Deployment Profile Template

For ebXML Registry 3.0 OASIS Specifications

Version 0.2.4

Draft OASIS Profile - February 2006

Document identifier:

ebRR-3.0-DeploymentProfileTemplate-wd-024

Location:

http://www.oasis-open.org/committees/regrep/documents/...

Editors:

Name	Affiliation
Ivan Bedini	France Telecom
Farrukh Najmi	Sun Microsystems
Nikola Stojanovic	GS1 US

Contributors:

Name	Affiliation
Diego Ballve	Individual

Abstract:

This document provides a template and guidelines on how to effectively create and define an ebXML Registry Profile that is specialized for specific domains and applications. It serves as a template for authoring such new profiles of ebXML Registry.

Copyright © OASIS Open 2005. All Rights Reserved.

Status:

This document is an OASIS ebXML Registry Technical Committee Working Draft Profile.

Committee members should send comments on this specification to the regrep@lists.oasis-open.org list. Others should subscribe to and send comments to the regrep-comment@lists.oasis-open.org list. To subscribe, send an email message to regrep-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

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

Copyright © OASIS Open 2005. All Rights Reserved.

Table of Contents

Copyright © OASIS Open 2005. All Rights Reserved.

Annexe Index

Copyright © OASIS Open 2005. All Rights Reserved.

Appendix Index

Copyright © OASIS Open 2005. All Rights Reserved.

Illustration Index

Copyright © OASIS Open 2005. All Rights Reserved.

Index of Tables

Copyright © OASIS Open 2005. All Rights Reserved.

1 Introduction

1.1 Overview

The ebXML Registry Repository is a part of the ISO 15000 standard and it provides an important piece in the architectures of applications. Not only it can supply a database system, in certain cases, but also it offers several added services for maintaining and managing stored data information.

The specified ebXML Registry Repository Information Model (ebRIM) is general enough to be capable to store any type of content. For this reason implementers of registry applications are often called to define their specific stored data semantic and structure in order to be able to better profiling their own registry usage.

This interesting registry features can some times lose its better purpose when different world wide applications wish to be easily connected between them. Therefore, in developing practical and effective interoperable solutions, the description and the definition of selected real world use cases can be relied on interoperability profiles.

The interest of developing an ebXML Registry Repository interoperability profile for a specific domain can be find between the following reasons:

- to provide a profile for a registry application;
- to guarantee interoperability between world wide registry applications;
- to share and benefit of world wide experiences;
- to guide and facilitate implementers to develop registry applications;
- ...to submit that to the OASIS ebXML Registry Repository Technical Committee for a formal approbation.

1.2 Purposes

The definition of an interoperability profile can be obtained with a customization of [ebRIM] and [ebRS] specifications for a given application domain. This means that the base specifications can be restricted or extended in such a manner that the profile does not contradict any of them (e.g., violate a mandatory constraint).

The main purpose of this document is to provide methods to customize the registry specifications. Therefore throughout this document is analysed what and how registry information model and registry services can be adapted to specifics needs.

Also the deployment profile template defines the necessary guidelines to customize ebXML Registry 3.0 specifications in order to provide a consistent and standardized registry package for the profiled domain.

Profile's editors also should follow the structure of this document in order to build consistent and standardised profiles.

It is not the purpose of this document to educate the reader on ebXML Registry [ebRIM], [ebRS], information modelling. The reader of this document should have a good understanding of the ebXML Registry specifications.

1.3 Audience

The target audience for the deployment profile template includes software developers to enable them to put into place a registry implementation.

The audience includes analysts and architects that are called to develop an interoperability

Copyright © OASIS Open 2005. All Rights Reserved.

profile for a specific domain and wish the support services provided by the Registry.

The audience also includes all users that are, in a certain manner, interested to acquire familiarity with ebXML Registry Repository features.

1.4 Terminology

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

- **Deployment Profile Template**: Document that lists the concepts in the source specification that may be adopted by a user community, that identifies content elements (e.g. ebRIM object types) the format and/or value of which may be further standardized by a community, and that also identifies specifics objects and taxonomies under which the source specification should be used, and selected by a user community for a specific domain.
- **Deployment Profile (or Deployment Guide)**: Document that is an instance of the Deployment Profile Template. It defines which concept should / should not be used by a community for a specific domain, which format or value some content elements should comply with.

1.5 Conventions

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

- Identifier Placeholders

Listings may contain values that reference ebXML Registry objects by their id attribute. These id values uniquely identify the objects within the ebXML Registry. For convenience and better readability, these key values are replaced by meaningful textual variables to represent such id values.
For example, the following placeholder refers to the unique id defined for the canonical ClassificationNode that defines the Organization ObjectType defined in [ebRIM]:

```
<id="${CANONICAL_OBJECT_TYPE_ID _ORGANIZATION}" >
```

- Constants

Constant values are printed in the `Courier New` font always, regardless of whether they are defined by this document or a referenced document. In addition, constant values defined by this document are printed using `bold face`. The following example shows the canonical id and lid for the canonical ObjectType ClassificationScheme defined by [ebRIM]:

```
<rim:ClassificationScheme
       lid="urn:oasis:names:tc:ebxml-regrep:classificationScheme:ObjectType"
       id="urn:uuid:3188a449-18ac-41fb-be9f-99a1adca02cb">
```

- **Example Values**

These values are represented in *italic* font. In the following, an example of a RegistryObject's name "*ACME Inc.*" is shown:

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


Copyright © OASIS Open 2005. All Rights Reserved.

1.6 How to use the Deployment Profile Template

There are several parts in the Deployment Profile Template that need to be instantiated in order to generate a Deployment Profile:

- The binding, or mapping, from the source model to the [ebRIM]. It defines how objects are stored within the registry.
- The definition of the Content Management Service. Here is defined how to create new content validation and cataloguing registry services and what services can be created for the specific domain.
- The definition of the discovery profile. Here the editor of the profile is called to the hard task to developing at least the most common queries for the domain. This task will improve the ability of implementers to provide a minimal set of standard queries that compose the specific domain library for all implementations complaints to the profile.
- The definition of the Event Notification feature. In this section editors are called to define all common subscriptions of interest for the specific domain with the notification action that the registry must perform after an event. More than this the extension, or restriction, of the pre-defined [ebRIM] *auditable event types* list must be provided.
- The definition of the Security profile. In this section are defined the roles, the groups and the rules needed to managing the users profile and access control policies for registry objects.
- Editors are highly encouraged to provide the XML instances for all extensions, restrictions, queries, subscriptions, actions and notifications applied in the profile. XML instances must be compliant to the submit, or remove, object protocol as specified into [ebRS]. XML definitions could be added as annexes to the document or as attached documents to the profile.

The definition of the profile can impact, modify, the following canonical lists of the standard ebXML Registry specification:

- **object types;**
- **association types;**
- **registry object status;**
- **event types;**
- **roles;**
- **groups and;**
- **classification schemes.**

The profile with all attached documents or appendixes can be considered as the registry package for the specified domain.

Copyright © OASIS Open 2005. All Rights Reserved.

2 Overview

This chapter MUST provide an overview of the specific domain (hereafter referred to as the "source model" too).

The source model can be expressed in any preferred form, such as a UML class diagram, XML Schema or in natural language.

The editor of a specific profile can insert here any information that she/he considers useful for implementers and readers of the profile for a good interpretation and understanding of what exactly is profiled throughout this document.

A guide of how a specific UML diagram can be mapped to [ebRIM] is shown in [ebRR-UML-TUT].

(A fictitious PIM is the source information model used as example throughout this document).

2.1 Overview of the Source Model

Throughout this document we use a fictitious domain specific information model called Person Information Model (PIM) as an example for constructing the profile. The PIM represents the source model and [ebRIM] is the target model for the mapping.

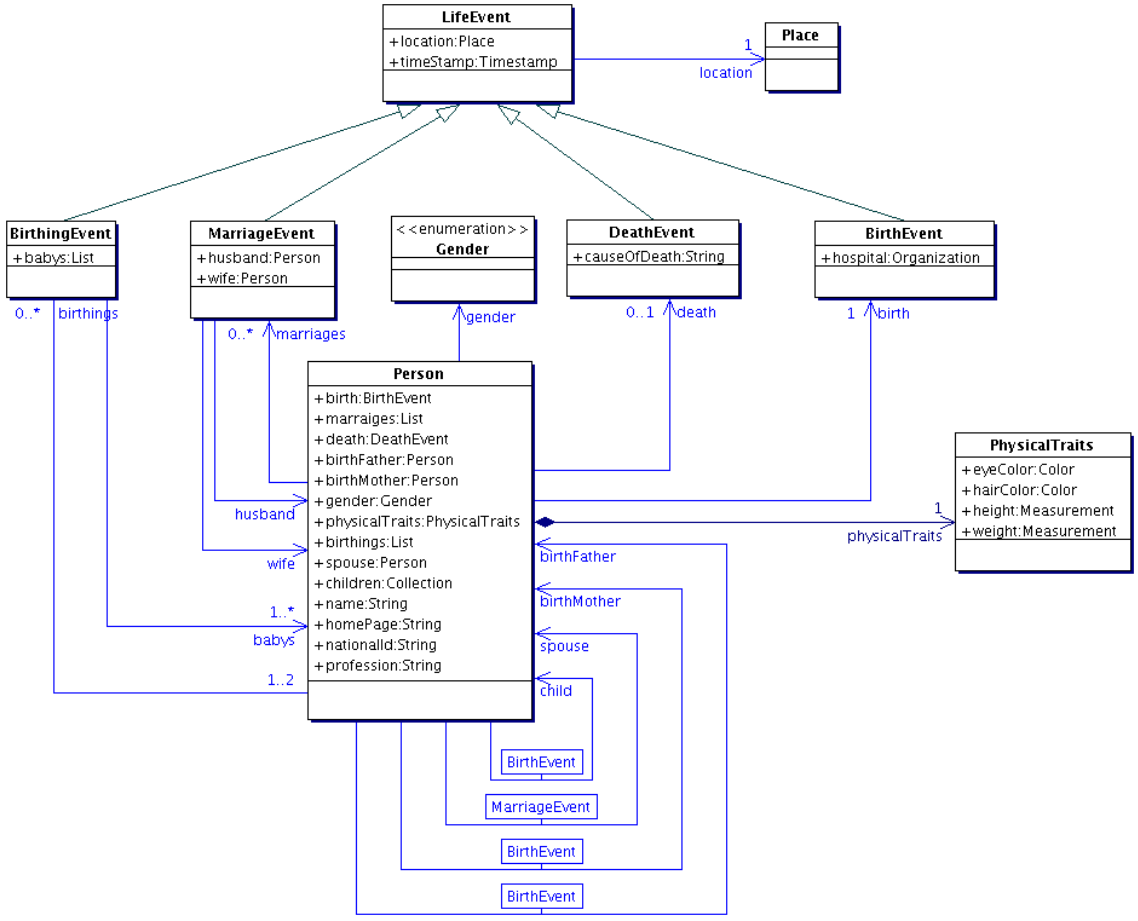


Figure 1: Person Information Model: A Sample Domain Specific Model

shows the UML Class diagram for the Person Information Model. The model shows that:

1. *A Person has several LifeEvents:*

Copyright © OASIS Open 2005. All Rights Reserved.

- *BirthEvent: Marks the birth of the associated Person*
- *MarriageEvent: Marks a marriage of the associated Person*
- *BirthingEvent: Marks a delivery of one or more babies where the associated person is a parent.*
- *DeathEvent: Marks the death of the associated Person*

2. *A Person has a PhysycalTraits which is a collection of various physical traits that describe the Person.*
3. *A Person has a birth mother and birth father which are also Person*
4. *A Person has chidlren which are also Person*
5. *Each class MAY define various attributes as shown within the box for each class.*

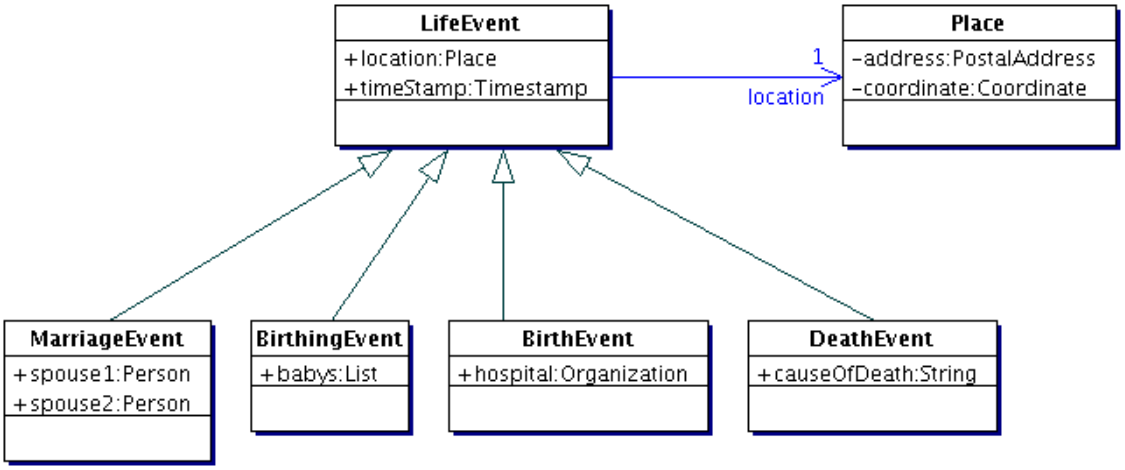


Figure 2: Source Information Model: Inheritance View

above shows another class diagram for the model that shows the inheritance view of the model. Here we see that the various Event classes inherit from the same LifeEvent base class and further specialize it for that specific event.

Copyright © OASIS Open 2005. All Rights Reserved.

3 Mapping the Source Model to [ebRIM]

This section reviews all the issues that should be considered when producing a specialized [ebRIM] profile for a particular domain, from the ebRIM point of view. [ebRIM] already defines several canonical objects for associations, classifications, object types for extrinsicObjects, event types, etc. In a specific application domain the list of these canonical objects needs to be specialized in order to better meet the characteristics of the considered domain.

Here users have to define the mapping of the source domain information model to the Registry Information Model and also define the extensions and/or restrictions needed by the source information model.

This task typically identifies the need for new object types and definitions that extend or restrict the [ebRIM] canonical *ClassificationScheme* (as defined at § 1,6 in [ebRIM]).

For example the *Person* class of PIM source model can be mapped to the canonical [ebRIM] *Person* registry object and the *spouse* association between Person instances, could be mapped to the canonical *AccessControlPolicyFor* association type, but effectively a new association type called simply *Spouse,* in this case, could be preferred.

Furthermore, the mapping operation results in a harmonized way to store domain specific objects and concepts in the ebXML Registry. This is important for improving interoperability issues between cooperating registries and between registries implementing this profile and client applications.

All registry implementations conforming to this profile MUST respect the defined mapping and, if any, create the extended canonical lists.

The ebRIM profiling operation MAY generate a list of RIM *ClassificationScheme* or ClassificationNode that extends the canonical [ebRIM] *ClassificationScheme* for the following RIM modules:

- ***Core***: This module covers the most commonly used information model classes defined by [ebRIM].
- **Association:** This information model defines the registry objects association types.
- **Classification:** This information model describes supports Classification of RegistryObject.
- **Event:** The Event information model enable the registry application to support the registry Event Notification feature.
- **Access Control:** Access Control Information Model is used by the registry to control access to RegistryObjects and RepositoryItems managed by it.

The annexes to this document provide the profile extensions of the canonical [ebRIM] lists (as XML file format) instances that editors of a profile SHOULD includes within this document or as attachments.

In order to maintain conformity between developed profiles, editors are encouraged to follow the structure of this chapter and the presentation of information, anyway they MAY be adapted for better reflect the source domain aspects.

3.1 Objects definition

[ebRIM] provides several canonical object types that are used by registry for its management purposes (such as *AdhocQuery, Notification, Federation, ...),* and often they don't correspond to a specific need. For that [ebRIM] gives the possibility to extend the object type classification adding the new specific objects as sub-node of the predefined *ExtrinsicObject* object type.

Copyright © OASIS Open 2005. All Rights Reserved.

3.1.1 Object Types

Here users define the new *objectTypes* needed by the source model and the correspondents mapping.

For example in the PIM source model the *LifeEvent* object can be mapped to a new [ebRIM] object type called *LifeEvent*, defined as sub-node of *ExtrinsicObject*.

Table 1 presents these information:

- *Source Concept*. Instances of this column represent a concept of the source model that must be mapped to a registry object type. This is mandatory.
- *ebRIM Parent Object Name*. It represents the link to the parent classification node as defined into [ebRIM] for the *ClassificationNode parent* attribute.It is optional If it is already provided into the attached submit object request list for object type.
- *ebRIM Object Name*. It represents the name of the registry object type (classification node) used for defining instances of the corresponding source model concept. It is mandatory.
- *ebRIM code*. It represents the code attribute for the classification node as defined into [ebRIM] for the *ClassificationNode code* attribute. It is optional If it is already provided into the attached submit object request list for object type.
- e*bRIM ObjectType ID.* It is the ebXML registry object ID for this profile. It is optional If it is already provided into the attached submit object request list for object type. The definition of the IDs for the specifics object types is useful for building standard registry ad hoc queries.
- *Comment.* Any additional comment. It is optional.

Editors MUST define here (or alternatively in the attached object types submit object request XML instance) the IDs, ebRIM Parent Object Name and ebRIM code for the new object types.

The table below defines the mapping for the source concepts to [ebRIM].

Source Concept	ebRIM Object Type Name	ebRIM Parent Type Name	ebRIM code	ebRIM ObjectType ID	Comment
PIM	*PIM*	*ExtrinsicObject*	*PIM*	`urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM`	*This instance of ClassificationNode, sub-node of the ExtrinsicObject object type, groups all object type for the domain.*
Person	*Person*	*RegistryObject*	*Person*	`urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:Person`	*This is the canonical [ebRIM] Person object type*
LifeEvent	*LifeEvent*	*PIM*	*LifeEvent*	`urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent`	*New ClassificationNode object type, sub-node of PIM, represents the LifeEvent PIM class*
BirthEvent	*BirthEvent*	*LifeEvent*	*BirthEvent*	`urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent:BirthEvent`	
...	*...*	*...*	*...*	*...*	*...*

Table 1: Source model mapping to [ebRIM]

3.1.2 Attributes binding

Here users have to specify all attributes correspondence between the source model concept defined in the previous paragraph and the [ebRIM] registry object.

Copyright © OASIS Open 2005. All Rights Reserved.

A content validation registry service MUST verify that registry object instances of each previously defined registry object type MUST respect the attribute mapping with the corresponding cardinality. (for ex.: a PIM BirthEvent instance can't have a *slot* referring to the *profession* attribute, etc.).

Where possible attributes MUST be directly mapped to the already defined [ebRIM] attributes. For all other cases a specific *Slot* is defined.

Table 2 below has the following information for each source concept (represented by the light blue line) defined previously:

- *Source Attribute*. The source attribute item represents the attribute of the source concept that Must be mapped to an already existing [ebRIM] registry object attribute or to a new slot instances that Must be defined here. It is mandatory.
- *[ebRIM] Attribute*. The definition of the mapped source attribute. It is mandatory. Slots attributes are defined as follow: Slot(*name, type,*'*value*').
  - *Name* is the corresponding name for the slot (ex: *urn:pim:Person:profession);*
  - *Type* is one of the admitted [ebRIM] types for a *slotType* as defined in [ebRIM];
  - *Value* can be a list of *admitted values* for this attribute that SHOULD be verified by the registry content validation service at the submission or '*any value*' if no constraints exists for the attribute. (for example an attribute can admit only 'Yes' or 'No' values)
- *Cardinality*. It represents the number of distinct values that this attribute can contain. A particular exception is for attributes with cardinality equal to 1 as max value of instances that are mapped to registry object attributes that provide a multilingual value, such as name and description for example, only one value for each language MUST be accepted.
- *Comment*. Optionally, any other additional comment.

The table below lists the attributes for the source model concepts.

Source Attribute	[ebRIM] Attribute	Cardinality	Comment
Source concept: ***Person***			
name	*name*	*1...1*	
homePage	*externalLink*	*0...1*	
nationalId	*ExternalIdentifier*	*0...1*	*“NationalIdentifierScheme” ClassificationScheme as identificationSchema*
profession	*Slot(urn:pim:Person:profession, String, 'any value')*	*0...**	
gender	*Classification*	*1...1*	*Referring to “Gender” ClassififcationScheme*
...			

Table 2: Attributes binding definition

3.2 Association binding

Each registry Association MUST have an *associationType* attribute that identifies the type of that association. The value of this attribute MUST be the id of a *ClassificationNode* under the canonical [ebRIM] *AssociationType* classification scheme. This list can be extended or restricted by users for specifics application domain purposes.

Copyright © OASIS Open 2005. All Rights Reserved.

Here users have to define the list of non canonical association types that MUST be added to the registry implementation and also the profile mapping of the association between the source model and the RIM.

Table 3 lists all new association types for the source model domain. This table is not mandatory if a detailed list is already provided as annexe or attached file for association types. The column meaning is:

- *AssociationType*. Items of this column represent the association concept from the source model that MUST be defined for the profile. It is mandatory.
- *ID*. it represents the registry unique identifier for the new association type. It is mandatory and it MUST be provided here or in the attached file defining the association type submitting object list.
- *Description*. This is not just the comment. Items of this column MAY be added as the [ebRIM] description attribute for the corresponding registry association classification node. It is optional.

AssociationType	ID	Description
Birthing	*urn:oasis:names:tc:ebxml-regrep:classificationScheme:AssociationType:Birthing*	
...	...	...

Table 3: List of non canonical [ebRIM] AssociationType for the profile

Table 4 defines the binding between the associations concepts from the source model and [ebRIM] associations. In this table only *Association Source Object Type, Association Target Object Type* and *the [ebRIM] Association Type* are mandatory.

Association Source Object Type	Association Target Object Type	[ebRIM] Association Type	[ebRIM] Association Name	Comment
Person	*BirthingEvent*	*Birthing*		
...	...	...	...	...

Table 4: List of non canonical AssociationType for the profile

The registry considers associations instances as normal registry objects. For this reason if an association concepts from the source model uses or needs attributes, they can be simply added as slot or directly to an existing registry object attribute to the registry associations instances as defined in §3.1.2.

3.3 Registry Classification system profile

[ebRIM] provides an excellent way to classify stored objects instances into the registry. It is easily extensible simply by adding one or more *ClassificationNodes* to an existing *ClassificationScheme* or by creating a new *ClassificationScheme*.

Here editors of the profile SHOULD define all taxonomies needed by the application domain.

The hierarchical structure for taxonomy can easily maintained by adding child elements to the defined *ClassificationScheme*. Registry object instances can be classified according to the defined taxonomy by adding one or more value to the registry object classification "attribute". Of course canonical taxonomies can be extended by adding child elements or restricted.

The table below defines the new classification scheme for the source model concepts.

Information for this table correspond to:

Copyright © OASIS Open 2005. All Rights Reserved.

- *Name*. The name for the new classification scheme or classification node. For new classification nodes editors MUST provide the corresponding classification scheme under whose they are added. It is mandatory.
- *ID*. the unique identifier for the registry object classification node or classification scheme. It is mandatory.
- *Reference*. A classification may be based on an existing list. A reference to the list can be inserted here. It is optional.
- *Comment*. Any comment. It is optional.

Name	ID	Reference	Comment
Gender	*urn:oasis:names:tc:ebxml-regrep:classificationScheme:Gender*		*This Class provides a classification for Person.Gender. All instances of this classification (Male, Female,...) are sub-node elements of Gender.*
NationalIdentifierScheme	*urn:oasis:names:tc:ebxml-regrep:classificationScheme:NationalIdentifierScheme*	*http://www.nationalidentifier.org/list.xml*	*ClassificationScheme used by person:nationalId external identifier attribute.*
...	*...*	*...*	*...*

Table 5: Classification profile for the source model concepts

3.4 Status attribute definition

Each RegistryObject instance has a status indicator. The canonical list of the status attributes is showed in Table 6.

Name	Description
Approved	Status of a *RegistryObject* that catalogues content that has been submitted to the registry and has been subsequently approved.
Deprecated	Status of a *RegistryObject* that catalogues content that has been submitted to the registry and has been subsequently deprecated.
Submitted	Status of a *RegistryObject* that catalogues content that has been submitted to the registry.
Withdrawn	Status of a *RegistryObject* that catalogues content that has been withdrawn from the registry. A repository item has been removed but its *ExtrinsicObject* still exists.

Table 6: Pre-defined choices for the RegistryObject status attribute

This list MAY be extended, or restricted. To extend this list is enough to add new status types to the canonical registry status type classification.

The columns of the table below represent:

- *Name.* The name of the new status type. It is mandatory.
- *Description.* The description of the new status type. It is mandatory.
- *ID*. The unique identifier under which the new status type is created in the registry. It is not mandatory if a detailed list is already provided as annexe or attachment for this

Copyright © OASIS Open 2005. All Rights Reserved.

469 profile.

470

Name	Description	ID
OnWork	*Status of a RegistryObject that...*	*urn:...*

Table 7: Non canonical Status Type list

Copyright © OASIS Open 2005. All Rights Reserved.

4 Publishing Profile

In this section profile's editors MAY add any special rules on how to publish the content for the profile SHOULD be described.

Copyright © OASIS Open 2005. All Rights Reserved.

5 Content Management Service Profile

Authors of new profiles may want to split this chapter into separate chapters one for each content service, depending on the domain issues or profile stuffs.

5.1 Defining Content Validation Services

Here editors of this profile SHOULD add all information needed for automating as much as possible the validation service at the submission of new registry objects.

At least a description of common use cases for the domain MUST be provided.

5.1.1 How to declare new content validation service. Step by Step method.

A registry uses one or more Content Validation Services to automatically validate the RegistryObjects and repository items when they are submitted to the registry.

In this section is provided a step by step method for declaring and use new registry validation services from the implementer point of view. More details on this feature can be found directly on the [ebRS] document.

1. Definition of the object type to associate to the content validation service.
   When new data instances are submitted to the registry, they must have a reference to the object type corresponding to the submitted data. This *objectType* can reference a canonical registry object type or an ad hoc object type explicitly added to the registry. The example below show how a specific object type is defined the PIM registry object instances.
   This specific object type is the object type that is used in this case to associate the defined content validation service (see step 2).
   Any creation of registry object of the associated object type to the service, awake the content validation service.

```
<rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject"
lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM"
id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM" code="PIM" />
```

2. Definition of the registry content validation service
   This step provide the definition of the service that is used to validate the new registry submission for associated registry object types.
   The listing below shows an example of a declaration of a registry content validation service. This service is declared as an *inLine* service with the error handling setted to "*LogErrorandContinue*" and with a reference to a documentation of the service.

```
<rim:Service id="{$service_ID}" objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:Service">
      <rim:Name>
            <rim:LocalizedString value="PIMValidationService"/>
      </rim:Name>
      <rim:Classification classifiedObject="{$service_ID}"
id="urn:oasis:names:tc:ebxml-
regrep:ContentManagementService:ContentValidationService"/>
      <rim:Classification classifiedObject="{$service_ID}"
id="urn:oasis:names:tc:ebxml-regrep:InvocationModel:Inline"/>
      <rim:Classification classifiedObject="{$service_ID}"
id="urn:oasis:names:tc:ebxml-
regrep:ErrorHandlingModel:LogErrorAndContinue"/>
      <rim:ServiceBinding id="{$serviceBinding_ID}"
      service="{$service_ID}"
```


Copyright © OASIS Open 2005. All Rights Reserved.

```
      objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ServiceBinding"
      accessURI="???">
            <rim:SpecificationLink id="{$SpecificationLink_ID}"
            specificationObject="{$ExternalLink_ID}"
            serviceBinding="{$serviceBinding_ID}" />
      </rim:ServiceBinding>
</rim:Service>
```

3. Definition of the optional link to a documentation of the service
   This link provide the access to the documentation of the service, if any.

```
<rim:ExternalLink id="{$ExternalLink_ID}"
externalURI="http://www.mydomainapplication.org/ContentApplicationValidat
ionRules/PIM_xsd.html" />
```

4. Creation of the *ContentManagementServiceFor* association between the created Content Management Validation Service and the object type.
   This association provides the association of the object type PIM to the validation service.

```
<rim:Association id="{$PIM_ContentMgmSrv_Ass_ID}"
associationType="urn:oasis:names:tc:ebxml-
regrep:AssociationType:ContentManagementServiceFor"
sourceObject="{$service_ID}"
targetObject="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM" />
```

5. Definition of the object type for Invocation Control File object instances.
   This step is not mandatory for the creation of the service, it permits only to group all invocation files used for the content management under a common and specific tree. A simple object of type *ExtrinsicObject* can be used for this purpose.

```
<rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM"
lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:InvocationFile"
id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:InvocationFile"
code="InvocationFile" />
```

6. Creation of the Invocation control file.
   This instance of a registry object must have an associated repository item representing the validation file which is invoked by the service when a new PIM registry object is submitted to the registry.

```
<rim:ExtrinsicObject id="{$PIM_InvocationControlFile_ValidationFile_ID}"
objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:InvocationFile">
      <rim:Name>
            <rim:LocalizedString value="PIMValidationXSDFile"/>
      </rim:Name>
      <!--########################################-->
      <!--###                  +                 ###-->
      <!--###      PIM_Validation_file.xsd       ###-->
      <!--###           Repository Item          ###-->
      <!--###                                    ###-->
      <!--########################################-->
</rim:ExtrinsicObject>
```

7. Creation of the *InvocationControlFileFor* association between the submitted Invocation control validating file and the object type
   This step is needed in order to associate the specific registry object type with the invocation file instance used to validate the submission.

```
<rim:Association id="{$PIM_InvocationControlFile_Ass_ID}"
      associationType="urn:oasis:names:tc:ebxml-
regrep:AssociationType:InfocationControlFileFor"
      sourceObject="{$PIM_InvocationControlFile_ValidationFile_ID}"
```


Copyright © OASIS Open 2005. All Rights Reserved.

```
      targetObject="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM" />
```

8. Creation of a new instance of the type defined in step 1 associated to the service. This step shows only the submission of a registry object that will wake up the registry content validation service for the specific object type.

```
<rim:ExtrinsicObject id="{$PIM_FabriceBourge_ID}"
      objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM">
      <rim:Name>
            <rim:LocalizedString value="PIM Fabrice Bourge
Specification"/>
      </rim:Name>
      <!--##############################################-->
      <!--###                      +                   ###-->
      <!--### PIM_FabriceBourge_Specification.xml ###-->
      <!--###           Repository Item            ###-->
      <!--###                                      ###-->
      <!--##############################################-->
</rim:ExtrinsicObject>
```

The method above could seem relatively complex, but a GUI tool could provide a simple way to implement the algorithm that hides the complexity of the method to the final users.

5.2 Defining Content Cataloguing Services

The ebXML Registry provides the ability for a user defined content cataloguing service to be configure for each *ObjectType* defined by the mapping. The purpose of cataloguing service is to selectively convert content into ebRIM compatible metadata when the content is submitted. The generated metadata enables, for example, the selected content to be used as parameter(s) in a domain specific parametrized query.

At least a description of common use cases for the domain content cataloguing service SHOULD be provided here.

5.2.1 How to declare new content cataloguing services. Step by Step method

The registry provides the ability to selectively convert submitted RegistryObject and repository items into metadata defined by [ebRIM], in a content specific manner.

Similarly to the content validation service, new content cataloguing registry services can be created following the step by step method defined below.

1. Definition of the object type used by content cataloguing service.

```
<rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject"
lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM"
id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM" code="PIM" />
```

2. Definition of the service for cataloguing an object submission

```
<rim:Service id="{$service_2_ID}" objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:Service">
      <rim:Name>
            <rim:LocalizedString value="PIMCatalogingService"/>
      </rim:Name>
      <rim:Classification classifiedObject="{$service_2_ID}"
id="urn:oasis:names:tc:ebxml-
regrep:ContentManagementService:ContentCatalogingService"/>
```


Copyright © OASIS Open 2005. All Rights Reserved.

```
      <rim:Classification classifiedObject="{$service_2_ID}"
id="urn:oasis:names:tc:ebxml-regrep:InvocationModel:Inline"/>
      <rim:Classification classifiedObject="{$service_2_ID}"
id="urn:oasis:names:tc:ebxml-
regrep:ErrorHandlingModel:LogErrorAndContinue"/>
      <rim:ServiceBinding id="{$serviceBinding_ID}"
      service="{$service_2_ID}"
      objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ServiceBinding"
      accessURI="???">
      <rim:SpecificationLink id="{$SpecificationLink_ID}"
            specificationObject="{$ExternalLink_ID}"
            serviceBinding="{$serviceBinding_ID}" />
      </rim:ServiceBinding>
</rim:Service>
```

3. Definition of the link to the documentation of the service

```
<rim:ExternalLink id="{$ExternalLink_ID}"
externalURI="http://www.mydomainapplication.org/ContentApplicationCatalog
uingRules/PIM.html" />
```

4. Creation of the ContentManagementServiceFor association between the created Content Management Cataloguing Service and the object type to catalogue.

```
<rim:Association  id="{$PIM_ContentMgmSrv_Ass_2_ID}"
associationType="urn:oasis:names:tc:ebxml-
regrep:AssociationType:ContentManagementServiceFor"
sourceObject="{$service_2_ID}"
targetObject="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM" />
```

5. Definition of the object type for Invocation Control File. This step is not mandatory for the creation of the service, it permits only to group all invocation files used for the content management under a common and specific tree. A simple object of type *ExtrinsicObject* can be used for this purpose.

```
rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM"
lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:InvocationFile"
id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:InvocationFile"
code="InvocationFile" />
```

6. Creation of the Invocation control file used to catalogue object instances

```
<rim:ExtrinsicObject id="{$PIM_InvocationControlFile_CatalogingFile_ID}"
objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:InvocationFile">
      <rim:Name>
            <rim:LocalizedString value="PIMCatalogingXSLTFile"/>
      </rim:Name>
      <!--##########################################-->
      <!--###                  +                 ###-->
      <!--###     PIM_Catalogning_file.xslt      ###-->
      <!--###           Repository Item          ###-->
      <!--###                                    ###-->
      <!--##########################################-->
</rim:ExtrinsicObject>
```

7. Creation of the *InvocationControlFileFor* association between the submitted Invocation control cataloguing file and the object type

```
<rim:Association id="{$PIM_InvocationControlFile_Ass_2_ID}"
```


Copyright © OASIS Open 2005. All Rights Reserved.

```
      associationType="urn:oasis:names:tc:ebxml-
regrep:AssociationType:InfocationControlFileFor"
      sourceObject="{$PIM_InvocationControlFile_CatalogingFile_ID}"
      targetObject="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM" />
```

8. Creation of a new instance of the type defined in step 1 associated to the service.

```
<rim:ExtrinsicObject id="{$PIM_FabriceBourge_ID}"
      objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM">
      <rim:Name>
            <rim:LocalizedString value="PIM Fabrice Bourge
Specification"/>
      </rim:Name>
      <!--##############################################-->
      <!--###                    +                   ###-->
      <!--### PIM_FabriceBourge_Specification.xml ###-->
      <!--###           Repository Item            ###-->
      <!--###                                       ###-->
      <!--##############################################-->
</rim:ExtrinsicObject>
```


Copyright © OASIS Open 2005. All Rights Reserved.

6 Discovery Profile

6.1 Defining Domain Specific Queries

The ebXML Registry provides the ability for domain specific queries to be defined as parametrized stored queries within the Registry as instances of the *AdHocQuery* class. When mapping a domain specific model one SHOULD define such domain specific queries.

The defined set of queries constitute the common registry queries library for the domain that provide a standard discovery library that MUST be added to the complaint registry implementations of this profile.

The first step in defining these domain specific queries is to identify the common use cases for discovering domain specific objects in the registry using natural language. The second step is to specify the *AdHocQueries* with theirs identifiers.

Profile's editors SHALL provide, as attachment or annexe to this document, the whole set of *AdHocQueries* in XML submit registry object list protocol format as defined in [ebRS].

For the profiled domain the identified specific discovery use cases as likely to be commonly needed can be summarized in a table as Table 8 below.

The table has the following information for each source concept (represented by the light blue line) or any other grouping concept for the query:

- *Search by* – items of this column represent the parameter for the queries. It is mandatory;
- *AdHocQuery* – in this column are defined the AdHocQueries in [ebRS] SQL Query syntax or [ebRS] Filter Query syntax. This column can be empty if an attached file or an annex to this profile already provided;
- *Description* – the description of the discovery query.

Search by (Parameters)	**AdHocQuery Expression**	**Description**
Source concept: **Person**		
$person.name *$person.gender*	*<rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxmlregrep:QueryLanguage:SQL-92">* *SELECT DISTINCT person.* FROM* *Person person, Slot personGender* *WHERE* *person.objectType = "urn:oasis:names:tc:ebxmlregrep:ObjectType:RegistryObject:Person"* *AND (personName_firstName LIKE UPPER ("$parent.name")* *AND (Person.id = personGender.parent AND personGender.name_ = "urn:oasis:names:tc:ebxmlregrep:profile:PIM:Gender"* *AND personGender.value LIKE "$person.gender")* *</rim:QueryExpression>* *</rim:AdhocQuery>*	*Parametrised discovery query for PIM Person registry object*
...	...	...

Table 8: Common AdHocQueries definition

6.2 Using stored query

A stored query can be invoked using the *AdHocQueryRequest* protocol as defined in [ebRS].

The following example illustrates how to find all Person instances that have a name containing the string "*Bourge*" and have a gender attribute equal to the string "*Male*". Note that additional supported parameters MAY also be specified if needed.

Copyright © OASIS Open 2005. All Rights Reserved.

```
<AdhocQueryRequest>
  <rim:AdhocQuery
id="urn:oasis:names:tc:ebxmlregrep:profile:pim:query:PersonDiscoveryQuery
">
    <rim:Slot name="$person.name">
      <rim:ValueList>
        <rim:Value>%Bourge%</rim:Value>
      </rim:ValueList>
    </rim:Slot>
    <rim:Slot name="$person.gender">
      <rim:ValueList>
        <rim:Value>Male</rim:Value>
      </rim:ValueList>
    </rim:Slot>
  </rim:AdhocQuery>
</AdhocQueryRequest>
```

Listing 1: Example of stored parametrized AdHocQuery invocation

Copyright © OASIS Open 2005. All Rights Reserved.

7 Event Notification Profile

The ebXML Registry provides the ability for a user or an automated service to create a subscription to events that match a specified criterea. Whenever an event matching the specified criteria occurs, the registry notifies the subscriber that the event transpired.

A mapping of a domain specific model to ebRIM SHOULD define template Subscriptions for the typical use cases for event notification within that domain.

7.1 Event types extension definition

The ebXML Registry provides an event management service for all registry object instances. To benefit of this feature is enough to associate registry instances with an ordered Set of *AuditableEvent* instances. For that users can profile specifics event types to extend the canonical list.

The following table lists pre-defined auditable event types.

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

Table 9: Canonical EventTypes

The table below lists the extended eventTypes.

Name	ID	Description
XXX	*urn:oasis:names:tc:ebxml-regrep:EventType:XXX*	

Table 10: Non canonical EventTypes

7.2 Use Cases for Event Notification

The following are some common use cases that may benefit from the event notification feature:

- A user may be using an object in the registry and may want to know when it changes. For example, they may be using an XML Schema as the schema for their XML documents. When a new version of that XML Schema is created they may wish to be notified so that they can plan the migration of their business processes to the new version of the XML Schema.
- A user may be interested in a certain type of object that does not yet exist in the registry. They may wish to be notified when such an object is published to the

Copyright © OASIS Open 2005. All Rights Reserved.

registry. For example, assume that a registry provides a dating service based upon PIM. Let us A person may create a subscription specifying interest in a female that has never been married before, has brown eyes, is between the age of 30 and 40 and who is a Doctor. Whenever, a Person instance is submitted that matches this criteria, the registry will notify the user.

- An automated service such as a software agent may be interested in certain types of events in the registry. For example, a state coroners office may operate a service that wishes to be notified of deaths where the cause of death was a bullet wound. To receive such notifications, the coroners office may create a subscription for pim.DeathEvents where pim.DeathEvent.causeOfDeath contained the word “bullet”.

7.3 Creating Subscriptions for Events

A user may create a subscription to events of interest by submitting a Subscription object to the registry as defined by ebRIM. The Subscription object MUST specify a selector parameter that identifies a stored query that the registry should use to select events that are of interest to the user for that Subscription.

```
<SubmitObjectsRequest >
  <rim:RegistryObjectList>

    <rim:Subscription id=${DEATH_SUBSCRIPTION_ID}
    selector="${SELECTOR_QUERY_ID}">

      <!-- email address endPoint for receiving notification via email --
>
      <rim:NotifyAction notificationOption="urn:uuid:84005f6d-419e-4138-
a789-fb9fecb88f44" endPoint="mailto:farrukh.najmi@sun.com"/>

      <!—Web Service endPoint for receiving notification via SOAP -->
      <rim:NotifyAction notificationOption="urn:uuid:84005f6d-419e-4138-
a789-fb9fecb88f44" endPoint="urn:uuid:2a13e694-b3ae-4cda-995a-
aee6b2bab3d8"/>
    </rim:Subscription>

    <!-- The query used as a selector for Subscription. -->
    <query:SQLQuery id="${SELECTOR_QUERY_ID}">
      <query:QueryString>SELECT * FROM ExtrinsicObject eo WHERE
eo.objectType =
''${DEATH_EVENT_CLASSIFICATION_NODE_ID}''</query:QueryString>
    </query:SQLQuery>

    <!-- The notification listener web service and its binding -->
    <rim:Service id="${DEATH_EVENT_LISTENER_SERVICE_ID}">
      <rim:Name>
        <rim:LocalizedString value="Listens for Death Events involving
bullet wounds" xml:lang="en-US"/>
      </rim:Name>

  <rim:ServiceBinding service=${DEATH_EVENT_LISTENER_SERVICE_ID}

accessURI="http://localhost:8080/NotificationListener/notificationListene
r"
    id=${DEATH_EVENT_LISTENER_SERVICE_BINDING_ID}>
        <rim:Name>
          <rim:LocalizedString value="Death events listener web service
binding"
              xml:lang="en-US"/>
        </rim:Name>
      </rim:ServiceBinding>
    </rim:Service>
  </rim:RegistryObjectList>
```


Copyright © OASIS Open 2005. All Rights Reserved.

```
</SubmitObjectsRequest>
```

Listing 2: Example of Defining a Subscription for DeathEvent

The above example show how a state coroner's office may create a Subscription to *DeathEvents*.

The following notes describe the example:

- The Subscription is submitted by sending a *SubmitObjectsRequest* to the registry as is the case when publishing any other type of *RegistryObject*.
- The Subscription object is assigned a unique id, lid and optional name and description like any other *RegistryObject*.
- The Subscription specifies the id of its selector query using the selector attribute.
- The *SubmitObjectsRequest* also contains an SQLQuery object that specifies the query used to select *DeathEvents*. The query could be further specialized for example to match only those death events where the cause of death has the word "*bullet*" in it.
- The subscription contains one or more *NotifyActions* describing how the registry should deliver notification of events matching the selector query for this subscription.
- The subscription contains a *NotifyAction* that specifies an email address where the registry should send email based notification of events matching the selector query for this subscription.
- The subscription also contains a *NotifyAction* that specifies the id of a *ServiceBinding*. This is the *ServiceBinding* for the automated listener service where the registry should send SOAP based notification of events matching the selector query for this subscription.
- The selector query and the *Service* / *ServiceBinding* MAY be submitted prior to the submission of the Subscription in a separate request.
- Note that registry implementations [IMPL] may simplify the task of creating and managing subscriptions by providing GUI tools.

Copyright © OASIS Open 2005. All Rights Reserved.

8 Security Profile

The ebXML Registry provides a powerful and extensible access control feature that makes sure that a user may only perform those actions on a RegistryObject or repository item for which they are authorized.

If you are familiar with concept of Access Control Lists (ACLs), you may think of the registry access control feature as a similar though functionally much richer capability.

The registry provides a Role Based Access Control (RBAC) where access to objects may be granted or denied based upon:

- Identity of the user. An example is to grant Sally the privilege of updating the Person instance for Marie Curie.
- Role(s) played by user. An example is to grant anyone with role of Coroner to update a DeathEvent instance.
- Group(s) the user belongs to. An example is to grant anyone who belongs to the group MarieCurieInstitute the privilege of updating the Person instance for Marie Curie.

Here users MAY profile the canonical classification for roles and groups.

8.1 Subject Role Extension

The ebXML Registry defines a set of pre-defined roles in the *SubjectRole* scheme. A domain specific mapping to ebRIM MAY define additional domain specific roles by extending the SubjectRole scheme.

The table below lists all non canonical roles used by the specific domain.

Name	ID	Comment
XXX	*urn:oasis:names:tc:ebxml-regrep:SubjectRole:XXX*	

Table 11: Non canonical roles

8.2 Subject Group Extension

The ebXML Registry defines a set of pre-defined roles in the *SubjectGroup* scheme. A domain specific mapping to ebRIM MAY define additional domain specific groups by extending the SubjectGroup scheme.

The table below lists all non canonical groups used by the specific domain.

Name	ID	Comment
XXX	*urn:oasis:names:tc:ebxml-regrep:classificationScheme:SubjectGroup:XXX*	

Table 12: Non canonical groups

8.3 Profiling Access Control Policies

The ebXML Registry provides a powerful and extensible access control feature that makes sure that a user may only perform those actions on a registry object or repository item for which they are authorized.

Copyright © OASIS Open 2005. All Rights Reserved.

Known Issues

- A better definition for the profile package must be provided.
- A better vision and a solution of how registry objects can be mapped to real concept could be provided. The idea is to provide a way for mange a set of registry objects as an implementation object. For example in the Core Component domain a core component is stored within the registry as a simple extrinsic object instance with several association to other related registry object. For final users a Core Component represents more than the simple extrinsic object, so a “getCoreComponent” query must consider the aspect that the user wish manage the whole Core Component with its Basic core component and association “object.
- Some skeleton of XACML instances could be provided or some rules to build simply a new policy.
- Some content validation and cataloguing rules and examples must be added.
- The Overview of [ebRS] section should be added
- Some guidelines on Cooperating Registries Support must be added.
- Some example throughout the document could be added in order to facilitate the comprehension.

Copyright © OASIS Open 2005. All Rights Reserved.

Tips and Tricks

Here editor's of the profile MAY add any additional information.

Copyright © OASIS Open 2005. All Rights Reserved.

Annexe A - Source model profile Object Type extension

```
<?xml version="1.0" encoding="UTF-8"?>
<SubmitObjectsRequest xmlns="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0"
xmlns:lcm="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd" xmlns:query="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:query21="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:rim="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rim30="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rs="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd">
<RegistryObjectList>
    <!-- ########################################################## -->
    <!-- ### Specifics ObjectType extensions                    ### -->
    <!-- ### Sub-nodes of ExtrinsicObject ClassificationScheme### -->
    <!-- ########################################################## -->
    <ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject" code="PIM"
lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM"
id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM">
      <!-- ObjectType for LifeEvent -->
      <ClassificationNode code="LifeEvent"
lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent"
id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent">
            <!-- ObjectType for BirthEvent -->
            <ClassificationNode code="BirthEvent"
lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent:BirthEvent
" id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent:BirthEvent
"/>
            <!-- ObjectType for MarriageEvent -->
            <ClassificationNode code="MarriageEvent"
lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent:MarriageEv
ent" id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent:MarriageEv
ent"/>
            <!-- ObjectType for BirthingEvent -->
            <ClassificationNode code="BirthingEvent"
lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent:BirthingEv
ent" id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent:BirthingEv
ent"/>
            <!-- ObjectType for DeathEvent -->
            <ClassificationNode code="DeathEvent"
lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent:DeathEvent
" id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent:DeathEvent
"/>
      </ClassificationNode>
            <!-- ObjectType for Place -->
            <ClassificationNode code="Place"
lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:Place"
id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:Place"/>
```


Copyright © OASIS Open 2005. All Rights Reserved.

```
            <!-- ObjectType for PhysicalTraits -->
            <ClassificationNode code="PhysicalTraits"
lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent:PhysicalTr
aits" id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:LifeEvent:PhysicalTr
aits"/>
    </ClassificationNode>
</RegistryObjectList>
</SubmitObjectsRequest>
```

Listing 3: Registry ObjectList profile for the source model

Copyright © OASIS Open 2005. All Rights Reserved.

Annexe B - Source model profile Association Type extension

```
<?xml version="1.0" encoding="UTF-8"?>
<SubmitObjectsRequest xmlns="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0"
xmlns:lcm="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd" xmlns:query="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:query21="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:rim="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rim30="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rs="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd">
<RegistryObjectList>
    <!-- ######################################################## -->
    <!-- ### Specifics AssociationType extensions             ### -->
    <!-- ### Sub-nodes of AssociationType ClassificationScheme### -->
    <!-- ######################################################## -->
    <!-- AssociationType for Birthing -->
    <ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType"
lid="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Birthing" code="Birthing"
id="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Birthing"/>
    <!-- AssociationType for Baby -->
    <ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType"
lid="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Baby" code="Baby"
id="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Baby"/>
    <!-- AssociationType for Spouse -->
    <ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType"
lid="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Spouse" code="Spouse"
id="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Spouse">
      <!-- AssociationType for Husband -->
      <ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Spouse"
lid="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Husband" code="Husband"
id="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Husband"/>
      <!-- AssociationType for Wife -->
      <ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Spouse"
lid="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Wife" code="Wife"
id="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Wife"/>
    </ClassificationNode>
    <!-- AssociationType for Marriage -->
    <ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType"
lid="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Marriage" code="Marriage"
id="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Marriage"/>
    <!-- AssociationType for Death -->
```


Copyright © OASIS Open 2005. All Rights Reserved.

```
    <ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType"
lid="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Death" code="Death"
id="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Death"/>
    <!-- AssociationType for Birth -->
    <ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType"
lid="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Birth" code="Birth"
id="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Birth"/>
    <!-- AssociationType for Child -->
    <ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType"
lid="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Child" code="Child"
id="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Child"/>
    <!-- AssociationType for BirthFather -->
    <ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType"
lid="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:BirthFather"
code="BirthFather" id="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:BirthFather"/>
    <!-- AssociationType for BirthMother -->
    <ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType"
lid="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:BirthMother"
code="BirthMother" id="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:BirthMother"/>
    <!-- AssociationType for Location -->
    <ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType"
lid="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Location" code="Location"
id="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType:Location"/>
</RegistryObjectList>
</SubmitObjectsRequest>
```


Copyright © OASIS Open 2005. All Rights Reserved.

Annexe C - Source model Classification profile extension

```
<?xml version="1.0" encoding="UTF-8"?>
<SubmitObjectsRequest xmlns="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0"
xmlns:lcm="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd" xmlns:query="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:query21="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:rim="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rim30="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rs="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd">
<RegistryObjectList>
    <!-- ########################################################## -->
    <!-- ##### Specifics Classification Scheme extensions     ##### -->
    <!-- ########################################################## -->
    <!-- ClassificationScheme for Gender Taxonomy -->
    <ClassificationScheme lid="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:Gender" id="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:Gender" isInternal="true"
nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode"
objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ClassificationScheme">
      <Name>
            <LocalizedString charset="UTF-8" xml:lang="en-US"
value="Gender"/>
      </Name>
      <Description>
            <LocalizedString charset="UTF-8" xml:lang="en-US"
value="Defines the Gender taxonomy."/>
      </Description>
      <!-- 'Female' taxonomy for Gender -->
      <ClassificationNode lid="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:Gender:Female" code="Female"
id="urn:oasis:names:tc:ebxml-regrep:classificationScheme:Gender:Female"/>
      <!-- 'Male' taxonomy for Gender -->
      <ClassificationNode lid="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:Gender:Male" code="Male"
id="urn:oasis:names:tc:ebxml-regrep:classificationScheme:Gender:Male"/>
      <!-- 'Other' taxonomy for Gender -->
      <ClassificationNode lid="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:Gender:Other" code="Other"
id="urn:oasis:names:tc:ebxml-regrep:classificationScheme:Gender:Other"/>
   </ClassificationScheme>
    <!-- ClassificationScheme for NationalIdentifierScheme Taxonomy -->
    <ClassificationScheme lid="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:NationalIdentifierScheme"
id="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:NationalIdentifierScheme" isInternal="true"
nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode"
objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ClassificationScheme">
      <Name>
            <LocalizedString charset="UTF-8" xml:lang="en-US"
value="NationalIdentifierScheme"/>
      </Name>
      <Description>
            <LocalizedString charset="UTF-8" xml:lang="en-US"
value="Defines the NationalIdentifierScheme taxonomy."/>
      </Description>
   </ClassificationScheme>
</RegistryObjectList>
</SubmitObjectsRequest>
```


Copyright © OASIS Open 2005. All Rights Reserved.

Annexe D - Source model Status Type profile extension

```
<?xml version="1.0" encoding="UTF-8"?>
<SubmitObjectsRequest xmlns="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0"
xmlns:lcm="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd" xmlns:query="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:query21="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:rim="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rim30="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rs="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd">
<RegistryObjectList>
    <!-- ######################################################## -->
    <!-- ### Specifics StatusType extensions                  ### -->
    <!-- ### Sub-nodes of StatusType ClassificationScheme     ### -->
    <!-- ######################################################## -->
    <!-- No Specifics PIM profile for StatusType -->

</RegistryObjectList>
</SubmitObjectsRequest>
```


Copyright © OASIS Open 2005. All Rights Reserved.

Annexe E - Source model Content Management profile extension

```
<?xml version="1.0" encoding="UTF-8"?>
<SubmitObjectsRequest xmlns="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0"
xmlns:lcm="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd" xmlns:query="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:query21="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:rim="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rim30="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rs="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd">
<RegistryObjectList>
<rim:ClassificationNode
    parent="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM"
    lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:InvocationFile"
    id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:PIM:InvocationFile"
    code="InvocationFile" />
<rim:Service id="urn:pim:ContentManagementService:Cataloguing:PIM"
    objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:Service">
    <rim:Name>
       <rim:LocalizedString value="PIMCatalogingService"/>
    </rim:Name>
    <rim:Classification
classifiedObject="urn:pim:ContentManagementService:Cataloguing:PIM"
id="urn:oasis:names:tc:ebxml-
regrep:ContentManagementService:ContentCatalogingService"/>
    <rim:Classification
classifiedObject="urn:pim:ContentManagementService:Cataloguing:PIM"
id="urn:oasis:names:tc:ebxml-regrep:InvocationModel:Inline"/>
    <rim:Classification
classifiedObject="urn:pim:ContentManagementService:Cataloguing:PIM"
id="urn:oasis:names:tc:ebxml-
regrep:ErrorHandlingModel:LogErrorAndContinue"/>
</rim:Service>
</RegistryObjectList>
</SubmitObjectsRequest>
```


Copyright © OASIS Open 2005. All Rights Reserved.

Annexe F - Source model Discovery profile extension

```
<?xml version="1.0" encoding="UTF-8"?>
<SubmitObjectsRequest xmlns="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0"
xmlns:lcm="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd" xmlns:query="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:query21="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:rim="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rim30="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rs="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd">
<RegistryObjectList>
<rim:AdhocQuery
lid="urn:oasis:names:tc:ebxmlregrep:profile:pim:query:PersonDiscoveryQuer
y"
id="urn:oasis:names:tc:ebxmlregrep:profile:pim:query:PersonDiscoveryQuery
">
    <rim:Name>
        <rim:LocalizedString value="label.BindingDiscoveryQuery"/>
    </rim:Name>
    <rim:Description>
        <rim:LocalizedString value="Parametrised discovery query for PIM
Person"/>
    </rim:Description>
    <rim:QueryExpression
queryLanguage="urn:oasis:names:tc:ebxmlregrep:QueryLanguage:SQL-92">
        SELECT DISTINCT person.* FROM
        Person person, Slot personGender
        WHERE
        person.objectType =
“urn:oasis:names:tc:ebxmlregrep:ObjectType:RegistryObject:Person”
        AND (personName_firstName LIKE UPPER ( ''$parent.name'' )
        AND (Person.id = personGender.parent AND personGender.name_ =
''urn:oasis:names:tc:ebxmlregrep:profile:PIM:Gender''
        AND personGender.value LIKE ''$person.gender'')
    </rim:QueryExpression>
</rim:AdhocQuery>
</RegistryObjectList>
</SubmitObjectsRequest>
```


Copyright © OASIS Open 2005. All Rights Reserved.

Annexe G - Source model Notification profile and Event Type extension

```
<?xml version="1.0" encoding="UTF-8"?>
<SubmitObjectsRequest xmlns="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0"
xmlns:lcm="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd" xmlns:query="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:query21="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:rim="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rim30="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rs="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd">
<RegistryObjectList>
    <!-- ######################################################## -->
    <!-- ### Specifics EventType extensions                   ### -->
    <!-- ### Sub-nodes of EventType ClassificationScheme      ### -->
    <!-- ######################################################## -->
    <!-- No Specifics PIM profile for EventType -->
    <rim:Subscription id=${DEATH_SUBSCRIPTION_ID}
    selector="${SELECTOR_QUERY_ID}">

      <!-- email address endPoint for receiving notification via email-->
      <rim:NotifyAction notificationOption="urn:uuid:84005f6d-419e-4138-
a789-fb9fecb88f44" endPoint="mailto:farrukh.najmi@sun.com"/>

      <!—Web Service endPoint for receiving notification via SOAP -->
      <rim:NotifyAction notificationOption="urn:uuid:84005f6d-419e-4138-
a789-fb9fecb88f44" endPoint="urn:uuid:2a13e694-b3ae-4cda-995a-
aee6b2bab3d8"/>
    </rim:Subscription>

    <!-- The query used as a selector for Subscription. -->
    <query:SQLQuery id="${SELECTOR_QUERY_ID}">
      <query:QueryString>SELECT * FROM ExtrinsicObject eo WHERE
eo.objectType =
''${DEATH_EVENT_CLASSIFICATION_NODE_ID}''</query:QueryString>
    </query:SQLQuery>

    <!-- The notification listener web service and its binding -->
    <rim:Service id="${DEATH_EVENT_LISTENER_SERVICE_ID}">
      <rim:Name>
        <rim:LocalizedString value="Listens for Death Events involving
bullet wounds" xml:lang="en-US"/>
      </rim:Name>

  <rim:ServiceBinding service=${DEATH_EVENT_LISTENER_SERVICE_ID}
accessURI="http://localhost:8080/NotificationListener/notificationListene
r" id=${DEATH_EVENT_LISTENER_SERVICE_BINDING_ID}>
        <rim:Name>
          <rim:LocalizedString value="Death events listener web service
binding"
              xml:lang="en-US"/>
        </rim:Name>
      </rim:ServiceBinding>
    </rim:Service>
</RegistryObjectList>
</SubmitObjectsRequest>
```


Copyright © OASIS Open 2005. All Rights Reserved.

Annexe H - Source model Subject Role profile extension

```
<?xml version="1.0" encoding="UTF-8"?>
<SubmitObjectsRequest xmlns="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0"
xmlns:lcm="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd" xmlns:query="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:query21="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:rim="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rim30="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rs="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd">
<RegistryObjectList>
    <!-- ########################################################## -->
    <!-- ### Specifics Role extensions                          ### -->
    <!-- ### Sub-nodes of Role ClassificationScheme             ### -->
    <!-- ########################################################## -->
    <!-- No Specifics PIM profile for Role-->
</RegistryObjectList>
</SubmitObjectsRequest>
```


Copyright © OASIS Open 2005. All Rights Reserved.

Annexe I - Source model Subject Group profile extension

```
<?xml version="1.0" encoding="UTF-8"?>
<SubmitObjectsRequest xmlns="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0"
xmlns:lcm="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd" xmlns:query="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:query21="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:rim="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rim30="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rs="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd">
<RegistryObjectList>
    <!-- ########################################################## -->
    <!-- ### Specifics Group extensions                         ### -->
    <!-- ### Sub-nodes of Group ClassificationScheme            ### -->
    <!-- ########################################################## -->
    <!-- No Specifics PIM profile for Group -->
</RegistryObjectList>
</SubmitObjectsRequest>
```


Copyright © OASIS Open 2005. All Rights Reserved.

Annexe J - Source model Access Control Policy profile extension

```
<?xml version="1.0" encoding="UTF-8"?>
<SubmitObjectsRequest xmlns="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0"
xmlns:lcm="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd" xmlns:query="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:query21="urn:oasis:names:tc:ebxml-
regrep:xsd:query:3.0" xmlns:rim="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rim30="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:rs="urn:oasis:names:tc:ebxml-regrep:xsd:rs:3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:ebxml-regrep:xsd:lcm:3.0
../schema/lcm.xsd">
<RegistryObjectList>
<!-- ................ -->
</RegistryObjectList>
</SubmitObjectsRequest>
```


Copyright © OASIS Open 2005. All Rights Reserved.

Appendix A - Overview of [ebRIM]

This section summarizes the ebXML Registry Information Model [ebRIM] whose represents the target model for mapping specifics domain artefacts and concepts.

This section has only an informative purpose here and it can be omitted in developing registry profiles for a specific domain.

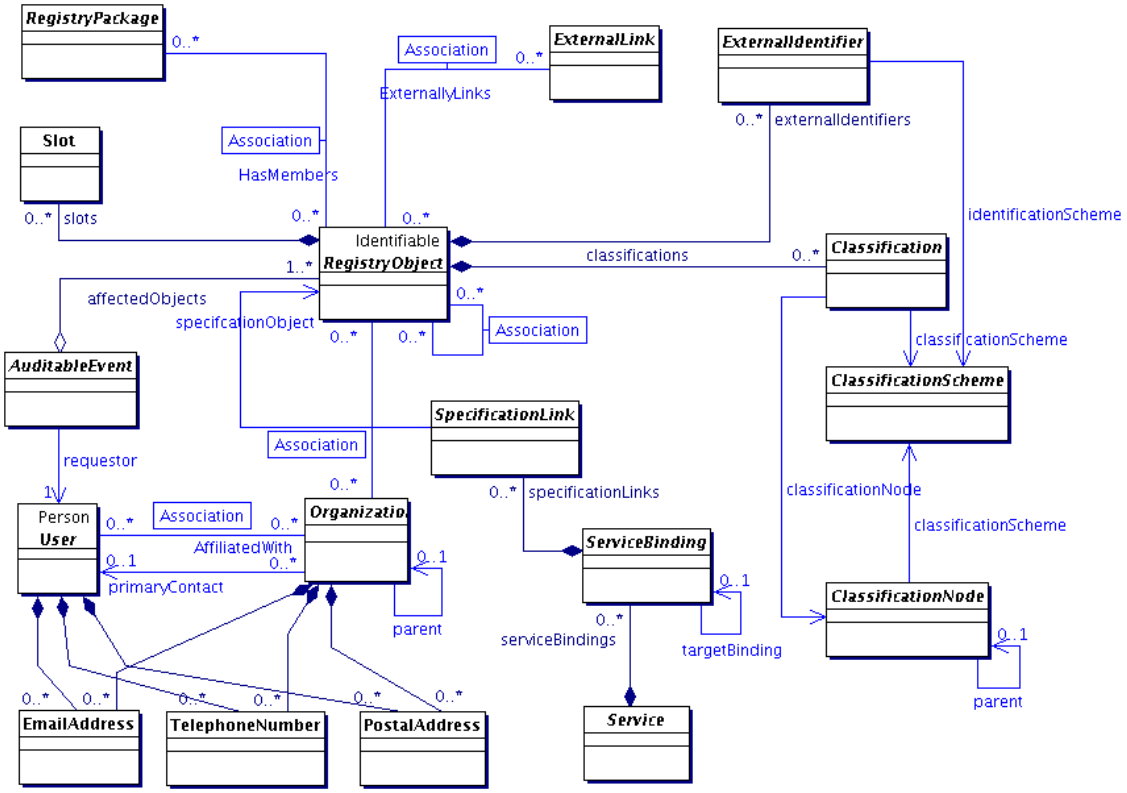


Figure 3: ebXML Registry Information Model, High Level Public View

The ebXML registry defines a Registry Information Model [ebRIM] that specifies the standard metadata that may be submitted to the registry. Figure 4 presents the UML class diagram representing the Registry Information Model. , shows the inheritance relationships in among the classes of the ebXML Registry Information Model.

Copyright © OASIS Open 2005. All Rights Reserved.

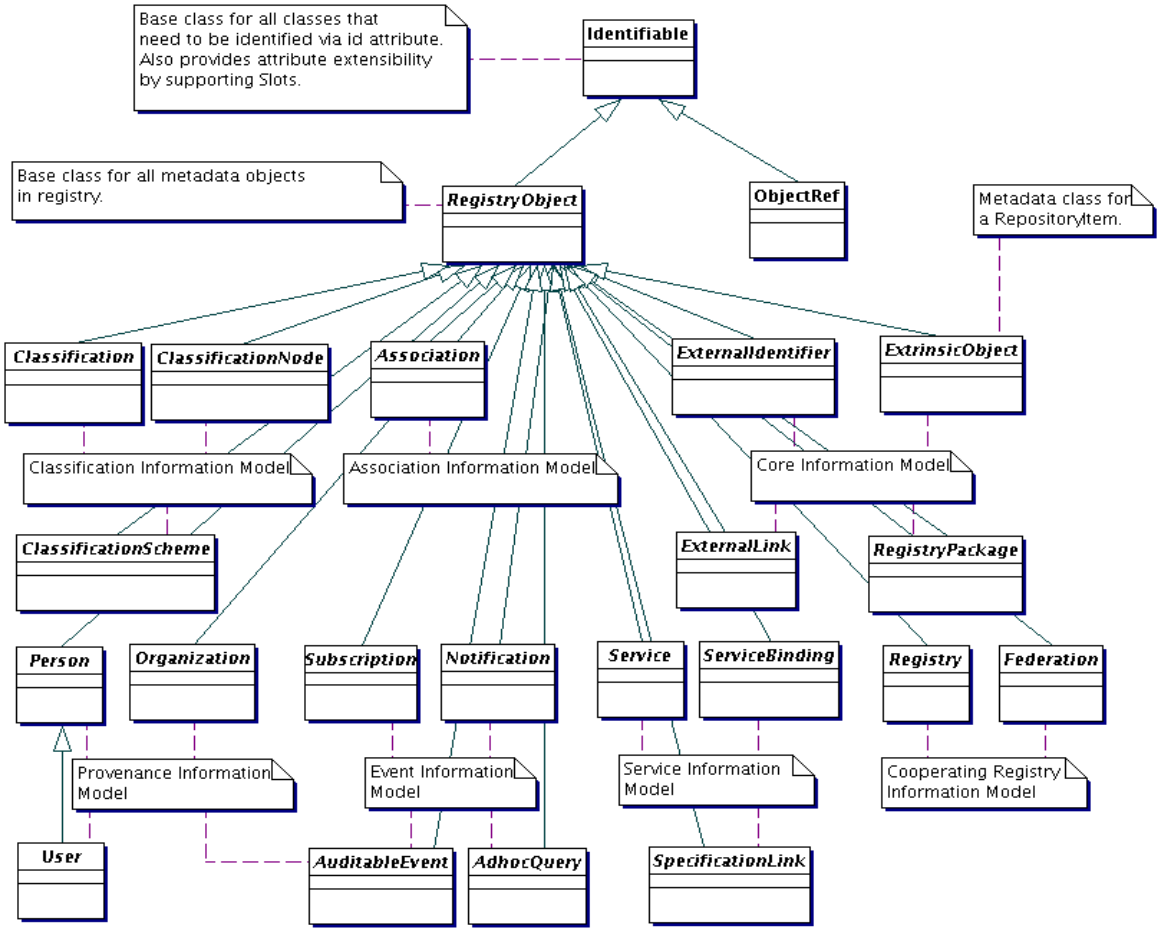


Figure 4: ebXML Registry Information Model, Inheritance View

The next few sections describe the main features of the information model.

A.1 RegistryObject

This is an abstract base class used by most classes in the model. It provides minimal metadata for registry objects. The following sections use the Organization sub-class of RegistryObject as an example to illustrate features of the model.

A.2 Object Identification

A *RegistryObject* has a globally unique id which is a URN. It MAY be a UUID based URN:

```
<rim:Organization id="urn:uuid:dafa4da3-1d92-4757-8fd8-ff2b8ce7a1bf" >
```

Listing 4: Example of UUID attribute

The id attribute value MAY potentially be human friendly.

```
<rim:Organization id="urn:oasis:Organization">
```

Listing 5: Example of human friendly id attribute

Since a RegistryObject MAY have several versions, a logical id (called lid) is also defined which is unique for different logical objects. However the lid attribute value MUST be the same for all versions of the same logical object. The lid attribute value is a URN that, as well for id attribute, MAY potentially be human friendly:

Copyright © OASIS Open 2005. All Rights Reserved.

```
<rim:Organization id=${ACME_ORG_ID}
       lid="urn:acme:ACMEOrganization">
```

Listing 6: Example of lid Attribute

A RegistryObject MAY also have any number of ExternalIdentifiers which may be any string value within an identified ClassificationScheme.

```
<rim:Organization id=${ACME_ORG_ID}
       lid="urn:acme:ACMEOrganization">

       <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
              identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
              value="ACME"/>
      </rim:ExternalIdentifier>

</rim:Organization>
```

Listing 7: Example of ExternalIdentifier

A.3 Object Naming and Description

A RegistryObject MAY have a name and a description which consists of one or more strings in one or more local languages. Name and description need not be unique across RegistryObjects.

```
<rim:Organization id=${ACME_ORG_ID}
       lid="urn:acme:ACMEOrganization">

      <rim:Name>
        <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
      </rim:Name>
      <rim:Description>
        <rim:LocalizedString value="ACME is a provider of Java software."
              xml:lang="en-US"/>
      </rim:Description>

        <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
              identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
              value="ACME"/>
      </rim:ExternalIdentifier>
</rim:Organization>
```

Listing 8: Example of Name and Description

A.4 Object Attributes

For each class in the model, [ebRIM] defines specific attributes. Examples of several of these attributes such as id, lid, name and description have already been introduced.

A.4.1 Slot Attributes

In addition the model provides a way to add custom attributes to any RegistryObject instance using instances of the Slot class. The Slot instance has a Slot name which holds the attribute name and MUST be unique within the set of Slot names in that RegistryObject. The Slot instance also has a ValueList that is a collection of one or more string values.

The following example shows how a custom attribute named "*urn:acme:slot:NASDAQSymbol*" and value "*ACME*" MAY be added to a RegistryObject using a Slot instance.

```
<rim:Organization id=${ACME_ORG_ID}
       lid="urn:acme:ACMEOrganization">

      <rim:Slot name="urn:acme:slot:NASDAQSymbol">
        <rim:ValueList>
```


Copyright © OASIS Open 2005. All Rights Reserved.

```
          <rim:Value>ACME</rim:Value>
        </rim:ValueList>
      </rim:Slot>

        <rim:Name>
         <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
      </rim:Name>
      <rim:Description>
        <rim:LocalizedString value="ACME makes Java. Provider of free Java
software."
              xml:lang="en-US"/>
      </rim:Description>
        <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
              identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
              value="ACME"/>
      </rim:ExternalIdentifier>
</rim:Organization>
```

Listing 9: Example of a Dynamic Attribute Using Slot

A.5 Object Classification

A RegistryObject may be classified using any number of Classification instances. A Classification instance references an instance of a ClassificationNode as defined by [ebRIM]. The ClassificationNode represents a value within the ClassificationScheme. The ClassificationScheme represents the classification taxonomy.

```
<rim:Organization id=${ACME_ORG_ID}
       lid="urn:acme:ACMEOrganization">
      <rim:Slot name="urn:acme:slot:NASDAQSymbol">
        <rim:ValueList>
          <rim:Value>ACME</rim:Value>
        </rim:ValueList>
      </rim:Slot>
        <rim:Name>
         <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
      </rim:Name>
      <rim:Description>
        <rim:LocalizedString value="ACME makes Java. Provider of free Java
        software." xml:lang="en-US"/>
      </rim:Description>
        <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
              identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
              value="ACME"/>
      </rim:ExternalIdentifier>

        <!--Classify Organization as a Software Publisher using NAICS Taxonomy-->
      <rim:Classification id=${CLASSIFICATION_ID}
              classificationNode=${NAICS_SOFTWARE_PUBLISHER_NODE_ID}
              classifiedObject=${ACME_ORG_ID}>

</rim:Organization>
```

Listing 10: Example of Object Classification

A.6 Object Association

Any RegistryObject MAY be associated with any other RegistryObject using an Association instance where one object is the sourceObject and the other is the targetObject of the Association instance. An Association instance MAY have an associationType which defines the nature of the association.

There are a number of predefined Association Types that a registry must support to be [ebRIM] compliant as shown in Table 1. [ebRIM] allows this list to be extensible.

The following example shows an Association between the ACME Organization instance and a Service instance with the associationType of "OffersService". This indicates that ACME Organization offers the specified service (Service instance is not shown).

Copyright © OASIS Open 2005. All Rights Reserved.

```
<rim:Association
       id=${ASSOCIATION_ID}
       associationType=${CANONICAL_ASSOCIATION_TYPE_OFFERS_SERVICE_ID}
       sourceObject=${ACME_ORG_ID}
       targetObject=${ACME_SERVICE1_ID}/>
```

Listing 11: Example of Object Association

A.7 Object References To Web Content

Any RegistryObject MAY reference web content that are maintained outside the registry using association to an ExternalLink instance that contains the URL to the external web content. The following example shows the ACME Organization with an Association to an ExternalLink instance which contains the URL to ACME's web site. The associationType of the Association MUST be of type "ExternallyLinks" as defined by [ebRIM].

```
<rim:ExternalLink externalURI="http://www.acme.com"
       id=${ACME_WEBSITE_EXTERNAL_ID}>
<rim:Association
       id=${EXTERNALLYLINKS_ASSOCIATION_ID}
       associationType=${CANONICAL_ASSOCIATION_TYPE_EXTERNALLY_LINKS_ID}
       sourceObject=${ACME_WEBSITE_EXTERNAL_ID}
       targetObject=${ACME_ORG_ID}/>
```

Listing 12: Example of Reference to Web Content Using ExternalLink

A.8 Object Packaging

RegistryObjects may be packaged or organized in a hierarchical structure using a familiar file and folder metaphor. RegistryPackage instances serve as folders while RegistryObject instances serve as files in this metaphor. A RegistryPackage instances groups logically related RegistryObject instances together as members of that RegistryPackage.

The following example creates a RegistryPackage for Services offered by ACME Organization organized in RegistryPackages according to the nature of the Service. Each Service is referenced using the ObjectRef type defined by [ebRIM].

```
<rim:RegistryPackage
       id=${ACME_SERVICES_PACKAGE_ID}>
       <rim:RegistryObjectList>
              <rim:ObjectRef id=${ACME_SERVICE1_ID}
              <rim:RegistryPackage
                     id=${ACME_PURCHASING_SERVICES_PACKAGE_ID}>
                     <rim:ObjectRef id=${ACME_ PURCHASING_SERVICE1_ID}
                     <rim:ObjectRef id=${ACME_ PURCHASING_SERVICE2_ID}
              </rim:RegistryPackage>
              <rim:RegistryPackage
                     id=${ACME_HR_SERVICES_PACKAGE_ID}>
                     <rim:ObjectRef id=${ACME_HR_SERVICE1_ID}
                     <rim:ObjectRef id=${ACME_HR_SERVICE2_ID}
              </rim:RegistryPackage>
       </rim:RegistryObjectList>
</rim:RegistryPackage>
```

Listing 13: Example of Object Packaging Using RegistryPackages

A.9 Service Description

Service description MAY be defined within the registry using the Service, ServiceBinding and SpecificationLink classes defined by [ebRIM]. This MAY be used to publish service descriptions such as WSDL and ebXML CPP/A.

Copyright © OASIS Open 2005. All Rights Reserved.

Appendix B - Method for mapping source concepts to [ebRIM]

This section provides a generic method for mapping source concepts to [ebRIM]. Editors of a profile MUSTN'T include this section within the specific profile.

B.1 Mapping of Concept

This section shows how a concept of the source model may find its corresponding binding to a [ebRIM] registry object, such as a UML class or Java method, or a XML element, or a SQL table and so on. A person applying these mapping patterns MAY choose to deviate from these patterns to compensate for special situations in the input. Any mapping pattern not covered by this document MAY be addressed in an ad hoc manner by the mapping.

More than the already known library/book, registry/repository metaphor (see [ebRIM] §1.5), every stored object within the registry has a defined “nature”, to say what it is and what it represents. This concept extends the library metaphor to a media library where the available media aren't only books, but also CDs, DVDs or paints, so the nature says if a contained object corresponds to a person, an organization, an association or whatever you want. To define this nature the registry provides some classification schemes with the idea that every contained object must have a corresponding value, the nature, within one of them. A policy, an event, a person, an image, etc.... Precisely one classification scheme serves to define the nature of the registry object, the *ObjectType* classification scheme, and several other classification schemes for defining a second level about the nature of a specific type of object. It is the case for association objects for example, where a dedicated classification scheme enumerates the available associations types, which are the “roles” played by the association in the relationship between two registry objects instances. For example an association between an organization and a person could be of type either *memberOf* or *employeeOf*.

Some of these classification schemes can be profiled by implementers for their own specific usage.

Throughout this section we try to depict some possible extensions that can be applied in a deployment profile.

B.2 Using ClassificationSchemes

The [ebRIM] classification scheme system adopts the tree structure graph where each node has zero or more child nodes, which are below it in the tree. So every new concept MAY be added within the registry as a child node under the corresponding classification scheme root node or sub-node. So concretely to extend a canonical list means to add one or more sub-nodes, *CassificationNode*, or root elements, *ClassificationScheme*.

B.2.1 Use Cases for ClassificationSchemes

The following are some of the many use cases for *ClassificationSchemes* in an ebXML Registry:

- Used to classify *RegistryObjects* to facilitate discovery based upon that classification. This is the primary role of *ClassificationSchemes* in ebXML Registry.
- Used to define all possible values of an Enumeration class. For example, the pim.Gender class is represented in ebRIM as a Gender *ClassificationScheme*.
- Used to define the data types supported by a registry (DataType scheme).
- Used to define the concepts supported by a registry (ObjectType scheme).
- Used to define the association types supported by the registry (AssociationType scheme).

Copyright © OASIS Open 2005. All Rights Reserved.

- Used to define the security roles that may be defined for users of the registry (SubjectRole scheme).
- Used to define the security groups that may be defined for users of the registry (SubjectGroup scheme).

B.2.2 Canonical ClassificationSchemes

There are several ClassificationSchemes that are specified by ebRIM and required to be present in every ebXML Registry. Such standard ClassificationSchemes are referred to as "canonical" ClassificationSchemes.

An ebXML Registry user MAY extend existing canonical ClassificationsSchemes or add new domain specific ClassificationSchemes. However, they cannot update/delete the existing canonical ClassificationScheme or update/delete its ClassificationNodes.

B.2.3 Extending ClassificationSchemes

A registry user MAY extend an existing ClassificationScheme regardless of whether it is a canonical scheme or a user defined scheme as long as the Access Control Policies for the scheme and its nodes allow the user that privilege. The user may extend an existing scheme by submitting new ClassificationNodes to the registry that reference existing ClassificationNodes or an existing ClassificationScheme as the value of their "parent" attribute. The user SHOULD assign a logical id (lid) to all user defined ClassifinationNodes for ease of identification.

B.2.3.1 Use Cases for Extending ClassificationSchemes

The following are some of the most common use cases for extending ClassificationSchemes:

- Extending the ObjectType scheme to define new Classes supported by a registry. Listing 14 shows an example of extending the ObjectType scheme.
- Extending the AssociationType scheme to define the association types supported by the registry. Listing 23 shows an example of extending the AssociationType scheme.
- Extending the SubjectRole scheme to define the security roles that may be defined for users of the registry.

B.2.4 Defining New ClassificationSchemes

A user may submit an entirely new ClassificationScheme to the registry. Often the scheme is a domain specific scheme for a specialized purpose. When mapping a domain specific model there are many situations where a new ClassificationScheme needs to be defined.

B.3 Mapping of Object

[ebRIM] already defines several canonical registry objects and object types that can be used as a target mapping for specific source concepts. These are for example *RegistryPackage, Service, Notification, Person, Organization, XML* and so on, but a specific implementation of a registry could change the outline of the canonical list in order to better matching the original concept within a registry. To change the standard outline is enough to extend the *objectType* classification scheme for objects and the *associationType* classification scheme for association of the standard registry classification schemes.

The most natural place where new concepts and objects can be added to the registry, if the standard registry objects don't meet the need, is under the [ebRIM] canonical *ExtrinsicObject* object type, as showed in the sub-section below.

B.3.1 Defining a Sub-Node of ExtrinsicObject

If a source concept doesn't fully match a canonical meaningful registry object than implementers SHOULD use the *ExtrinsicObject* registry object. The *ExtrinsicObject* is a generic registry object that serves as the primary metadata object for a *RepositoryItem,* but its nature can be specialized using the *objectType* attribute in order to provide a simple way

Copyright © OASIS Open 2005. All Rights Reserved.

for discovering and grouping the registry content.

The value of the *objectType* attribute MUST be a reference to a *ClassificationNode* in the canonical *ObjectType ClassificationScheme,* but this list MAY be extended and new *ClassificationNodes* MAY be added as childs or descendent of the canonical *ClassificationNode* for *ExtrinsicObject*.

For example to extend the *ObjectType ClassificationScheme* for the LifeEvent classes in PIM (illustrated in section Figure 1) the following *ClassificationNode* hierarchy MUST be submitted to the ebXML Registry via a *SubmitObjectsRequest*.

Note that:

- The id attribute values SHOULD have actual id values.
- The parent attribute of the LifeEvent ClassificationNode is the id of the ExtrinsicObject ClassificationNode in the ObjectType ClassificationScheme.
- Figure 5 shows the structure of the ObjectType ClassificationScheme before and after the extension for mapping the LifeEvent classes from PIM.

```
<!-- Add LifeEvent classes to ObjectType ClassificationScheme -->
<rim:ClassificationNode code="LifeEvent" id="${LIFE_EVENT_NODE_ID}"
       parent="urn:uuid:baa2e6c8-873e-4624-8f2d-b9c7230eb4f8">
  <rim:Name>
    <rim:LocalizedString charset="UTF-8" value="LifeEvent"/>
  </rim:Name>
  <rim:ClassificationNode code="BirthEvent"
       id="${BIRTH_EVENT_NODE_ID}">
    <rim:Name>
      <rim:LocalizedString charset="UTF-8" value=" BirthEvent "/>
    </rim:Name>
  </rim:ClassificationNode>
  <rim:ClassificationNode code="MarriageEvent"
       id="${MARRIAGE_EVENT_NODE_ID}">
    <rim:Name>
      <rim:LocalizedString charset="UTF-8" value=" MarriageEvent "/>
    </rim:Name>
  <rim:ClassificationNode code="BirthingEvent"
       id="${BIRTHING_EVENT_NODE_ID}">
    <rim:Name>
      <rim:LocalizedString charset="UTF-8" value=" BirthingEvent "/>
    </rim:Name>
  </rim:ClassificationNode>
  <rim:ClassificationNode code="DeathEvent"
       id="${DEATH_EVENT_NODE_ID}">
    <rim:Name>
      <rim:LocalizedString charset="UTF-8" value=" DeathEvent "/>
    </rim:Name>
  </rim:ClassificationNode>
</rim:ClassificationNode>
<rim:ExtrinsicObject id="${EO_BirthEvent_UUID}"
objectType="${BIRTH_EVENT_NODE_ID}">
       <rim:Slot name="timeStamp" slotType="urn:oasis:names:tc:ebxml-
regrep:DataType:DateTime" >
              <rim:ValueList><rim:Value>2006-05-
15T11:12:12</rim:Value></rim:ValueList>
       </rim:Slot>
       <rim:Slot name="location" slotType="urn:oasis:names:tc:ebxml-
regrep:String" >
              <rim:ValueList><rim:Value>Caen</rim:Value></rim:ValueList>
       </rim:Slot>
       <rim:Slot name="hospital" slotType="urn:oasis:names:tc:ebxml-
regrep:String" >
              <rim:ValueList><rim:Value>Clinique Saint
Martin</rim:Value></rim:ValueList>
```


Copyright © OASIS Open 2005. All Rights Reserved.

```
      </rim:Slot>
</rim:ExtrinsicObject>
```

Listing 14: Example of Adding LifeEvent Classes and sub-classes to ObjectType ClassificationScheme with a BirthEvent Extrinsic object instance

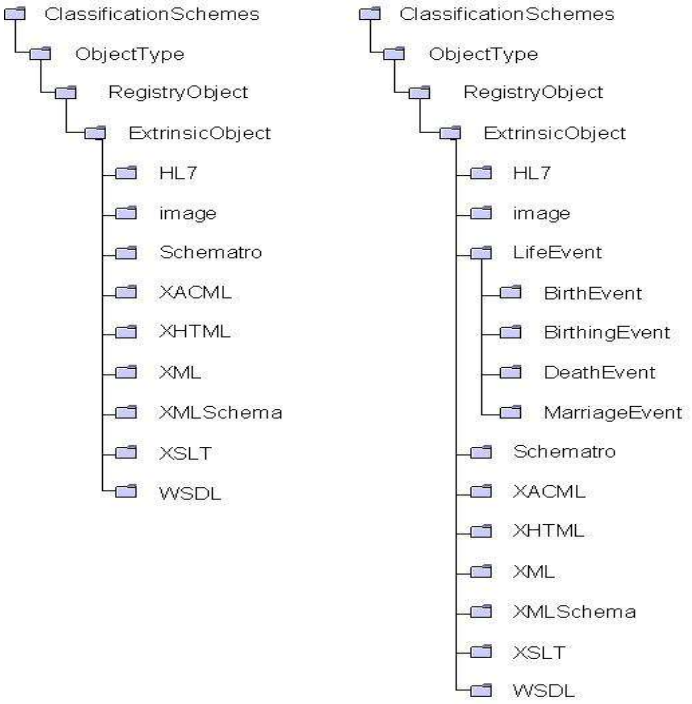


Figure 5: ObjectType ClassificationScheme: Before and After Extension for LifeEvent

B.4 Mapping of Attributes

This section defines how attributes of a class in the source model are mapped to [ebRIM]. Mapping of the source class to [ebRIM] has been discussed in section .

B.4.1 Mapping to Identifier

Section above describes the various ways that a RegistryObject may be identified in [ebRIM].

B.4.1.1 Mapping to id Attribute

If the identifier value in source model conforms to a UUID based URN as shown below,

```
urn:uuid:dafa4da3-1d92-4757-8fd8-ff2b8ce7a1bf
```

Listing 15: Example of id attribute

and if it provides a globally unique identifier for the source class then it MUST be mapped to the id attribute in the target [ebRIM] object. Note that if the identifier value in the source model MUST be the same across different versions of the same logical instance of the source object then it MUST not be mapped to the id attribute. Instead it SHOULD be mapped to the Logical id (lid) attribute as defined next.

Even if [ebRIM] permits to define ids in a more natural language, implementer should consider the usage of UUID for instances of registry objects, an exception can be done for new classification nodes, and use the LID attribute for ids in natural language.

Copyright © OASIS Open 2005. All Rights Reserved.

For a detailed description of the versioning capabilities of ebXML Registry and the lid attribute please see [ebRS] and [ebRIM] respectively.

B.4.1.2 Mapping to Logical Id (lid) Attribute

If the identifier value in the source model may be the same across all versions of an instance of the class then it SHOULD be mapped to the lid attribute of the target class in [ebRIM]. The registry requires that the lid attribute value:

- MAY be a URN
- MUST be unique across all logical RegistryObjects in the registry
- MUST be the same across all versions of the same logical RegistryObject

The lid attribute is a good way to assign a meaningful identifier to a RegistryObject. If the source attribute is a human friendly identifier for the source class then it MAY be a good candidate to be mapped to the lid attribute. Note that the source attribute value need not be a URN. If it is not a URN, then the mapping SHOULD define a deterministic algorithm for mapping the non-URN value to a URN value that meets above constraints on lid attribute values.

For example, the name attribute of a Person instance in PIM MAY be mapped to the lid attribute on the Person class in [ebRIM] sing the following algorithm:

```
lid = "urn:pim:" + Person.name
```

For example the rim.Person instance for "MarieCurie" would look like:

```
<rim:Person id=${MARIECURIE_PERSON_ID}
       lid = "urn:pim:MarieCurie">
…
</rim:Person>
```

Note that above example is slightly flawed because use of a person's name in the algorithm does not guarantee that the lid would be unique since another person could have the same exact name. Also note that the urn:pim namespace MUST be registered with IANA to truly guarantee that it is a unique name space.

B.4.1.3 Mapping to ExternalIdentifier

If the attribute in the source model is an identifier for the source class instances but does not map to an id or lid attribute then it SHOULD be mapped to an *ExternalIdentifier* in [ebRIM]. The mapping MUST specify a *ClassificationScheme* instance that MUST be used as *identificationScheme* for the *ExternalIdentifier*.

For example, the *nationalId* attribute of the Person class in PIM may be mapped to an *ExternalIdentifier* that uses a *ClassificationScheme* named "*NationalIdentifierScheme*" as its *identificationScheme* attribute value. The mapping is responsible for defining the "*NationalIdentifierScheme*" *ClassificationScheme* as described in section 8.3.

```
<rim:Person id=${MARIECURIE_PERSON_ID}
       lid="urn:pim:MarieCurie">

       <rim:ExternalIdentifier id=${NATIONAL_ID_EXTERNAL_IDENTIFIER_ID}
              identificationScheme=${NATIONAL_ID_CLASSIFICATIONSCHEME_ID}
              value="123-45-6789"/>
     </rim:ExternalIdentifier>

       …
</rim:Person>
```

Listing 16: Example of Mapping to ExternalIdentifier

Copyright © OASIS Open 2005. All Rights Reserved.

B.4.2 Mapping to Name and Description

If the source attribute provides a name or description for the source class instance then it SHOULD be mapped to the name or description attribute of the *RegistryObject* class in [ebRIM]. The rim.RegistryObject.name and rim.RegistryObject.description attributes are of type *InternationalString* which can contain the name and description value is multiple locales as composed *LocalizedString* instances. This means that the mapping SHOULD map the name and description to the appropriate locale.

For example the pim.Person class has a name attribute of data type String. The mapping SHOULD map it to the rim.Person.name attribute as shown below:

```
<rim:Person id=${MARIECURIE_PERSON_ID}
       lid="urn:pim:MarieCurie">

      <rim:Name>
        <rim:LocalizedString value="Marie Curie" xml:lang="en-US"/>
        <rim:LocalizedString value="Marie Curie" xml:lang="fr"/>
      </rim:Name>
       …
</rim:Person>
```

Listing 17: Example of Mapping to name Attribute

Note that the xml:lang attribute in above example SHOULD be omitted when the default locale is implied. Since a person's name does not change with locale the above example would be better off specifying a single *LocalizedString* with no xml:lang attribute specified. It is showing multiple locales for illustration purposes only.

B.4.3 Mapping to Classification

If the source attribute is somehow classifying or categorizing the class instance then it SHOULD be mapped to a Classification in [ebRIM]. For an overview of Classification see section .

For example, the rim.Person.gender attribute is of data type Gender which is an Enumeration class where the enumerated set of values are "Male", "Female" and "Other". The mapping MAY map pim.Person.gender to a Classification on a rim.Person instance. Since a Classification requires a *ClassificationScheme*, the mapping MUST specify the *ClassificationScheme*.

```
<rim:Person id=${MARIECURIE_PERSON_ID}
       lid="urn:pim:MarieCurie">

       <!--Classify Person as a Female using the Gender Taxonomy-->
      <rim:Classification id=${GENDER_CLASSIFICATION_ID}
              classificationNode=${GENDER_FEMALE_NODE_ID}
              classifiedObject=${MARIECURIE_PERSON_ID}>
       …
</rim:Person>
```

Listing 18: Example of Mapping to name Attribute

Note that in above example the Gender *ClassificationScheme* is indirectly referenced via the *ClassificationNode* for "Female" within that taxonomy.

B.4.4 Mapping to ExternalLink

If the source attribute will always contain a URL (or a URN) then it SHOULD be mapped to an *ExternalLink*. For an overview of *ExternalLink* see section .

For example, the rim.Person.homepage attribute, if not null, always contain the URL for the Person's homepage. It SHOULD therefore be mapped to an *ExternalLink* as shown below.

Note that an *ExternalLink* MUST be related to a *RegistryObject* using an *Association* instance in [ebRIM]. This allows the same *ExternalLink* to be shared by many *RegistryObject*

Copyright © OASIS Open 2005. All Rights Reserved.

instances.

```
<rim:Person id=${MARIECURIE_PERSON_ID}
        lid="urn:pim:MarieCurie">
        …
</rim:Person>

<rim:ExternalLink externalURI="http://www.aip.org/history/curie/"
        id=${MARIECURIE_WEBSITE_EXTERNAL_LINK_ID}>

<rim:Association
        id=${MARIECURIE_HOMEPAGE_EXTERNALLYLINKS_ASSOCIATION_ID}
        associationType=${CANONICAL_ASSOCIATION_TYPE_EXTERNALLY_LINKS_ID}
        sourceObject=${MARIECURIE_WEBSITE_EXTERNAL_LINK_ID}
        targetObject=${MARIECURIE_PERSON_ID}/>
```

Listing 19: Example of Mapping to ExternalLink

B.4.5 Direct Mapping to ebRIM Attribute

In some cases an attribute in the source model concept may closely match an attribute in the [ebRIM] registry object. This is the most direct and preferred attribute mapping.

For example the Person class in PIM has an attribute "phone" (referred to as pim.Person.phone) whose semantics closely match the attribute "telephoneNumbers" in the Person class in [ebRIM] (referred to as rim.Person.telephoneNumbers). Thus it is preferred that the pim.Person.phone attribute is mapped to rim.Person.telephoneNumbers. Impedance mismatches between the source attribute data type and target attribute data type MAY be handled by the mapper using domain specific knowledge. For example the pim.Person.phone attribute is of data type String while the rim.Person.telephoneNumbers attribute is of data type *TelephoneNumber* where *TelephoneNumber* consists of several String attributes:

- "areaCode"
- "countryCode"
- "number"

Thus the mapper MUST choose which rim.TelephoneNumber attribute should be used for the pim.Person.phone attribute mapping. As an example they MAY chose to map it the rim.TelephoneNumber.number attribute. Alternatively, they may define a domain specific algorithm for splitting the pim.Person.phone attribute into one, two or three components that map to the various *TelephoneNumber* attributes in a deterministic manner.

B.4.6 Mapping to Slot

When all other options for mapping the source attribute are inadequate then the attribute MUST be mapped to a Slot.

B.4.6.1 Mapping to rim.Slot.slotName

The source attribute name SHOULD be mapped to the rim.Slot.slotName attribute. To prevent name conflicts the mapping MAY define a mapping algorithm that generates a URN with the source attribute name as its last component. It is also suggested that the source class name be the second last component of the URN.

For example, the pim.Person.profession attribute SHOULD be mapped to a URN like:

```
<rim:Person id=${MARIECURIE_PERSON_ID}
    lid="urn:pim:MarieCurie">
      <rim:Slot name="urn:pim:Person:profession">
    …
      </rim:Slot>
    …
</rim:Person>
```

Listing 20: Example of Mapping pim.Person.Profession to slotName

Copyright © OASIS Open 2005. All Rights Reserved.

B.4.6.2 Mapping to rim.Slot.slotType

The rim.Slot.slotType attribute value SHOULD be defined so it conveys the data type semantics of the Slot value. The value of the rim.Slot.slotType attribute SHOULD be the lid attribute value of a ClassificationNode in the canonical DataType ClassificationScheme.

For example, the data type of the pim.Person.profession in PIM is String. It MUST therefore be mapped to the rim.Slot.slotType value of:

```
<rim:Person id=${MARIECURIE_PERSON_ID}
    lid="urn:pim:MarieCurie">
     <rim:Slot name="urn:pim:Person:profession"
       slotType="urn:oasis:names:ebXML-regrep:DataType:String">
       …
     </rim:Slot>
    …
</rim:Person>
```

Listing 21: Example of Mapping DataType to slotType

Note that if the data type happens to be a Collection then the *slotType* should reflect the data type of the Collection elements. In case of a heterogeneous Collection the most specific data type from the *DataType ClassificationScheme* MUST be used.

B.4.6.3 Mapping to rim.Slot.values

The rim.Slot.values (ValueList in XML Schema) SHOULD be defined as follows:

- If the value is a reference (datatype/slotType is urn:oasis:names:ebXML-regrep:DataType:ObjectRef) to another RegistryObject then the value MUST be the value of the id attribute of the RegistryObject being referenced.
- If the datatype of the source attribute is not a Collection then there should only be a single “rim:Value” within the ValueList.
- If the datatype of the source attribute is a Collection then there MAY be a multiple “rim:Value” within the ValueList.

The following example shows how the pim.Person.profession attribute is specified when mapping a pim.Person instance to a rim.Person instance.

```
<rim:Person id=${MARIECURIE_PERSON_ID}
    lid="urn:pim:MarieCurie">
     <rim:Slot name="urn:pim:Person:profession"
       slotType="urn:oasis:names:ebXML-regrep:DataType:String">
       <rim:ValueList>
         <rim:Value>Scientist</rim:Value>
       </rim:ValueList>
     </rim:Slot>
    …
</rim:Person>
```

Listing 22: Example of Mapping Attribute value to Value

B.4.7 Enumerated Type Mapping

A source attribute whose data type is an Enumeration class SHOULD be mapped to a Classification on the target *RegistryObject*. An example of this has been provided with the mapping of the pim.Person.gender attribute in section .

B.5 Mapping of Associations

If a source concept corresponds to a relationship between two other concepts, or registry objects, implementers should map that to the [ebRIM] *Association* registry object. An important feature that the registry adds to the association is that it considers them as real objects and such that they can be used not only as a link between objects but an object itself. This means that as for *ExtrinsicObject,* they have attributes and slots. So for example a

Copyright © OASIS Open 2005. All Rights Reserved.

UML Association class can be directly mapped to a registry object *Association.*

[ebRIM] already defines a set of canonical types of association and this list can be extended as for the canonical *objectType* classification scheme seen above.

B.5.1 Defining a New Association Type

This section provides the steps to define a new Association Type.

To define a Association Type implementers MUST extend the canonical *AssociationType ClassificationScheme* and add a new *ClassificationNode* as a child or descendent of the *AssociationType ClassificationScheme*.

For example to extend the *AssociationType ClassificationScheme* for the "spouse", "husband" and "wife" association in PIM the following *ClassificationNode* hierarchy SHOULD be submitted to the ebXML Registry via a *SubmitObjectsRequest*.

Note that:

- Figure 6 shows the structure of the *AssociationType ClassificationScheme* before and after the extension for mapping the Spouse Association Types from PIM.
- It is a good idea to organize *AssociationTypes* hierarchically even though the source model may not have those semantics defined. For example it makes good sense to define the "Husband" and "Wife" *AssociationTypes* as children of the "Spouse" *AssociationType*.

```
<!-- Add Spouse, Husband, Wife to AssociationType ClassificationScheme --
>
<rim:ClassificationNode code="Spouse" id="${SPOUSE_NODE_ID}"
      parent="urn:oasis:names:tc:ebxml-
regrep:classificationScheme:AssociationType">
  <rim:Name>
    <rim:LocalizedString charset="UTF-8" value="Spouse"/>
  </rim:Name>
  <rim:ClassificationNode code="Husband"
       id="${HUSBAND_NODE_ID}">
    <rim:Name>
      <rim:LocalizedString charset="UTF-8" value=" Husband "/>
    </rim:Name>
  </rim:ClassificationNode>
  <rim:ClassificationNode code="Wife"
       id="${WIFE_NODE_ID}">
    <rim:Name>
      <rim:LocalizedString charset="UTF-8" value=" Wife "/>
    </rim:Name>
</rim:ClassificationNode>
```

Listing 23: Example of Adding Spouse Association Types

Copyright © OASIS Open 2005. All Rights Reserved.

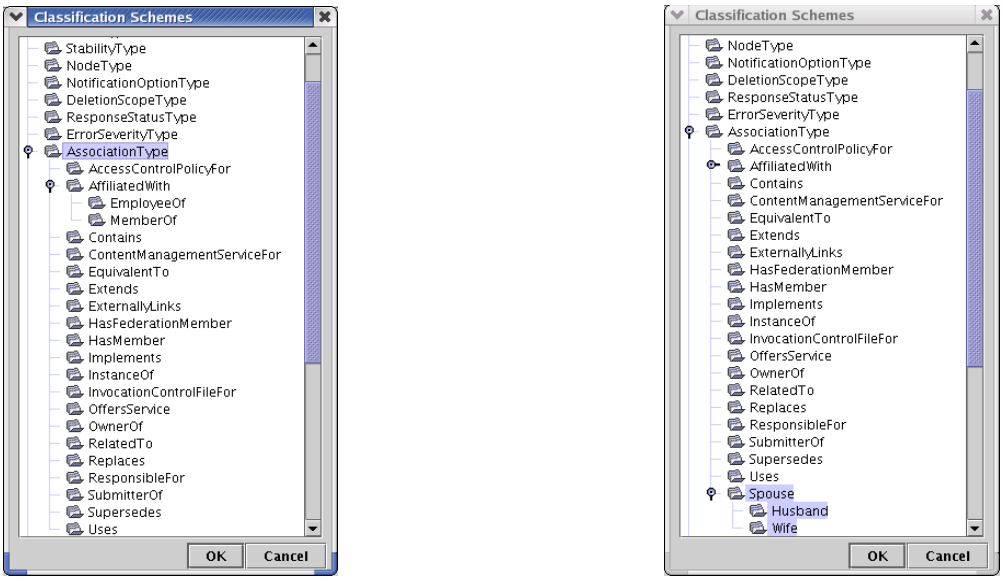


Figure 6: ObjectType ClassificationScheme: Before and After Extension For Spouse

Figure 7 shows an example UML instance diagram to show two Associations between Person "PierreCurie" and Person "MarieCurie" in PIM. Note that the husbandToWife association has "PierreCurie" as the sourceObject and "MarieCurie" as the targetObject while the wifeToHusband associations has the two reversed.

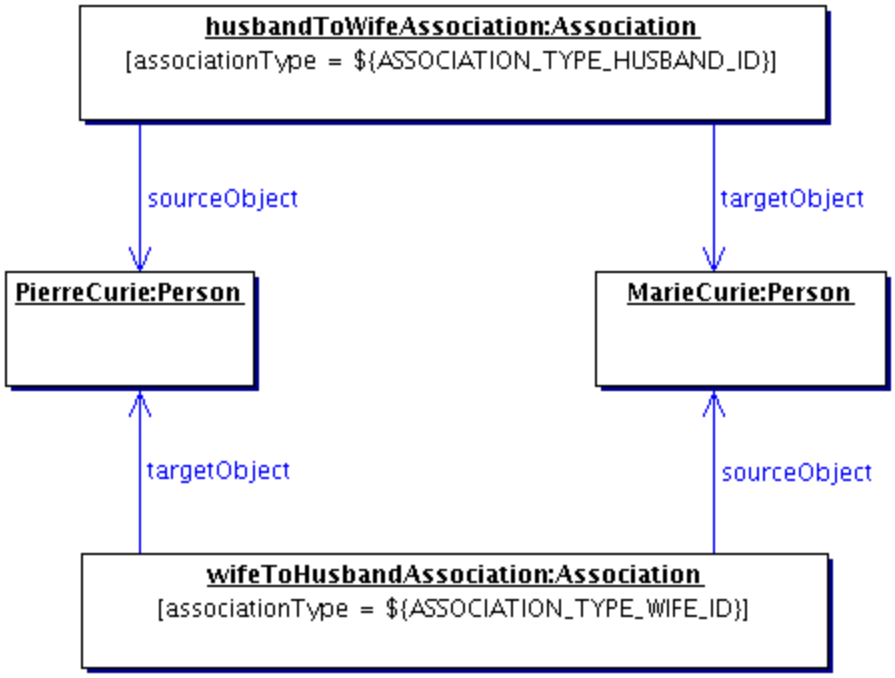


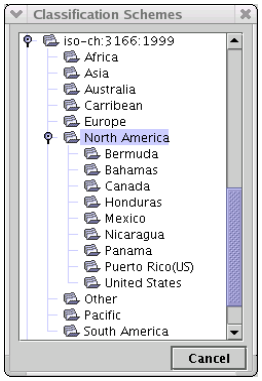
Figure 7: Sample Association instance between a Husband and Wife pair

B.6 Mapping of Taxonomies to the registry classification system

The ebXML Registry provides a powerful, simple and flexible capability to create, extend and apply taxonomies to address a wide set of use cases. A taxonomy in ebRIM is mapped to a

Copyright © OASIS Open 2005. All Rights Reserved.

2029 *ClassificationScheme*. The allowed values in a *ClassificationScheme* are represented by
2030 *ClassificationNode* instances within [ebRIM].



2031
2032 **Figure 8: Geography ClassificationScheme Example**

2033 Figure 8 shows a geography *ClassificationScheme*. It is a hierarchical tree structure where
2034 the root of the tree “iso-ch:3166:1999” is the name of the *ClassificationScheme* while the
2035 rest of the nodes in the tree are *ClassificationNodes*.

Copyright © OASIS Open 2005. All Rights Reserved.

2036 # Appendix C - Revision History

Rev	Date	By Whom	What
0.1	June 15, 2005	Ivan Bedini Farrukh Najmi Nikola Stojanovic	First draft
0.1.1	July 13, 2005	Ivan Bedini	Document Aligned to ebXML IIC profile template. Added profiling [ebRIM] chapter. Added profiling [ebRS] chapter Added Appendix A
0.1.2	September 2005	Ivan Bedini	Assembled the UML guide in the chapter 3 Mineur editing changes. Added Diego's suggests and comments.
0.2	December 2005	Ivan Bedini	Several changes to the document structure. Removed the UML mapping to ebRIM tutorial part.
0.2.1	January 2006	Ivan Bedini	Added F. Najmi comments
0.2.2	January 2006	Ivan Bedini	Changed the introduction section, added appendixes and other minor changes
0.2.3	February 2006	Ivan Bedini	Added NS comments and added the method mapping section
0.2.4	February 2006r	Ivan Bedini	Changed the document structure

2037

Copyright © OASIS Open 2005. All Rights Reserved.

Appendix D - References

D.1 Normative

[ebRIM] ebXML Registry Information Model version 3.0

http://docs.oasis-open.org/regrep/regrep-rim/v3.0/regrep-rim-3.0-os.pdf

[ebRS] ebXML Registry Services Specification version 3.0

http://docs.oasis-open.org/regrep/regrep-rs/v3.0/regrep-rs-3.0-os.pdf

[UML] Unified Modeling Language version 1.5
http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.pdf

D.2 Informative

[ebRR-UML-TUT] ebXML Registry Tutorial: UML to ebRIM mapping

[CMRR] Web Content Management Using OASIS ebXML Registry
http://ebxmlrr.sourceforge.net/presentations/xmlEurope2004/04-02-02.pdf
http://ebxmlrr.sourceforge.net/presentations/xmlEurope2004/xmlEurope2004-webcm-ebxmlrr.sxi
http://ebxmlrr.sourceforge.net/presentations/xmlEurope2004/xmlEurope2004-webcm-ebxmlrr.ppt

[IMPL] ebXML Registry 3.0 Implementations
freebXML Registry: A royalty free, open source ebXML Registry Implementation
http://ebxmlrr.sourceforge.net

[TUT] UML Tutorials
Borland Tutorial
http://bdn.borland.com/article/0,1410,31863,00.html
Sparx Systems UML Tutorial
http://www.sparxsystems.com.au/UML_Tutorial.htm

Copyright © OASIS Open 2005. All Rights Reserved.