 the Federation and its members. Such synchronization MAY be based upon the registry event notification
- 1118 capability.

1119

1120 10 Access Control Information Model

1121 This chapter defines the Information Model used to control access to RegistryObjects and RepositoryItems managed
1122 by it. It also defines a normative profile of [XACML] for ebXML RegRep.

1123 It is assumed that the reader is already familiar with [XACML]. This specification does not provide any introduction
1124 to [XACML].

1125 A server MUST support the roles of both Enforcement Point (PEP) and a Policy Decision Point (PDP) as defined in
1126 [XACML].

1127 The Access Control Model attempts to reuse terms defined by [XACML] wherever possible. The definitions of some
1128 key terms are duplicated here from [XACML] for convenience of the reader:

1129

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

1130

1131 10.1 Defining an Access Control Policy

1132 A RegistryObjectType instance is associated with exactly one Access Control Policy that governs “who” is author-
1133 ized to perform “what” action on that RegistryObject. This Access Control Policy is expressed as an [XACML] doc-
1134 ument which is the repositoryItem for an ExtrinsicObjectType instance. The Access Control Policy is published to
1135 the server as an ExtrinsicObject and repositoryItem pair using the Submit protocol defined by [regrep-rs-v4.0].

1136 The objectType attribute of this ExtrinsicObject MUST reference a descendent of the “XACML” ClassificationNode
1137 (e.g. “Policy” or PolicySet”) in the canonical ObjectType ClassificationScheme.

1138 10.2 Assigning Access Control Policy to a RegistryObject

1139 An Access Control Policy MAY be assigned to a RegistryObjectType instance using the canonical slot
1140 “urn:oasis:names:tc:ebxml-regrep:rim:RegistryObject:accessControlPolicy”. The value slot references the Extrinsic-
1141 icObject representing the Access Control Policy and contains the id of that ExtrinsicObject.

1142 If a RegistryObjectType instance does not have an Access Control Policy explicitly associated with it via the canon-
1143 ical slot with name “urn:oasis:names:tc:ebxml-regrep:rim:RegistryObject:accessControlPolicy”, then it is implicitly
1144 associated with the [default Access Control Policy](#) defined for the server.

1145

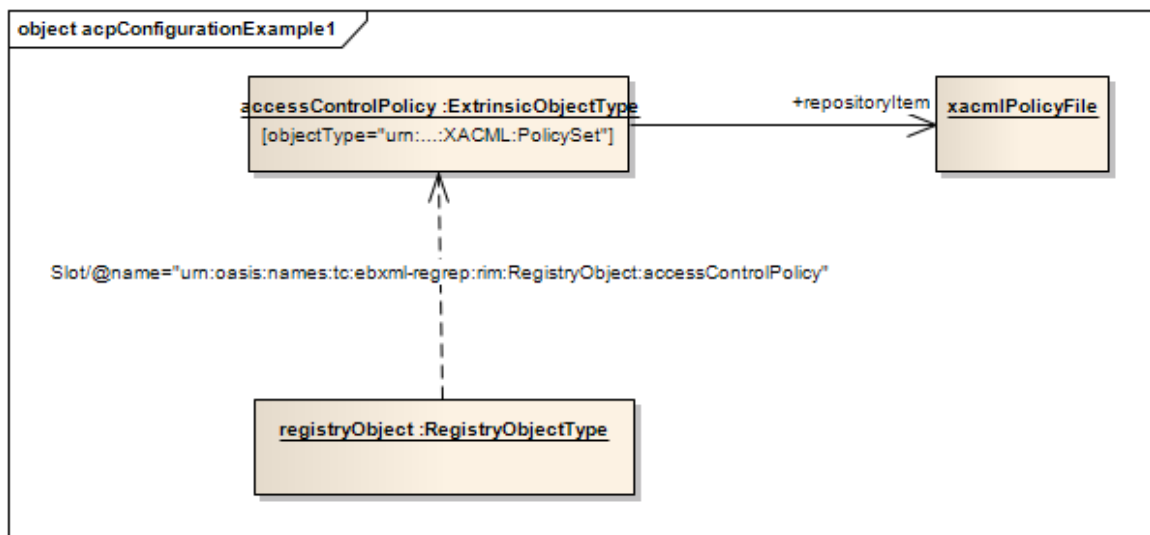


Illustration 11: Assigning Access Control Policy to a RegistryObject

1146

1147

1148 Illustration 11 shows a UML instance diagram where an Organization instance *org* references an ExtrinsicObject in-
1149 stance *accessControlPolicy* as its Access Control Policy object using the canonical *accessControlPolicy* slot.

1150 10.2.1 Default Access Control Policy for a RegistryObject

1151 A server MUST support a default Access Control Policy. A server MAY implement any default access control policy.
1152 The default Access Control Policy applies to all RegistryObjectType instances that do not explicitly have an Access
1153 Control Policy assigned.

1154 This following specify the semantics of a suggested default Access Control Policy that a server SHOULD imple-
1155 ment:

- 1156 ● An unauthenticated client is permitted to perform *read* actions (that do not modify the state of resources) on
1157 any resource

- 1158 ● An authenticated client with authentication credentials of a registered user is permitted all actions on Re-
1159 gistryObjects submitted by the user
- 1160 ● A authenticated client with authentication credentials that are assigned the canonical subject role of
1161 "urn:oasis:names:tc:ebxml-regrep:SubjectRole:RegistryAdministrator" is permitted to perform any action
1162 on any object

1163 10.2.2 Access Control Policy Inheritance

1164 A RegistryObjectType instance that does not explicit define an Access Control Policy MAY inherit an Access Con-
1165 trol Policy from its nearest RegistryPackageType ancestor if it is in the membership hierarchy of a RegistryPackage-
1166 Type instance.

1167 An Access Control Policy for members of a RegistryPackageType instance MAY be assigned to the RegistryPack-
1168 ageType instance using the canonical slot "urn:oasis:names:tc:ebxml-regrep:rim:RegistryPackage:memberAc-
1169 cessControlPolicy". The value slot references the ExtrinsicObject representing the Access Control Policy and con-
1170 tains the id of that ExtrinsicObject. The member Access Control Policy is implicitly inherited as the applicable Ac-
1171 cess Control Policy for any member RegistryObjectType instance that does not have an explicit Access Control
1172 Policy assigned to it.

1173 In the event that a RegistryObjectType instance has a member Access Control Policy defined from two RegistryP-
1174 ackageType ancestors at the same ancestry level a server MAY choose any mechanism to select one of the two
1175 member Access Control Policies.

1176 10.2.2.1 Algorithm for Getting Applicable Access Control Policy

1177 A server MUST implement the following algorithm for determining the applicable Access Control Policy for a Re-
1178 gistryObjectType instance:

- 1179 ● If an Access Control Policy is explicitly assigned to the object then use it
- 1180 ● If no Access Control Policy is explicitly assigned to the object then get the member Access Control Policy
1181 from a nearest RegistryPackageType ancestor of the object
- 1182 ● If no Access Control Policy is explicitly assigned to the object or inherited from a RegistryPackageType
1183 ancestor of the object then use the system-wide default Access Control Policy

1184 10.2.3 Performance Implications

1185 Excessive use of custom Access Control Policies MAY result in slower processing of registry requests in some re-
1186 gistry implementations. It is therefor suggested that, whenever possible, a submitter SHOULD reuse an existing Ac-
1187 cess Control Policy. Submitters SHOULD use good judgment on when to reuse or extend an existing Access Control
1188 Policy and when to create a new one.

1189

1190 10.3 Defining a Contextual Role

1191 A contextual role may be define by a RoleType instance within a server as defined next.

1192 10.3.1 RoleType

1193 **Base Type:** [RegistryObjectType](#)

1194 10.3.1.1 Syntax

```
<complexType name="RoleType">  
  <complexContent>  
    <extension base="tns:RegistryObjectType">
```



```

    <attribute name="type" type="tns:objectReferenceType" use="required"/>
  </extension>
</complexContent>
</complexType>

```

1195 10.3.2 Example

1196 The following examples shows a RoleType instances representing a contextual role of “ProjectLead” within the or-
 1197 ganizational context of an Organization TestSubmittingOrg1.

```

<rim:RegistryObject xsi:type="rim:RoleType"
  type="urn:oasis:names:tc:ebxml-regrep:SubjectRole:ProjectLead" ...>
  <rim:Slot name="urn:oasis:names:tc:ebxml-
  regrep:RoleAssociation:organizationContext">
    <rim:SlotValue xsi:type="rim:StringValueType">
      <rim:Value>urn:test:Organization:TestSubmittingOrg1</rim:Value>
    </rim:SlotValue>
  </rim:Slot>
</rim:RegistryObject>

```

1198 10.3.3 Description

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

1199

- 1200 ● Attribute type - Each RoleType instance MUST have a *type* attribute that identifies the type of role.
 - 1201 ○ The value of the type attribute SHOULD be a reference to a ClassificationNode within the canonical
 - 1202 SubjectRole ClassificationScheme.
 - 1203 ○ A server MUST support the canonical subject role types as defined by the canonical SubjectRole Clas-
 - 1204 sificationScheme. Deployments and profiles may extend the canonical SubjectRole ClassificationS-
 - 1205 cheme by adding additional ClassificationNodes to it.
- 1206 ● The RoleType instance may have any number of Slots that provide context for the role as a name/value
- 1207 pair. The name attribute of each such context slot provides the context key while the value of the Slot (typ-
- 1208 ically a string) provides the context value

1209 10.4 Assigning a Contextual Role to a Subject

1210 A subject such as a registered user MAY be assigned a contextual role by associating the id attribute value of the Ro-
 1211 leType instance representing the contextual role with the id of the subject. This specification does not define how
 1212 such an association is made. Implementations may provide this association in an implementation specific manner.
 1213 For example, in an LDAP based identity management system this may be done by making the node representing the
 1214 subject to be a member of a groupOfName node as illustrated below:

```

#person ProjectLead1 definition
dn: uid=ProjectLead1,...
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Project Lead 1
sn: ProjectLead1
uid: ProjectLead1

#Role TestSubmittingOrg1ProjectLead definition
dn: cn=urn:test:Role:TestSubmittingOrg1ProjectLead,...
objectclass: top

```

```
objectclass: groupOfNames
cn: urn:test:Role:TestSubmittingOrg1ProjectLead
member: uid=ProjectLead1,ou=people,...
```

1215 10.5 Action Matching

1216 An XACML Access Control Policy MAY use an action identifier associated with the action as an action attribute
1217 within <xacml:ActionMatch> elements to match the action that is authorized for a subject on a resource.

1218 The following requirements are defined for a server for action matching:

- 1219 ● A server MUST specify the action identifier in the request context for an XACML decision request using a
1220 <xacml:Request>/<xacml:Action>/<xacml:Attribute> element
 - 1221 ○ The <xacml:Attribute> element MUST have an AttributeId attribute with content
1222 "urn:oasis:names:tc:xacml:1.0:action:action-id"
 - 1223 ○ The <xacml:Attribute> element MUST have a DataType attribute with value
1224 "http://www.w3.org/2001/XMLSchema#string"
 - 1225 ○ The <xacml:Attribute> MUST have a <xacml:AttributeValue> element whose content MUST be the
1226 id attribute of the ClassificationNodeType instance within the canonical ActionTypeScheme that rep-
1227 resent the requested action
- 1228 ● The following requirements are defined for an Access Control Policy for action matching:
- 1229 ● The policy MAY match an action using an <xacmlp:ActionMatch> element
- 1230 ● The <xacmlp:ActionMatch> element MUST have a <xacmlp:AttributeValue
1231 DataType="http://www.w3.org/2001/XMLSchema#string"> element whose content MUST be the id attrib-
1232 ute of a ClassificationNodeType instance within the canonical ActionTypeScheme
- 1233 ● The <xacmlp:ActionMatch> element MUST have a <xacmlp:ActionAttributeDesignator
1234 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" DataType="http://www.w3.org/2001/XMLS-
1235 chema#string"/> element

1236

1237 The following example shows an Action that matches the "Read" action.

```
<Target>
  <Actions>
    <Action>
      <ActionMatch
        MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string">
          urn:oasis:names:tc:exml-regrep:ActionType:read
        </AttributeValue>
        <ActionAttributeDesignator
          AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
```

1238

1239 **10.5.1 Action Attribute: *reference-source***

1240 This attribute is only relevant to the “Reference” action. This attribute MAY be used to specify the object from
1241 which the reference is being made to the resource being protected. The AttributeId of this attribute MUST be
1242 “urn:oasis:names:tc:ebxml-regrep:rim:acp:subject:reference-source”. The value of this attribute MUST be the value
1243 of the id attribute for the object that is the source of the reference. A server MUST specify this attribute for a refer-
1244 ence action.

1245 **10.5.2 Action Attribute: *reference-source-attribute***

1246 This attribute is only relevant to the “Reference” action. This attribute MAY be used to specify the attribute name
1247 within the RegistryObjectType that the reference-source object is an instance of. A server MUST specify this attribute
1248 for a reference action. The AttributeId of this attribute MUST be “urn:oasis:names:tc:ebxml-regrep:rim:acp:subject:
1249 reference-source-attribute”. The value of this attribute MUST be the name of an attribute within the RIM type that is
1250 the type for the reference source object.

1251 For example, if the reference source object is an Association instance then the reference-source-attribute MAY be
1252 used to specify the values “sourceObject” or “targetObject” to restrict the references to be allowed from only spe-
1253 cific attributes of the source object. This enables, for example, a policy to only allow reference to objects under its
1254 protection only from the sourceObject attribute of an Association instance.

1255 **10.6 Subject Matching**

1256 An XACML Access Control Policy MAY use the identity and roles associated with the subject as subject attributes
1257 within <xacml:SubjectMatch> elements to match the subject that is authorized for an action on a resource.

1258 The following requirements are defined for a server for subject matching:

- 1259 ● A server MUST specify the subject identifier in the request context for an XACML decision request using a
1260 <xacml:Request>/<xacml:Subject>/<xacml:Attribute> element
 - 1261 ○ The <xacml:Attribute> element MUST have an AttributeId attribute with value
1262 "urn:oasis:names:tc:xacml:1.0:subject:subject-id"
 - 1263 ○ The <xacml:Attribute> element MUST have a DataType attribute with value
1264 "<http://www.w3.org/2001/XMLSchema#string>"
 - 1265 ○ The <xacml:Attribute> MUST have a <xacml:AttributeValue> element whose content MUST be the
1266 unique id associated with the requestor
- 1267 ● A server MUST specify any subject roles in the request context for an XACML decision request using a
1268 <xacml:Request>/<xacml:Subject>/<xacml:Attribute> element. This specification does not define how
1269 roles are assigned to a subject. Implementations SHOULD provide that functionality in an implementation-
1270 specific manner.
 - 1271 ○ The <xacml:Attribute> element MUST have an AttributeId attribute with value
1272 "urn:oasis:names:tc:ebxml-regrep:3.0:rim:acp:subject:role"
 - 1273 ○ The <xacml:Attribute> element MUST have a DataType attribute with value
1274 "<http://www.w3.org/2001/XMLSchema#string>"
 - 1275 ○ The <xacml:Attribute> MUST have a <xacml:AttributeValue> element whose content MUST be the
1276 id attribute value of a RoleType instance associated with the requestor

1277 **10.6.1 Matching Subjects By Id**

1278 The following requirements are defined for an Access Control Policy for subject matching by the subject id:

- 1279 ● The policy MAY match a subject by id using an <xacmlp:SubjectMatch> element

1280 ○ The <xacmlp:SubjectMatch> element MUST have a <xacmlp:AttributeValue
1281 DataType="http://www.w3.org/2001/XMLSchema#string"> element whose content MUST be the
1282 unique id of the subject

1283 ○ The <xacmlp:SubjectMatch> element MUST have a <xacmlp:SubjectAttributeDesignator
1284 AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
1285 DataType="http://www.w3.org/2001/XMLSchema#string"/> element

1286

1287 The following example shows a Subject that matches a registered user with id “urn:acme:person:Danyal”:

```
<Target>
  <Subjects>
    <Subject>
      <SubjectMatch
        MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string">
            urn:acme:person:Danyal
        </AttributeValue>
        <SubjectAttributeDesignator
          AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
  </Target>
```

1288

1289 10.6.2 Matching Subject By Role

1290 The following requirements are defined for an Access Control Policy for subject matching by a subject role:

- 1291 ● The policy MAY match a subject by a contextual role using an <xacmlp:Condition> element
 - 1292 ○ The <xacmlp:Condition> element MUST use an
1293 <xacmlp:Apply/@FunctionId="urn:oasis:names:tc:ebxml-regrep:4.0:rim:acp:function:matches-role">
1294 element to invoke the “[matches-role](#)” XACML function as defined by this specification

1295

1296 The following example shows a Subject that matches a subject role “ProjectLead” within the organizational context
1297 of Organization TestSubmittingOrg1 and register context of Register TestRegister1:

```
<Rule Effect="Permit" RuleId="urn:test:customACP1:rule:restricted-delete">
  <Target/>

  <Condition>
    <Apply FunctionId="urn:oasis:names:tc:ebxml-
regrep:4.0:rim:acp:function:matches-role">
      <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:ebxml-
regrep:3.0:rim:acp:subject:role"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
      <AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#string">urn:oasis:names:tc:ebxml-
regrep:SubjectRole:ProjectLead</AttributeValue>
      <AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#string">urn:oasis:names:tc:ebxml-
regrep:RoleAssociation:organizationContext</AttributeValue>
      <AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#string">urn:test:Organization:TestS
ubmittingOrg1</AttributeValue>
    </Apply>
  </Condition>
</Rule>
```

```

<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">urn:oasis:names:tc:ebxml-
regrep:RoleAssociation:registerContext</AttributeValue>
  <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">urn:test:Register:TestRegis
ter1</AttributeValue>
  </Apply>
</Condition>

</Rule>

```

1298

1299 10.7 Resource Matching

1300 An XACML Access Control Policy MAY use the id associated with the resource as resource attributes within
 1301 <xacml:ResourceMatch> elements to match a resource within an authorization decision for an action on the re-
 1302 source.

1303 The following requirements are defined for a server for resource matching:

- 1304 ● A server MUST specify the resource identifier in the request context for an XACML decision request using
 1305 a <xacml:Request>/<xacml:Resource>/<xacml:Attribute> elementThe <xacml:Attribute> element
 1306 MUST have an AttributeId attribute with value "urn:oasis:names:tc:xacml:1.0:resource:resource-id"
 - 1307 ○ The <xacml:Attribute> element MUST have a DataType attribute with value
 1308 "<http://www.w3.org/2001/XMLSchema#string>"
 - 1309 ○ The <xacml:Attribute> MUST have a <xacml:AttributeValue> element whose content MUST be the
 1310 unique id associated with the resource
- 1311 ● A server MUST specify the owner attribute value of the RegistryObjectType resource in the request context
 1312 for an XACML decision request using a <xacml:Request>/<xacml:Resource>/<xacml:Attribute> ele-
 1313 ment
 - 1314 ○ The <xacml:Attribute> element MUST have an AttributeId attribute with value
 1315 "urn:oasis:names:tc:ebxml-regrep:3.0:rim:acp:resource:owner"
 - 1316 ○ The <xacml:Attribute> element MUST have a DataType attribute with value
 1317 "<http://www.w3.org/2001/XMLSchema#string>"
 - 1318 ○ The <xacml:Attribute> MUST have a <xacml:AttributeValue> element whose content MUST be the
 1319 owner attribute value of the RegistryObjectType resource

1320

1321 10.7.1 Matching a Resource By Id

1322 The following requirements are defined for an Access Control Policy for resource matching by the resource's id:

- 1323 ● The policy MAY match a resource by id using an <xacmlp:ResourceMatch> element
 - 1324 ○ The <xacmlp:ResourceMatch> element MUST have a <xacmlp:AttributeValue
 1325 DataType="http://www.w3.org/2001/XMLSchema#string"> element whose content MUST be the
 1326 unique id of the resource
 - 1327 ○ The <xacmlp:ResourceMatch> element MUST have a <xacmlp:ResourceAttributeDesignator Attrib-
 1328 uteId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
 1329 DataType="http://www.w3.org/2001/XMLSchema#string"/> element

1330

1331 The following example shows a ResourceMatch that matches a resource with id “urn:acme:person:Danyal”:

```
<Target>
  <Resources>
    <Resource>
      <ResourceMatch
        MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string">
          urn:acme:person:Danyal
        </AttributeValue>
        <ResourceAttributeDesignator
          AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
```

1332 10.7.2 Matching a Resource Using XPATH Expression

1333 An XACML Access Control Policy MAY use any node in the XML document representing a RegistryObjectType in-
1334 stance within an <xacml:ResourceMatch> element. In this case, the <xacml:ResourceMatch> element SHOULD use
1335 an XPATH expression to match any part of the XML element representing the RegistryObjectType instance.

1336 The following example uses XPATH expression to match resource if it has a Slot with name “someSlotName”.

```
<Resource>
  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      urn:oasis:names:tc:ebxml-regrep:xsd:rim:4.0
    </AttributeValue>
    <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
  <ResourceMatch
    MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      //<rim:Slot>/@name="someSlotName"
    </AttributeValue>
    <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </ResourceMatch>
</Resource>
```

1337

1338 10.8 Canonical XACML Functions

1339 Section A.3 of [XACML] defines a set of standard functions. This section defines addition XACML functions that
1340 MUST be supported by an ebXML RegRep server that supports XACML based custom access control policies.
1341 XACML specifies the following functions. If an argument of one of these functions were to evaluate to "Indetermin-
1342 ate", then the function MUST be set to "Indeterminate".

1343 10.8.1 Function AssociationExists

1344 **Function ID:** urn:oasis:names:tc:ebxml-regrep:rim:acp:function:AssociationExists

1345

Parameter /	Name	Description	Data Type
-------------	------	-------------	-----------

Return			
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

1346

1347 **10.8.2 Function ClassificationNodeCompare**

1348 **Function ID:** urn:oasis:names:tc:ebxml-regrep:rim:acp:function:ClassificationNodeCompare

1349 A client MAY use this XACML function to test whether a resource's objectType attribute matches a specific object-
1350 Type or its sub-types.

1351

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

1352

1353 **10.8.3 Function matches-role**

1354 **Function ID:** urn:oasis:names:tc:ebxml-regrep:4.0:rim:acp:function:matches-role

1355

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		<p>MUST return “True” if and only if at least one RoleType instance assigned to the subject meets the following conditions:</p> <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 <p>MUST return “False” otherwise.</p>	http://www.w3.org/2001/XMLSchema#boolean

1356

1357 10.9 Constraints on XACML Binding

1358 This specification normatively defines the following constraints on the binding of the Access Control Model to
1359 [XACML]. These constraints MAY be relaxed in future versions of this specification.

- 1360 ● All Policy and PolicySet definitions MUST reside within an ebXML Registry as RepositoryItems.

1361

1362 10.10 Resolving Policy References

1363 An XACML PolicySet MAY reference XACML Policy objects defined outside the repository item containing the
1364 XACML PolicySet. A server implementation MUST be able to resolve such references. To resolve such references
1365 efficiently a server SHOULD be able to find the repository item containing the referenced Policy without having to
1366 load and search all Access Control Policies in the repository. This section describes the normative behavior that en-
1367 ables a server to resolve policy references efficiently.

1368 A server SHOULD define a Content Cataloging Service for the canonical XACML PolicySet objectType. The Poli-
1369 cySet cataloging service MUST automatically catalog every PolicySet upon submission to contain a special Slot
1370 with name ComposedPolicies. The value of this Slot MUST be a Set where each element in the Set is the id for a
1371 Policy object that is composed within the PolicySet.

1372 Thus a server is able to use an ad hoc query to find the repositoryItem representing an XACML PolicySet that con-
1373 tains the Policy that is being referenced by another PolicySet.