
**Electronic business eXtensible
Markup Language (ebXML) —**

**Part 3:
Registry and repository**





COPYRIGHT PROTECTED DOCUMENT

© ISO 2023

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Foreword.....	viii
Introduction.....	ix
1 Scope.....	1
2 Normative references.....	1
3 Terms and definitions	2
4 Registry information model.....	5
4.1 Introduction.....	5
4.1.1 Overview	5
4.1.2 XML Schema.....	5
4.1.3 Information model types: inheritance view	5
4.1.4 Extending ebRIM.....	6
4.1.5 Canonical ClassificationSchemes	6
4.2 Core Information Model	6
4.2.1 Overview	6
4.2.2 InternationalStringType	7
4.2.3 LocalizedStringType	8
4.2.4 ExtensibleObjectType	8
4.2.5 SlotType	9
4.2.6 ValueType.....	10
4.2.7 IdentifiableObjectType.....	11
4.2.8 RegistryObjectType	12
4.2.9 VersionInfoType	14
4.2.10 objectReferenceType	15
4.2.11 ObjectRefType.....	18
4.2.12 DynamicObjectRefType.....	18
4.2.13 ExtrinsicObjectType	19
4.2.14 CommentType.....	20
4.2.15 RegistryPackageType.....	21
4.2.16 ExternalIdentifierType.....	23
4.2.17 ExternalLinkType	24
4.3 Association information model	25
4.3.1 Overview	25
4.3.2 Source and target objects	26
4.3.3 Type of an association.....	26
4.3.4 AssociationType	26
4.3.5 Access control	27
4.4 Classification information model.....	27
4.4.1 Overview	27
4.4.2 TaxonomyElementType	28
4.4.3 ClassificationSchemeType.....	29
4.4.4 ClassificationNodeType.....	30
4.4.5 ClassificationType	32
4.5 Provenance information model.....	33
4.5.1 Overview	33
4.5.2 PostalAddressType	34
4.5.3 TelephoneNumberType	35
4.5.4 EmailAddressType	38

4.5.5	PartyType.....	37
4.5.6	PersonType	38
4.5.7	PersonNameType.....	38
4.5.8	OrganizationType	39
4.5.9	Associating organization with persons.....	40
4.5.10	Associating organization with organizations	40
4.5.11	Associating organizations with registry objects.....	41
4.6	Service information model	41
4.6.1	Overview	41
4.6.2	ServiceType.....	41
4.6.3	ServiceEndpointType.....	42
4.6.4	ServiceBindingType	43
4.6.5	ServiceInterfaceType.....	44
4.7	Query information model.....	44
4.7.1	Overview	44
4.7.2	QueryDefinitionType	45
4.7.3	ParameterType.....	46
4.7.4	QueryExpressionType	48
4.7.5	StringQueryExpressionType.....	48
4.7.6	XMLQueryExpressionType	49
4.7.7	QueryType	50
4.8	Event information model.....	51
4.8.1	Overview	51
4.8.2	AuditableEventType.....	52
4.8.3	ActionType.....	54
4.8.4	SubscriptionType	55
4.8.5	DeliveryInfoType	56
4.8.6	NotificationType.....	58
4.9	Federation information model.....	59
4.9.1	Overview	59
4.9.2	Federation configuration	59
4.9.3	RegistryType.....	59
4.9.4	FederationType.....	61
4.10	Access control information model	62
4.10.1	Overview	62
4.10.2	Defining an access control policy	63
4.10.3	Assigning access control policy to a registry object	63
4.10.4	Defining a contextual role.....	65
4.10.5	Assigning a contextual role to a subject.....	66
4.10.6	Action matching.....	66
4.10.7	Subject matching	68
4.10.8	Resource matching	69
4.10.9	Canonical XACML functions.....	71
4.10.10	Constraints on XACML binding	74
4.10.11	Resolving policy references.....	74
5	Registry services.....	74
5.1	Overview.....	74
5.2	Abstract protocol.....	74
5.2.1	Overview	74
5.2.2	RegistryRequestType	74
5.2.3	RegistryResponseType.....	75
5.2.4	RegistryExceptionType	76
5.2.5	Server Plugins.....	76
5.3	QueryManager interface.....	77
5.3.1	Overview	77

5.3.2	Parameterized queries	77
5.3.3	Query protocol	77
5.3.4	Parameterized query definition	82
5.3.5	Canonical Query: AdhocQuery	82
5.3.6	Canonical query: BasicQuery	83
5.3.7	Canonical query: ClassificationSchemeSelector	84
5.3.8	Canonical query: FindAssociations	85
5.3.9	Canonical query: FindAssociatedObjects	86
5.3.10	Canonical query: GarbageCollector	87
5.3.11	Canonical query: GetAuditTrailById	87
5.3.12	Canonical query: GetAuditTrailByLid	88
5.3.13	Canonical query: GetAuditTrailByTimeInterval	89
5.3.14	Canonical query: GetChildrenByParentId	89
5.3.15	Canonical query: GetClassificationSchemesById	91
5.3.16	Canonical query: GetRegistryPackagesByMemberId	91
5.3.17	Canonical query: GetNotification	92
5.3.18	Canonical query: GetObjectById	92
5.3.19	Canonical query: GetObjectsByLid	93
5.3.20	Canonical query: GetReferencedObject	93
5.3.21	Canonical query: KeywordSearch	94
5.3.22	Canonical query: RegistryPackageSelector	96
5.3.23	Query functions	97
5.3.24	Common patterns in query functions	100
5.3.25	Canonical functions	100
5.3.26	Query plugins	103
5.4	LifecycleManager interface	103
5.4.1	Overview	103
5.4.2	SubmitObjects protocol	103
5.4.3	UpdateObjects protocol	106
5.4.4	RemoveObjects Protocol	110
5.5	Version control	112
5.5.1	Overview	112
5.5.2	Version controlled resources	113
5.5.3	Versioning and id attribute	114
5.5.4	Versioning and lid attribute	114
5.5.5	Version identification for RegistryObjectType	114
5.5.6	Version identification for RepositoryItem	114
5.5.7	Versioning and references	115
5.5.8	Versioning of RegistryPackages	115
5.5.9	Versioning and RegistryPackage membership	115
5.5.10	Inter-version association	116
5.5.11	Version removal	116
5.5.12	Locking and concurrent modifications	116
5.5.13	Version creation	116
5.6	Validator interface	117
5.6.1	Overview	117
5.6.2	ValidateObjects protocol	117
5.6.3	Validator plugins	118
5.7	Cataloger interface	119
5.7.1	Overview	119
5.7.2	CatalogObjects protocol	119
5.7.3	Cataloger plugins	122
5.8	Subscription and notification	123
5.8.1	Overview	123
5.8.2	Server events	123
5.8.3	Notifications	124

5.8.4	Creating a subscription	124
5.8.5	Event delivery	125
5.8.6	NotificationListener interface	126
5.8.7	Notification protocol	126
5.8.8	Pulling notification on demand	127
5.8.9	Deleting a subscription	127
5.9	Multi-server features	127
5.9.1	Overview	127
5.9.2	RemoteObjects reference	127
5.9.3	Local replication of remote objects	128
5.9.4	Registry federations	130
5.10	Governance features	133
5.10.1	Overview	133
5.10.2	Representing a governance collaboration	134
5.10.3	Scope of governance collaborations	135
5.10.4	Assigning a governance collaboration	135
5.10.5	Determining applicable governance collaboration	136
5.10.6	Determining the registry process in a governance collaboration	136
5.10.7	Starting the registry process for a governance collaboration	136
5.10.8	Incoming messageFlows to registry process	137
5.10.9	Outgoing messageFlows from registry process	137
5.10.10	Canonical task patterns	137
5.10.11	XPATH extension functions	140
5.10.12	Default governance collaboration	141
5.11	Security features	142
5.11.1	Overview	142
5.11.2	Message integrity	142
5.11.3	Message confidentiality	142
5.11.4	User registration and identity management	142
5.11.5	Authentication	143
5.11.6	Authorization and access control	143
5.11.7	Audit trail	143
5.12	Native language support (NLS)	143
5.12.1	Overview	143
5.12.2	Terminology	143
5.12.3	NLS and registry protocol messages	144
5.12.4	NLS support in RegistryObjects	144
5.12.5	NLS and repository items	145
5.13	REST binding	146
5.13.1	Overview	146
5.13.2	Canonical URL	146
5.13.3	Query Protocol REST binding	147
5.14	SOAP binding	149
5.14.1	Overview	149
5.14.2	WS-Addressing SOAP headers	150
6	Conformance	150
6.1	Conformance for ebXML RegRep	150
6.2	QueryManager interface	151
6.2.1	Overview	151
6.2.2	Canonical queries	151
6.2.3	Canonical query functions	152
6.3	LifecycleManager interface	152
6.4	Version control	153
6.5	Validator interface	153
6.6	Cataloger interface	153

6.7 Subscription and notification	153
6.8 Multi-server features.....	154
6.9 Governance features.....	154
6.10 Security features.....	154
6.11 Native language support	155
6.12 REST binding.....	155
6.13 SOAP binding.....	156
Annex A (Normative) Protocol exceptions	157
Annex B (Normative) Namespace definitions.....	159
Annex C (Informative) Namespace references.....	161
Bibliography	162

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by the OASIS ebXML (Electronic business eXtensible Markup Language) Registry Technical Committee (as the "OASIS ebXML RegRep Version 4.0 specification") and drafted in accordance with its editorial rules. It was assigned to Technical Committee ISO/TC 154, *Processes, data elements and documents in commerce, industry and administration* and adopted under the "fast-track procedure".

A list of all parts in the ISO 15000 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

ebXML RegRep (registry and repository) is a standard defining service interfaces, protocols and information model for an integrated registry and repository. The repository stores electronic content while the registry stores metadata that describes the content in the repository.

To explain what an ebXML RegRep is we use the following familiar analogy. The ebXML RegRep is to electronic content, what your local library is to books and other published content. To make this analogy clearer one could look at comparisons enumerated in Table 1:

Your Local Library	ebXML RegRep
Manages books and all types of published material	Manages all types of electronic content
Has book shelves containing books and other published material	Has a "repository" containing all types of electronic content
Has a card catalog that describes the published material that is available in the book shelves	Has a "registry" that describes the electronic content that is available in the repository
Multiple libraries can voluntarily participate in a cooperative network and offer a unified service	Multiple ebXML RegRep instances can voluntarily participate in a cooperative network and offer a unified service

Table 1: ebXML RegRep comparison with your local library

An ebXML RegRep is capable of storing any type of electronic content such as XML documents, text documents, images, sound and video. Instances of such content are classified as "RepositoryItem" instances. Repository items are stored in a content *repository* provided by ebXML RegRep.

In addition to repository items ebXML RegRep also stores standardized metadata. Instances of such metadata are classified as "RegistryObject" instances (or one of its sub-types, as described later in this document). Registry objects are stored in a *registry* provided by ebXML RegRep. A registry object is like a card in the card catalog. All registry objects conform to a standard just like the cards in the card catalog conform to a standard. Every repository item shall have a registry object that describes it, just like every book must have a card in the card catalog. At the same time there are registry objects that do not describe any repository item and serve the purpose of defining capabilities of an ebXML RegRep instance.

This document describes Electronic business eXtensible Markup Language (ebXML) Part 3: Registry and Repository. The main clauses are:

- The Registry Information Model (ebRIM) which defines the types of metadata that can be stored in an ebXML RegRep server (Clause 4).
- The Registry Services and Protocols (ebRS) which define the services provided by an ebXML RegRep server and the protocols used by clients to interact with these services (Clause 5).
- Conformance to ebXML Registry and Repository (Clause 6).

This document also includes the following annexes:

- Annex A lists the standard exceptions that may be returned by various protocols defined in this document.
- Annex B provides a listing of namespaces that are defined in this document.
- Annex C provides a listing of namespaces that are referenced in this document.

Separately provided are:

- a) A set of XML schema files at <https://standards.iso.org/iso/15000/-3/ed-1/en/xsd/> define XML schema types and elements referenced in the core specification clauses.
- b) A set of WSDL files at <https://standards.iso.org/iso/15000/-3/ed-1/en/wsdl/> define the WSDL interfaces, bindings and services referenced in the core specification clauses.
- c) A set of XML files at <https://standards.iso.org/iso/15000/-3/ed-1/en/xml/> that contain the canonical data that shall be supported within every ebXML RegRep server.

Electronic business eXtensible Markup Language (ebXML) —Part 3: Registry and Repository

1 Scope

This part of ISO 15000 specifies service interfaces, protocols and information model for an integrated registry and repository. The repository stores electronic content while the registry stores metadata that describes the content in the repository.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

INTERNET ENGINEERING TASK FORCE (IETF). RFC 2828, *Internet Security Glossary*, 2000. Edited by R. Shirey. <https://www.rfc-editor.org/info/rfc2828>.

ISO. Information technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation, Schematron, International Standard ISO/IEC 19757-3, Geneva, Switzerland : ISO.

OASIS. *Web Services Security: Web Services Security: SAML Token Profile 1.1*. 1 February 2006. <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SAMLTOKENProfile.pdf>

OASIS. *Web Services Security: WS-Security Kerberos Token Profile 1.1*, February 2006. <http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-KerberosTokenProfile.pdf>

OASIS. *Web Services Security: SOAP Message Security 1.0*, 2004. Edited by Anthony Nadalin, et al Available from <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>.

OASIS. *Web Services Security UsernameToken Profile 1.0*, 2004. Edited by P. Hallam-Baker, et al. Available from <http://docs.oasis-open.org/wss/2004/01/>.

OASIS. *Web Services Security X.509 Certificate Token Profile*, 2004. Edited by P. Hallam-Baker, et al Available from <http://docs.oasis-open.org/wss/2004/01/>.

OASIS. *Web Services Security: SOAP Message Security 1.1*, 2005. Edited by Anthony Nadalin, et al Available from <http://docs.oasis-open.org/wss/v1.1/>.

OASIS. *Web Services Security UsernameToken Profile 1.1*, 2006. Edited by A. Nadalin, et al. Available from <http://docs.oasis-open.org/wss/v1.1/>.

OASIS. *Web Services Security X.509 Certificate Token Profile 1.1*, 2006. Edited by A. Nadalin, et al. Available from <http://docs.oasis-open.org/wss/v1.1/>.

OASIS. *eXtensible Access Control Markup Language (XACML) Version 2.01*. Edited by T. Moses. Available from http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

WORLD WIDE WEB CONSORTIUM (W3C). *SOAP Version 1.2 Part 1: Messaging Framework*. M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. Frystyk Nielsen, Editors, W3C Recommendation, June 24, 2003, <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>.

WORLD WIDE WEB CONSORTIUM (W3C). *SOAP Version 1.2 Part 2: Adjuncts*, M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, H. Frystyk Nielsen, Editors, W3C Recommendation, June 24, 2003, <http://www.w3.org/TR/2003/REC-soap12-part2-20030624/>.

WORLD WIDE WEB CONSORTIUM (W3C). *Web Services Addressing 1.0 - Core*, M. Gudgin, M. Hadley, T. Rogers, Editors, W3C Recommendation, May 9, 2006, <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>. Latest version available at <http://www.w3.org/TR/ws-addr-core/>.

WORLD WIDE WEB CONSORTIUM (W3C). *Web Services Addressing 1.0 - SOAP Binding*, M. Gudgin, M. Hadley, T. Rogers, Editors, W3C Recommendation, May 9, 2006, <http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/>.

WORLD WIDE WEB CONSORTIUM (W3C). *Web Services Addressing 1.0 - Metadata*, M. Gudgin, M. Hadley, T. Rogers, Ü. Yalçinalp, Editors, W3C Recommendation, September 4, 2007, <http://www.w3.org/TR/2007/REC-ws-addr-metadata-20070904/>.

WORLD WIDE WEB CONSORTIUM (W3C). *Web Services Description Language (WSDL) Version 1.1*, E. Christensen et al., Editors, W3C Note, March, 2001, <https://www.w3.org/TR/2001/NOTE-wsdl-20010315>.

WORLD WIDE WEB CONSORTIUM(W3C). *XML Schema Part 1: Structures Second Edition*, 2004. Edited by H. Thompson, <http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/>.

WORLD WIDE WEB CONSORTIUM(W3C). *XML Schema Part 2: Datatypes Second Edition*, 2004. Edited by P. Biron, et al. <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>.

WORLD WIDE WEB CONSORTIUM. *Extensible Markup Language (XML) 1.0* (Second Edition), 2000. Edited by T. Bray, et al. Available from <http://www.w3.org/TR/2000/REC-xml-20001006>.

WORD WIDE WEB CONSORTIUM. *XSL Transformations (XSLT) Version 1.0*, 1999. Edited by J. Clark.. Available from <http://www.w3.org/TR/1999/REC-xslt-19991116>.

3 Terms and definitions

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

- ISO Online browsing platform: available at <https://www.iso.org/obp>
- IEC Electropedia: available at <https://www.electropedia.org/>

3.1

access control

authority definitions and their enforcement for various types of actors regarding access to a *registry object* (3.21)

3.2

association

registry object (3.21) that represents directional relationship between two *registry objects* (3.21)

3.3

auditable event

long-term record of what actions caused changes in the state of a *registry object* (3.21)

3.4**canonical artefact**

set of *registry object* (3.21) values defined by the standard

3.5**cataloging**

registry service (3.23) that creates and stores additional metadata related to a *registry object* (3.21)

3.6**classification**

registry object (3.21) that classifies another *registry object* (3.21) by assigning a *classification node* (3.7) to it

3.7**classification node**

registry object (3.21) that represents a specific value that is part of a *classification scheme* (3.8)

3.8**classification scheme**

registry object (3.21) formed as a parent-child hierarchy of *classification nodes* (3.7)

3.9**external identity**

identity of a *registry object* (3.21) as per known identification scheme

3.10**federation**

set of *registry server* (3.22) instances that form a loosely coupled union

3.11**governance**

process of controlling the lifecycle of *registry objects* (3.21)

3.12**identity**

globally unique identity that is different between different versions of a *registry object* (3.21)

3.13**logical identity**

globally unique identity of a *registry object* (3.21) that does not change during its lifetime

3.14**messaging**

secure and reliable exchange of messages between a *registry client* (3.20) and a *registry server* (3.22)

3.15**native language support**

support to express properties of a *registry object* (3.21) in different languages

3.16**notification**

registry object (3.21) sent to a *registry client* (3.20) as an effect of registry events matching a relevant *subscription* (3.28)

3.17

object types classification scheme

canonical classification scheme that represents types of objects that a *registry server* (3.22) supports

3.18

provenance

set of data and processes that represents ownership and history of custodianship of a *registry object* (3.21)

3.19

query definition

registry object (3.21) that represents query that could be invoked by a *registry client* (3.20)

3.20

registry client

client that communicates with a *registry server* (3.22) via ebXML RegRep protocols

3.21

registry object

metadata stored in a *registry server* (3.22)

3.22

registry server

server that provides *registry services* (3.23)

3.23

registry service

service that provides storage and discovery of electronic artefacts (repository) and their metadata (registry)

3.24

replication

registry service (3.23) where a copy of a *registry object* (3.21) is stored and maintained in another *registry server* (3.22)

3.25

repository item

electronic artefact stored in a repository

3.26

service

registry object (3.21) that represents services offered by an actor

3.27

slot

typed name-value property that provides extensibility of a *registry object* (3.21)

3.28

subscription

registry object (3.21) that represents the interest of a *registry client* (3.20) interest in receiving *notifications* (3.16) when certain events happen in a *registry server* (3.22).

3.28

validation

registry service (3.23) that checks whether a *registry object* (3.21) is valid or not

3.30

version

distinct state of a *registry object* (3.21)

4 Registry information model

4.1 Introduction

4.1.1 Overview

This clause defines the ebXML RegRep registry information model (ebRIM).

4.1.2 XML Schema

The ebXML Registry Information Model is defined as an XML Schema in <https://standards.iso.org/iso/15000/-3/ed-1/en/xsd/rim.xsd>. It defines the metadata types and their relationships within this specification.

4.1.3 Information model types: inheritance view

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

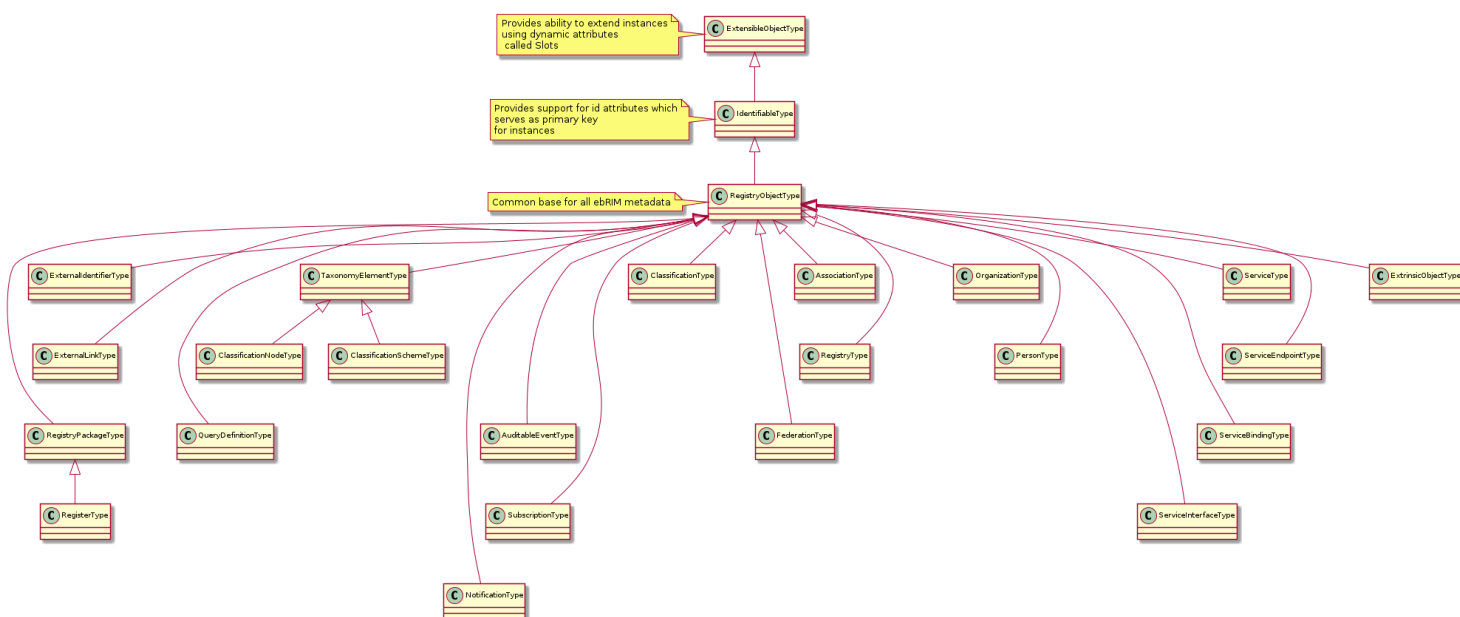


Figure 1 Information model inheritance view

Figure 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.

4.1.4 Extending ebRIM

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

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

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

- a) The server protocols as defined by the registry services (see clause 5) shall support extended types in a manner equivalent to pre-defined types. Specifically they shall support submit, update, versioning and removal of extended types derived directly or indirectly from RegistryObjectType
- b) The server shall be able to faithfully persist instances of extended types including all extension attributes and elements without any information loss
- c) The server shall be able to faithfully return instances of extension types including extension attributes and elements within a query response without any information loss
- d) This specification does not prescribe how a server may support the addition of new extension types to the server

4.1.5 Canonical ClassificationSchemes

ClassificationSchemes are defined in detail in the Classification Information Model (see clause 4.4). They are used by the specification for a wide variety of purposes within this specification.

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

The directory <https://standards.iso.org/iso/15000/-3/ed-1/en/xml/minDB/> contains the canonical ClassificationSchemes defined by this specification. The canonical ClassificationSchemes and ClassificationNodes are typically described using the rim:Description element within these files.

These canonical ClassificationSchemes shall be present in all conforming ebXML RegRep servers. These Canonical ClassificationSchemes may be extended by adding additional ClassificationNodes. However, a ClassificationNode defined normatively in the canonical ClassificationScheme definitions shall not be modified within a registry. In particular they shall preserve their canonical id attributes in all servers.

4.2 Core Information Model

4.2.1 Overview

The core information model is centered around the RegistryObjectType type as shown in Figure 2 below. Each type will be defined in detail in subsequent clauses.

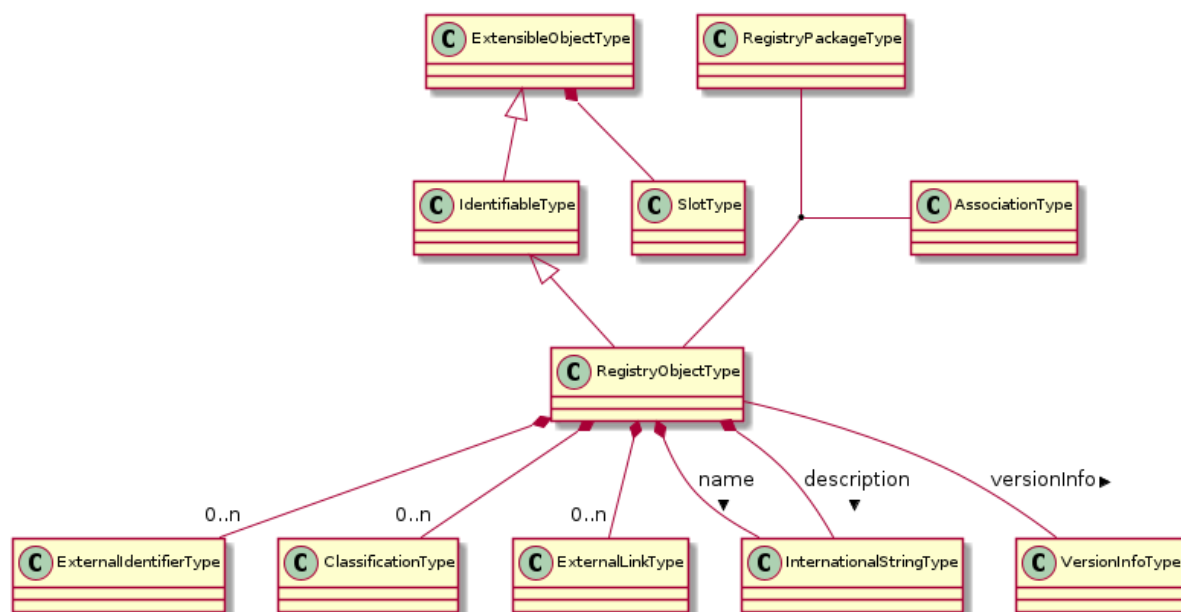


Figure 2 Core Information Model

4.2.2 InternationalStringType

4.2.2.1 Overview

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

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

4.2.2.2 XML schema type definition

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

EXAMPLE:

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

4.2.2.3 Description

The InternationalStringType is defined in Table 2.

Node	Type	Cardinality	Default Value	Specified By	Mutable

LocalizedString	LocalizedStringType	0..*		Client	Yes
-----------------	---------------------	------	--	--------	-----

Table 2 InternationalStringType

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

4.2.3 LocalizedStringType

4.2.3.1 Overview

This type allows the definition of a string value using the specified local language. It is used within the InternationalStringType as the type of the LocalizedString sub-element. The value of the `xml:lang` attribute shall be set to a tag for identifying a language as defined in RFC 4646. Note that the character set for all LocalizedStringType instances in an XML document is defined by the `charset` attribute within the *Content-Type* mime header for the XML document as shown in example below:

4.2.3.2 XML schema type definition

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

EXAMPLE:

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

4.2.3.3 Description

The LocalizedStringType is described in Table 3.

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

Table 3 LocalizedStringType

- Attribute `xml:lang` - Each LocalizedStringType instance may have a *xml:lang* attribute that specifies the language used by that LocalizedStringType instance. The `xml:lang` attribute and legal values for it are defined by Extensible Markup Language (XML) .
- Attribute `value` - Each LocalizedStringType instance shall have a *value* attribute that specifies the string value used by that LocalizedStringType instance

4.2.4 ExtensibleObjectType

4.2.4.1 Overview

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

4.2.4.2 XML schema type definition

```
<complexType name="ExtensibleObjectType" abstract="true">
```

```

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

```

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

```

<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>

```

4.2.4.3 Description

The `ExtensibleObjectType` is described in Table 4.

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

Table 4 `ExtensibleObjectType`

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

4.2.5 SlotType

4.2.5.1 Overview

Base Type: `ExtensibleObjectType`

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

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

Note that `SlotType` extends `ExtensibleObjectType` which means that a `SlotType` element may itself have `SlotType` sub-elements.

4.2.5.2 XML schema type definition

```

<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>

```

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

EXAMPLE

```
<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>
```

4.2.5.3 Description

Table 5 shows the description of SlotType.

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

Table 5 SlotType

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

4.2.6 ValueType

4.2.6.1 Overview

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

4.2.6.2 XML schema type definition

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

4.2.6.3 Description

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

- AnyValueType – This concrete sub-type of ValueType is used as a container for any well-formed XML element value in any namespace
- BooleanValueType - This concrete sub-type of ValueType is used as a container for a boolean value

- c) **CollectionValueType** - This concrete sub-type of **ValueType** is used as a container for a collection of values. It may be used to represent a **SlotValue** that is a collection of values where each value is represented by a **ValueType** instance

Attribute **collectionType** – Defines the type of collection for the **CollectionValueType**. Must be an **objectReferenceType** that references a **ClassificationNode** in the canonical **ClassificationScheme** **CollectionTypeScheme**. A server shall enforce the following semantics associated with the following canonical collection types:

- 1) **List** – Server shall maintain the order of the values in the collection
 - 2) **Set** – Server shall not allow duplicate values in the collection
 - 3) **Sorted Set** – Server shall not allow duplicate values in the collection and shall maintain a sort order according to the alphanumeric ordering of its elements according to the default locale associated with the server
 - 4) **Bag** – Server shall allow duplicate values and may not maintain order of values
- a) **DateTimeValueType** - This concrete sub-type of **ValueType** is used as a container for a **dateTime** value
- b) **DurationValueType** - This concrete sub-type of **ValueType** is used as a container for a **duration** value
- c) **FloatValueType** - This concrete sub-type of **ValueType** is used as a container for a **float** value
- d) **IntegerValueType** - This concrete sub-type of **ValueType** is used as a container for an **integer** value
- e) **InternationalStringValueType** - This concrete sub-type of **ValueType** is used as a container for an **InternationalStringType** value capable of holding strings in multiple locales
- f) **MapValueType** - This concrete sub-type of **ValueType** is used as a container for a **map** value. A **map** consists of **Entry** sub-elements where each **Entry** consists of an **EntryKey** and **EntryValue** both of which are of type **ValueType**
- g) **SlotValueType** – This concrete sub-type of **ValueType** is used as a container for a **SlotType** value
- h) **StringValueType** – This concrete sub-type of **ValueType** is used as a container for a **string** value
- i) **VocabularyTermValueType** - This concrete sub-type of **ValueType** is used as a container for a **VocabularyTermType** value. It is used to reference a term in some externally defined coded vocabulary (e.g. Dublin Core)

4.2.7 IdentifiableObjectType

4.2.7.1 Overview

Base Type: **ExtensibleObjectType**

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

4.2.7.2 XML schema type definition

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

```
</complexType>
```

EXAMPLE:

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

4.2.7.3 Description

Table 6 shows the description of IdentifiableObjectType.

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

Table 6 IdentifiableObjectType

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

4.2.8 RegistryObjectType**4.2.8.1 Overview**

Base Type: IdentifiableType

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

4.2.8.2 XML schema type definition

```
<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>
```

4.2.8.3 Description

Table 7 shows the description of RegistryObjectType.

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

Classification	ClassificationType	0..*		Client	Yes
Description	InternationalStringType	0..1		Client	Yes
ExternalIdentifier	ExternalIdentifierType	0..*		Client	Yes
ExternalLink	ExternalLinkType	0..*		Client	Yes
lid	string	0..1		Client or Server	No
Name	InternationalStringType	0..1		Client	Yes
objectType	objectReferenceType	0..1		Client or Server	No
owner	string	0..1		Server	Yes
status	objectReferenceType	0..1		Server	Yes
VersionInfo	VersionInfoType	0..1		Server	No

Table 7 RegistryObjectType

- a) Element Classification - A RegistryObjectType instance may have zero or more ClassificationType instances that are composed within the RegistryObject. A ClassificationType instance classifies the RegistryObject using a value within a ClassificationScheme
- b) Element Description - A RegistryObjectType instance may have textual description in a human readable and user-friendly form. This element is of type InternationalStringType and therefore capable of containing textual values in multiple local languages and character sets.
- c) 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.
- d) Attribute lid - A RegistryObjectType instance shall have a lid (Logical Id) attribute. The lid is used to refer to a logical RegistryObject in a version independent manner.
 - 1) All versions of a RegistryObject shall have the same value for the lid attribute. Note that this is in contrast with the id attribute that shall be unique for each version of the same logical RegistryObject.
 - 2) The lid attribute shall be specified by the client when creating the original version of a RegistryObject.
 - 3) The lid attribute specified when submitting the original version of a RegistryObject shall be globally unique and shall not be already in use as lid by another object.
- e) 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 therefore capable of containing textual values in multiple local languages and character sets.
- f) Attribute objectType - A RegistryObjectType instance has an *objectType* attribute.
 - 1) The value of the objectType attribute shall be a reference to a ClassificationNode in the canonical ObjectType ClassificationScheme.

- 2) A server shall support the object types as defined by the canonical ObjectType ClassificationScheme. The canonical ObjectType ClassificationScheme may easily be extended by adding additional ClassificationNodes to the canonical ObjectType ClassificationScheme.
 - 3) The *objectType* attribute shall be assigned by the server for all RegistryObjectType instances that are not instances of ExtrinsicObjectType.
 - 4) The *objectType* attribute may be assigned by the client for all RegistryObjectType instances that are instances of ExtrinsicObjectType
 - 5) If the client does not specify an objectType for an ExtrinsicObject then the server shall set its value to the id of the ClassificationNode representing ExtrinsicObject within the canonical ObjectType ClassificationScheme.
 - 6) A server shall set the correct objectType on a RegistryObject when returning it as a response to a client request.
- h) Attribute owner – Specifies the identifier associated with the registered user that owns the RegistryObjectType instance. It is used for access control and may be referenced within custom access control policies.
- i) Attribute status - A RegistryObjectType instance shall have a life cycle status indicator. The status is assigned by the server. Profiles may define additional status values if needed as slots on the RegistryObjectType instance. Such slots should have a type attribute with value “urn:oasis:names:tc:ebxml-regrep:rim:Slot:type:status”.
- 1) A server shall set the correct status on a RegistryObject when returning it as a response to a client request.
 - 2) A client should not set the status on a RegistryObject when submitting the object as this is the responsibility of the server.
 - 3) A server shall ignore the status on a RegistryObject when it is set by the client during submission or update of the object.
 - 4) The value of the status attribute should be a reference to a ClassificationNode in the canonical StatusType ClassificationScheme.
 - 5) A Registry shall support the status types as defined by the StatusType ClassificationScheme. The canonical StatusType ClassificationScheme may easily be extended by adding additional ClassificationNodes to the canonical StatusType ClassificationScheme.
- j) Element VersionInfo - Provides information about the specific version of a RegistryObjectType instance. The VersionInfo element is set by the server.

A server shall set a VersionInfo element for a RegistryObjectType instance. The VersionInfo element shall contain a versionName attribute whose value shall be unique for all versions of that logical RegistryObjectType.

4.2.9 VersionInfoType

4.2.9.1 Overview

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

4.2.9.2 XML schema type definition

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

EXAMPLE:

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

4.2.9.3 Description

Table 8 shows the description of VersionInfoType.

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

Table 8 VersionInfoType

- a) Attribute userVersionName - Represents a client-specified version name associated with the VersionInfo for a specific RegistryObject version
 - 1) A client may directly provide a value for the userVersionName attribute when submitting or updating an object
 - 2) A server shall persist any client specified userVersionName for an object without altering it in any form
- b) Attribute versionName - Represents the registry assigned version name identifying the VersionInfo for a specific RegistryObject version.
 - 1) The value for this attribute should not be specified by the client
 - 2) A server may silently ignore the value for this attribute if specified by the client
 - 3) The value for this attribute shall be automatically generated by the server and shall be defined for RegistryObjectType instances returned by server responses. The server is free to choose any scheme for generating the value for this attribute as long as the value is uniquely identifies a version for objects that have the same lid attribute value.

4.2.10 objectReferenceType

4.2.10.1 Overview

Base Type: xs:string

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

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

4.2.10.2 XML schema type definition

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

EXAMPLE:

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

4.2.10.3 Description

4.2.10.3.1 Local and Remote References

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

4.2.10.3.2 Static and Dynamic References

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

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

4.2.10.3.3 Encoding of objectReferenceType

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

- a) A static reference to a local reference target should be encoded as the value of the id attribute of the reference target.

The following example shows the reference attribute named primaryContact within Organization element. Its value is the value of the id attribute of a Person element.

EXAMPLE 1:

```
<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>
```

- b) A dynamic reference to a local reference target should be encoded to contain the id of a `DynamicObjectRefType` instance. The reference target is determined by the singleton result returned by the Query within the `DynamicObjectRef` instance.

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

EXAMPLE 2:

```
<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>
```

- c) A static or dynamic reference to a local reference target may be encoded to contain a Canonical URL for the local object as defined by the REST binding in the registry services (see clause 5).
- d) A static or dynamic reference to a remote reference target shall be encoded to contain a Canonical URL for the local object as defined by the REST binding in the registry services (see clause 5).

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

EXAMPLE 3:

```
<!-- 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>
```

4.2.11 ObjectRefType

4.2.11.1 Overview

Base Type: ExtensibleObjectType

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

4.2.11.2 XML schema type definition

```
<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"/>
```

4.2.11.3 Description

Table 9 shows the description of ObjectRefType.

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

Table 9 ObjectRefType

Attribute *id* - Every ObjectRef instance shall have an *id* attribute. The *id* attribute shall contain the value of the *id* attribute of the RegistryObject being referenced.

4.2.12 DynamicObjectRefType

4.2.12.1 Overview

Base Type: ObjectRefType

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

4.2.12.2 XML schema type definition

```
<complexType name="DynamicObjectRefType">
  <complexContent>
    <extension base="tns:ObjectRefType">
      <sequence>
```

```

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

```

4.2.12.3 Description

Table 10 shows the description of DynamicObjectRefType.

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

Table 10 DynamicObjectRefType

Element Query – Specifies the query that shall be invoked in order to determine the reference target.

- 1) This query shall match zero or one RegistryObjectType instances.
- 2) When the query matches zero RegistryObjectType instances, the dynamic object reference is considered to be unresolved.
- 3) A server shall return a ConfigurationException fault message if the query matches more than 1 RegistryObjectType instances.

4.2.13 ExtrinsicObjectType

4.2.13.1 Overview

Base Type: RegistryObjectType

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

4.2.13.2 XML schema type definition

```

<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:expectedContentTypes="*/*" type="base64Binary"/>
        </choice>
      </sequence>
      <attribute name="mimeType" type="tns:LongText" use="optional" />
    </extension>
  </complexContent>
</complexType>

```

EXAMPLE:

```

<rim:RegistryObject xsi:type="rim:ExtrinsicObjectType" mimeType="text/xml"
    objectType="urn:frebxml:registry:sample:profile:cpp:objectType:cppa: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>

```

4.2.13.3 Description

Table 11 shows the description of `ExtrinsicObjectType`.

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

Table 11 `ExtrinsicObjectType`

- a) Element `ContentVersionInfo` - Provides information about the specific version of a `RepositoryItem` that is associated with an `ExtrinsicObjectType` instance. The `ContentVersionInfo` element is set by the server.
 - 1) A server shall not set a `ContentVersionInfo` element for an `ExtrinsicObjectType` instance that does not have a `RepositoryItem`.
 - 2) A server shall set a `ContentVersionInfo` element for an `ExtrinsicObjectType` instance that has a `RepositoryItem`. The `ContentVersionInfo` element shall contain a `versionName` attribute whose value shall be unique for all versions of that `RepositoryItem`.
- b) Attribute `contentType` - An `ExtrinsicObjectType` instance may have a `contentType` attribute defined. The `contentType` provides information on the type of repository item cataloged by the `ExtrinsicObject` instance. The value of this attribute should be a registered MIME media type at <http://www.iana.org/assignments/media-types>.
- c) Element `repositoryItem` - Provides a base64 binary encoded representation of the repository item associated with the `ExtrinsicObjectType` instance (if any).
- d) Element `repositoryItemRef` - This element may be specified as an alternative to the `repositoryItem` element. Its type is `SimpleLinkType`. It uses `xlink:simpleAttrs` to specify a reference to a file on the client's local file system. This provides client libraries an alternative way to specify local files as repository item. The client library shall convert a `repositoryItemRef` element to a `repositoryItem` element prior to submitting it to the server.

4.2.14 CommentType

4.2.14.1 Overview

Base Type: `ExtrinsicObjectType`

This type represents a comment that may be associated with a `RegistryObjectType` instance. A comment associated with a `RegistryObject` models the familiar yellow POST-IT note metaphor used in attaching comments to paper documents.

4.2.14.2 XML schema type definition

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

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>
```

4.2.14.3 Description

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

- a) An authorized client may attach one or more comments to any RegistryObjectType instance using an Association between the RegistryObjectType instance and the CommentType instance
 Since a CommentType is itself a RegistryObjectType, a client may attach one or more comments to any CommentType instance
- b) The type of the Association shall reference the canonical HasComment ClassificationNode within the Canonical AssociationType ClassificationScheme
- c) The sourceObject of the Association shall be the RegistryObjectType instance
- d) The targetObject of the Association shall be the CommentType instance

4.2.15 RegistryPackageType

4.2.15.1 Overview

Base Type: RegistryObjectType

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

- a) A RegistryObjectType instance may be a member of multiple RegistryPackageType instances.
- b) A RegistryPackageType instance may have multiple RegistryObjectType instances as its members.
- c) Membership of a RegistryObjectType instance in a RegistryPackageType instance is established via an AssociationType instance where the type attribute references the canonical "HasMember" AssociationType within the canonical AssociationTypeScheme ClassificationScheme.
- d) As a convenience, the RegistryPackageType allows a RegistryObjectList to be specified by the client as a sub-element during submission of a RegistryPackage. The RegistryObjectList contains the set of RegistryObjectType instances that are members of the RegistryPackageType instance.

4.2.15.2 XML schema type definition

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

4.2.15.3 Examples

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

EXAMPLE 1

```
<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>
```

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

EXAMPLE 2

```
<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"/>
```

4.2.15.4 Description

Table 12 shows the description of RegistryPackageType.

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

RegistryObjectList	RegistryObjectList-Type	0..1		Client	Yes
--------------------	-------------------------	------	--	--------	-----

Table 12 RegistryPackageType

Element RegistryObjectList – This element allows clients to specify members of the RegistryPackage instance using a simpler alternative to “HasMember” AssociationType instances.

A server shall replace the RegistryObjectList with AssociationType instances such that each RegistryObjectType instance is replaced with an AssociationType instance with type “urn:oasis:names:tc:ebxml-regrep:AssociationType:HasMember”, with sourceObject specifying the id of the RegistryPackage instance and with targetObject specifying the id of the RegistryObjectType instance

4.2.16 ExternalIdentifierType

4.2.16.1 Overview

Base Type: RegistryObjectType

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

4.2.16.2 XML schema type definition

```
<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>
```

4.2.16.3 Example

The following example shows an Organization instance with its tax payer id specified using an ExternalIdentifierType instance.

EXAMPLE

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

4.2.16.4 Description

Table 13 shows the description of ExternalIdentifierType.

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

identificationScheme	objectReferenceType	1		Client	Yes
registryObject	objectReferenceType	0..1		Client	No
value	LongText	1		Client	Yes

Table 13 ExternalIdentifierType

- a) Attribute identificationScheme - Each ExternalIdentifier instance shall have an identificationScheme attribute that references a ClassificationScheme. This ClassificationScheme defines the namespace within which an identifier is defined using the value attribute for the RegistryObjectType instance referenced by the RegistryObject attribute.
- b) Attribute registryObject - Each ExternalIdentifier instance may have a *registryObject* attribute specified. This attribute references the parent RegistryObjectType instance for which this is an ExternalIdentifier.
 - 1) This attribute shall be specified when a client submits an ExternalIdentifier separately from its parent RegistryObjectType instance
 - 2) This attribute may be unspecified when a client submits an ExternalIdentifier as a sub-element of its parent RegistryObjectType instance. In such cases the server shall set this attributes value to the value of the id attribute of the parent RegistryObjectType instance.
 - 3) Attribute value - Each ExternalIdentifier instance shall have a *value* attribute that provides the identifier value for this ExternalIdentifier (e.g., the tax payer id in example above).

4.2.17 ExternalLinkType

4.2.17.1 Overview

Base Type: RegistryObjectType

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

4.2.17.2 XML schema type definition

```
<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>
```

4.2.17.3 Example

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

EXAMPLE

```
<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>

```

4.2.17.4 Description

Table 14 shows the description of ExternalLinkType.

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

Table 14 ExternalLinkType

- Element ExternalRef - Each ExternalLink instance shall have an ExternalRef sub-element defined. This element provides a URI to the external resource pointed to by this ExternalLink instance.
- Attribute registryObject – references the parent RegistryObjectType instance within which the ExternalLinkType instance is composed. The value shall be provided by client when an ExternalLink is submitted separate from its parent object. The value shall be set by the server if the ExternalLink is submitted as part of the submission of its parent object.

4.3 Association information model

4.3.1 Overview

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

In Figure 3 below, an AssociationType instance with type “urn:oasis:names:tc:ebxml-regrep:AssociationType:Supersedes” is used to indicate that the NAICS2001 ClassificationScheme supersedes the NAICS1997 ClassificationScheme.

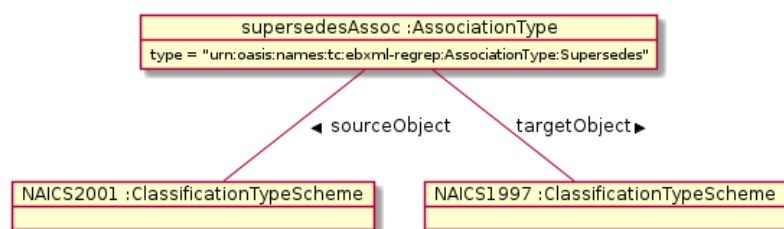


Figure 3 association example

4.3.2 Source and target objects

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

4.3.3 Type of an association

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

4.3.4 AssociationType

Base Type: RegistryObjectType

4.3.4.1 XML schema type definition

```
<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>
```

EXAMPLE: an Organization instance that has an “OffersService” association with a Service that 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" .../>
```

4.3.4.2 Description

Table 15 shows the description of AssociationType.

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

Table 15 AssociationType

- a) Attribute sourceObject - Each Association shall have a *sourceObject* attribute that references the RegistryObjectType instance that is the source of that Association.

- b) Attribute *targetObject* - Each Association shall have a *targetObject* attribute that references the RegistryObjectType instance that is the target of that Association.
- c) Attribute *type* - Each Association shall have a *type* attribute that identifies the type of that association.
 - 1) The value of the *type* attribute shall be a reference to a ClassificationNode within the canonical AssociationType ClassificationScheme.
 - 2) A server shall support the canonical association types as defined by the canonical AssociationType ClassificationScheme. Deployments and profiles may extend the canonical AssociationType ClassificationScheme by adding additional ClassificationNodes to it.

4.3.5 Access control

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

4.4 Classification information model

4.4.1 Overview

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

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

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

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

Figure 4 Classification example shows how a RegistryObject may be classified by multiple ClassificationNodeType instances under multiple ClassificationScheme instances (e.g., IndustryScheme, GeographyScheme).

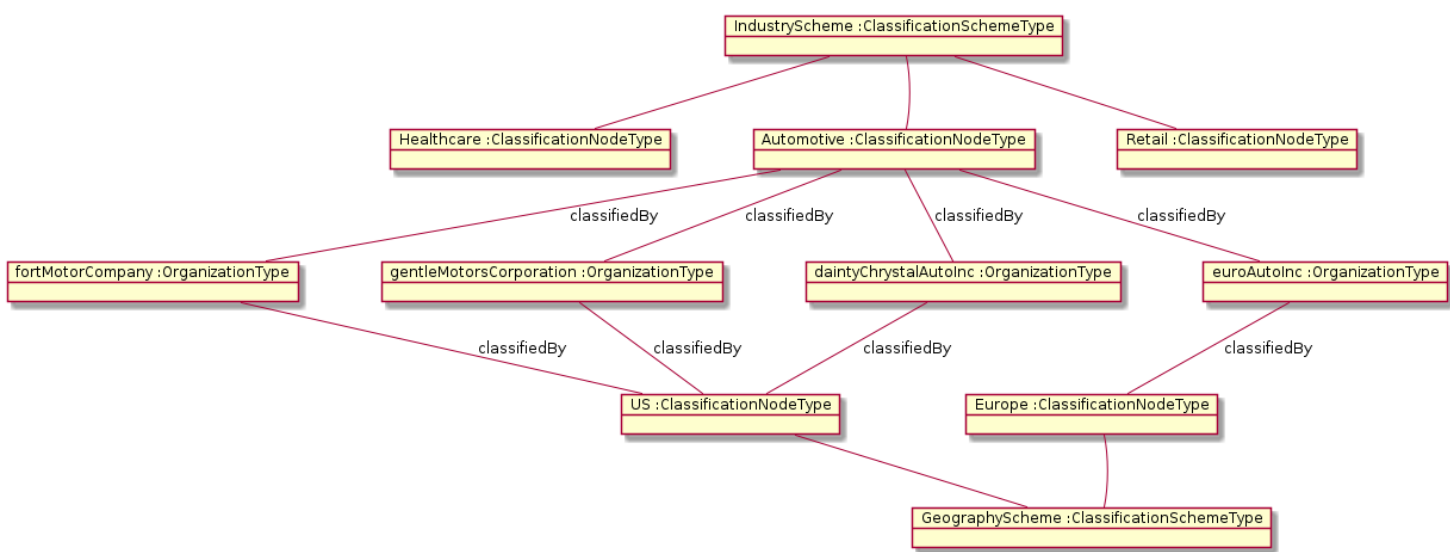


Figure 4 Classification example

Figure 5 shows the classification information model.

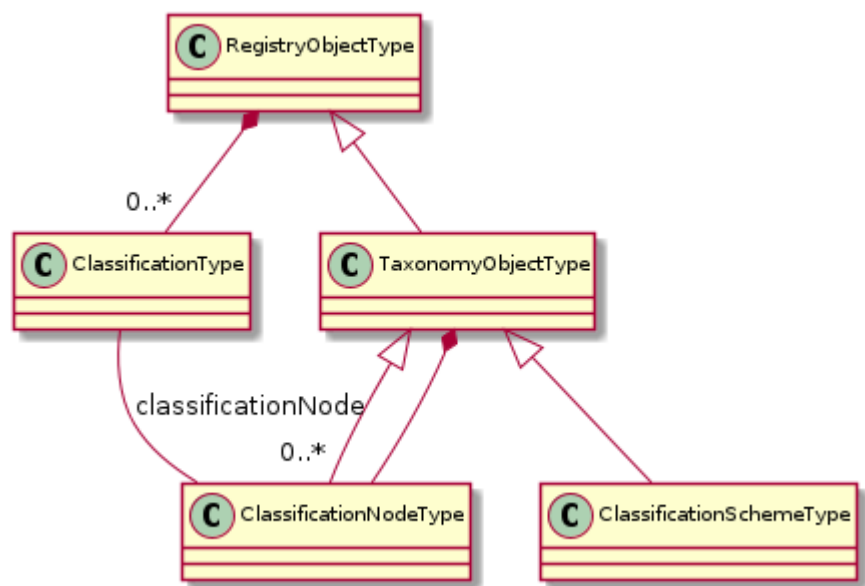


Figure 5 Classification information model

4.4.2 TaxonomyElementType

4.4.2.1 Overview

Base Type: RegistryObjectType

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

4.4.2.2 XML schema type definition

```
<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>
```

4.4.2.3 Description

Table 16 `TaxonomyElementType` shows the description of `TaxonomyElementType`.

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

Table 16 `TaxonomyElementType`

Element `ClassificationNode` – This element represents a `ClassificationNode` child of a parent `TaxonomyElementType` instance. A `TaxonomyElementType` instance may have any number of `ClassificationNode` child elements.

4.4.3 ClassificationSchemeType

4.4.3.1 Overview

Base Type: `TaxonomyElementType`

A `ClassificationScheme` instance represents a taxonomy.

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

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

4.4.3.2 XML schema type definition

```
<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>
```

EXAMPLE: 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>

```

4.4.3.3 Description

Table 17 ClassificationSchemeType shows the description of ClassificationSchemeType.

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

Table 17 ClassificationSchemeType

- a) Attribute isInternal - When submitting a ClassificationSchemeType instance the client shall declare whether the ClassificationSchemeType instance represents an internal or an external taxonomy. This allows the server to validate the subsequent submissions of ClassificationNodeType and ClassificationType instances in order to maintain the type of ClassificationScheme consistent throughout its lifecycle.
- b) Attribute nodeType - When submitting a ClassificationScheme instance the client shall declare the structure of taxonomy nodes within the ClassificationScheme via the nodeType attribute. The value of the nodeType attribute shall be a reference to a ClassificationNodeType instance within the canonical NodeType ClassificationScheme. A server shall support the node types as defined by the canonical NodeType ClassificationScheme. The canonical NodeType ClassificationScheme may easily be extended by adding additional ClassificationNodes to it.

Table 18 NodeType classification scheme lists the canonical ClassificationNode defined as values for the NodeType ClassificationScheme:

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.

Table 18 NodeType classification scheme

4.4.4 ClassificationNodeType

4.4.4.1 Overview

Base Type: TaxonomyElementType

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

4.4.4.2 XML schema type definition

```
<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>
```

4.4.4.3 Description

Table 19 ClassificationNodeType shows the description of ClassificationNodeType.

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

Table 19 ClassificationNodeType

- a) Attribute code - A ClassificationNodeType instance shall have a *code* attribute. The code attribute contains a code that represents a value within a ClassificationScheme.

The code attribute of a ClassificationNodeType instance shall be unique with respect to all sibling ClassificationNodes that are immediate children of the same parent TaxonomyElementType instance.
- b) Attribute parent - A ClassificationNodeType instance may have a *parent* attribute. The parent attribute references the parent TaxonomyElementType instance. This is either another ClassificationNodeType instance or the ClassificationSchemeType instance.
- c) Attribute path - A ClassificationNodeType instance may have a *path* attribute. The path attribute represents a hierarchical path from the root ClassificationSchemeType to the ClassificationNodeType instance. The syntax of the path attribute value is defined in 4.4.4.4.
 - 1) A server shall set the path attribute for any ClassificationNodeType instance when it is submitted by a client.
 - 2) The path attribute shall be ignored by the server if it is specified by the client during the submission of the ClassificationNodeType instance.
 - 3) The path attribute of a ClassificationNode shall be unique within a server.

4.4.4.4 Canonical Path Syntax

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

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

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

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

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

EXAMPLE:

/urn:acme:GenderScheme/Male

4.4.5 ClassificationType

4.4.5.1 Overview

Base Type: RegistryObjectType

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

4.4.5.2 XML schema type definition

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

EXAMPLE: a Person instance is classified using the sample Gender ClassificationScheme used in earlier examples.

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

4.4.5.3 Description

Table 20 ClassificationType shows the description of ClassificationType.

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

Table 20 ClassificationType

- Attribute *classificationNode* - If the *ClassificationType* instance represents an internal classification, then the *classificationNode* attribute is required.
The *classificationNode* value shall reference a *ClassificationNodeType* instance.
- Attribute *classifiedObject* - For both internal and external classifications, the *classifiedObject* attribute is required and it references the *RegistryObjectType* instance that is classified by this *Classification*.
- Attribute *classificationScheme* - If the *ClassificationType* instance represents an external classification, then the *classificationScheme* attribute is required.
The *classificationScheme* value shall reference a *ClassificationScheme* instance.
- Attribute *nodeRepresentation* - If the *ClassificationType* instance represents an external classification, then the *nodeRepresentation* attribute is required. It is a representation of a taxonomy value from a classification scheme.
- A canonical slot with name “urn:oasis:names:tc:ebxml-regrep:rim:Classification:context” may be optionally specified to provide additional context for a *ClassificationType* instance

4.5 Provenance information model

4.5.1 Overview

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

The ebRIM information model supports types and relationships that may be used to represent the

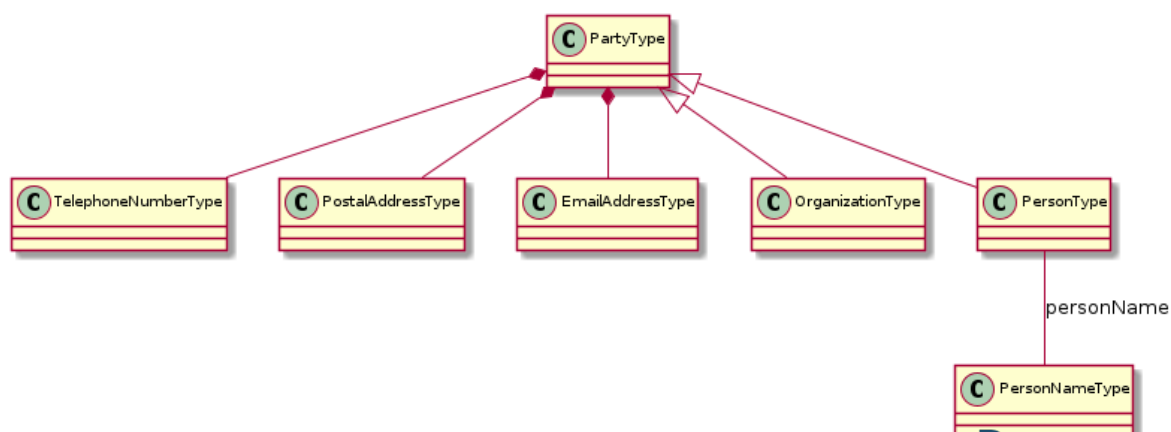


Figure 6 provenance information model

provenance of RegistryObjectType instances. Figure 6 presents the significant types defined by the provenance information model.

4.5.2 PostalAddressType

4.5.2.1 Overview

Base Type: ExtensibleObjectType

This type represents a postal or mailing address.

4.5.2.2 XML schema type definition

```
<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"/>
      <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>
```

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>
```

4.5.2.3 Description

Table 21 PostalAddressType shows the description of PostalAddressType.

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

Table 21 PostalAddressType

- Attribute city - A PostalAddressType instance may have a *city* attribute identifying the city for that address.
- Attribute country - A PostalAddressType instance may have a *country* attribute identifying the country for that address.

- c) Attribute `postalCode` - A `PostalAddressType` instance may have a *postalCode* attribute identifying the postal code (e.g., zip code) for that address.
- d) Attribute `stateOrProvince` - A `PostalAddressType` instance may have a *stateOrProvince* attribute identifying the state, province or region for that address.
- e) Attribute `street` - A `PostalAddressType` instance may have a *street* attribute identifying the street name for that address.
- f) Attribute `streetNumber` - A `PostalAddressType` instance may have a *streetNumber* attribute identifying the street number (e.g., 65) for the street address.

4.5.3 TelephoneNumberType

4.5.3.1 Overview

Base Type: `ExtensibleObjectType`

This type defines attributes of a telephone number.

4.5.3.2 XML schema type definition

```
<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>
```

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>
```

4.5.3.3 Description

Table 22 `TelephoneNumberType` shows the description of `TelephoneNumberType`.

Node	Type	Cardinality	Default Value	Specified By	Mutable
<code>areaCode</code>	String8	0..1		Client	
<code>countryCode</code>	String8	0..1		Client	
<code>extension</code>	String8	0..1		Client	
<code>number</code>	String16	0..1		Client	
<code>type</code>	objectRef	0..1		Client	

Table 22 `TelephoneNumberType`

- a) Attribute *areaCode* - A *TelephoneNumberType* instance may have an *areaCode* attribute that provides the area code for that telephone number.
- b) Attribute *countryCode* - A *TelephoneNumberType* instance may have a *countryCode* attribute that provides the country code for that telephone number.
- c) Attribute *extension* - A *TelephoneNumberType* instance may have an *extension* attribute that provides the extension number, if any, for that telephone number.
- d) Attribute *number* - A *TelephoneNumberType* instance may have a *number* attribute that provides the local number (without area code, country code and extension) for that telephone number.
- e) Attribute *type* - A *TelephoneNumberType* instance may have a *type* attribute that provides the type for the *TelephoneNumber*. The value of the *phoneType* attribute shall be a reference to a *ClassificationNode* in the canonical *PhoneType ClassificationScheme*.

4.5.4 EmailAddressType

4.5.4.1 Overview

Base Type: *ExtensibleObjectType*

This type defines attributes of an email address.

4.5.4.2 XML schema type definition

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

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>
```

4.5.4.3 Description

Table 23 *EmailAddressType* shows the description of *EmailAddressType*.

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

Table 23 *EmailAddressType*

- a) Attribute *address* - An *EmailAddressType* instance shall have an *address* attribute that provides the actual email address.

- b) Attribute type - An EmailAddressType instance may have a *type* attribute that provides the type for that email address. The value of the type attribute shall be a reference to a ClassificationNode in the canonical EmailType ClassificationScheme.

4.5.5 PartyType

4.5.5.1 Overview

Base Type: RegistryObjectType

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

4.5.5.2 XML schema type definition

```
<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>
```

4.5.5.3 Description

Table 24 shows the description of PartyType.

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

Table 24 PartyType

- Element EmailAddress - A PartyType instance may have any number of EmailAddress sub-elements. Each EmailAddress provides an email address for that PartyType instance. A PartyType instance should have at least one EmailAddress.
- Element PostalAddress - A PartyType instance may have any number of PostalAddress sub-elements. Each PostalAddress element provides a postal address for that PartyType instance. A PartyType instance should have at least one PostalAddress.
- Element TelephoneNumber - A PartyType instance may have any number of TelephoneNumber sub-elements. Each TelephoneNumber element provides a TelephoneNumber for that PartyType instance. A PartyType instance should have at least one TelephoneNumber.

4.5.6 PersonType

4.5.6.1 Overview

Base Type: PartyType

This type represents a person.

4.5.6.2 XML schema type definition

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

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>
```

4.5.6.3 Description

Table 25 PersonType shows the description of PersonType.

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

Table 25 PersonType

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

4.5.7 PersonNameType

4.5.7.1 Overview

Base Type: ExtensibleObjectType

This represents the name for a PersonType instance.

4.5.7.2 XML schema type definition

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

```

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

```

EXAMPLE:

```

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

```

4.5.7.3 Description

Table 26 shows the description of PersonNameType.

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

Table 26 PersonNameType

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

4.5.8 OrganizationType**4.5.8.1 Overview**

Base Type: PartyType

This type represents an organization or entity.

4.5.8.2 XML schema type definition

```

<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>

```

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>

```

4.5.8.3 Description

Table 27 shows the description of ObjectRefType.

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

Table 27 OrganizationType

- a) Element Organization – This element allows clients to specify sub-organizations of the Organization instance using a simpler alternative to specifying “HasMember” AssociationType instances between the parent and child Organizations.

A server shall replace any nested Organization elements within an OrganizationType instance with AssociationType instances such that each nested Organization element is replaced with an AssociationType instance with type “urn:oasis:names:tc:ebxml-regrep:AssociationType:HasMember”, with sourceObject specifying the id of the parent OrganizationType instance and with targetObject specifying the id of the nested Organization element

- b) Attribute primaryContact - An OrganizationType instance should have a *primaryContact* attribute that references the Person instance for the person that is the primary contact for that organization.

4.5.9 Associating organization with persons

There are many situations where a person is related to an organization. Such relationship should be defined by AssociationType instances between an OrganizationType instance and a PersonType instance as follows:

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

4.5.10 Associating organization with organizations

There are many situations where an organization is related to another organization. Such relationship should be defined by AssociationType instances between an OrganizationType instance and another OrganizationType instance.

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