ebXML Registry – A Tutorial

Version 0.5

Draft Technical Note, 15 Avril, 2005

Document identifier:

regrep-tutorial-05

Location:

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

Editors:

Name	Affiliation
Farrukh Najmi	Sun Microsystems
Nikola Stojanovic	RosettaNet

Contributors:

Name	Affiliation
Diego Ballve	Individual
Ivan Bedini	France Telecom

Abstract:

This document is a tutorial on how to effectively customize and use an ebXML Registry Repository for specific domains and applications. The document includes a standard methodology for mapping a domain specific information model (in UML format) to the ebXML Registry Information Model.

Status:

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

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/).

Table of Contents

1 Introduction 5
  1.1 Terminology 5
  1.2 Conventions 5
2 Overview 7
  2.1 Overview of UML 7
  2.2 Overview of Person Information Model 7
  2.3 Overview of ebXML Registry Information Model 8
    2.3.1 RegistryObject 10
    2.3.2 Object Identification 10
    2.3.3 Object Naming and Description 11
    2.3.4 Object Attributes 11
      2.3.4.1 Slot Attributes 11
    2.3.5 Object Classification 12
    2.3.6 Object Association 13
    2.3.7 Object References To Web Content 13
    2.3.8 Object Packaging 14
    2.3.9 Service Description 14
3 Mapping a Domain Specific Model to ebRIM 15
  3.1 Class Mapping 15
    3.1.1 Defining a Sub-Class of ExtrinsicObject 15
  3.2 Interface Mapping 17
  3.3 Inheritance Mapping 17
    3.3.1 Mapping of Multiple Inheritance 17
  3.4 Method Mapping: 17
  3.5 Association Mapping 18
    3.5.1 Navigability / Direction Mapping 18
    3.5.2 Role Name / Association Name Mapping 18
    1.1.1.1Defining a New Association Type 18
    3.5.4 Aggregation Mapping 20
    3.5.5 Composition Mapping 20
    3.5.6 N-ary Association Mapping 21
    3.5.7 XOR Associations 21
  3.6 Attribute Mapping 22
    3.6.1 Mapping to Identifier 23
      3.6.1.1 Mapping to id Attribute 23
      3.6.1.2 Mapping to Logical Id (lid) Attribute 23
      3.6.1.3 Mapping to ExternalIdentier 24
    3.6.2 Mapping to Name and Description 24
    3.6.3 Mapping to Classification 25
    3.6.4 Mapping to ExternalLink 25
    3.6.5 Direct Mapping to ebRIM Attribute 26
    3.6.6 Mapping to Slot 26
      3.6.6.1 Mapping to rim.Slot.slotName 26

3.6.6.2 Mapping to rim.Slot.slotType....27
3.6.6.3 Mapping to rim.Slot.values....27
3.7 Enumerated Type Mapping....28
4 Using ClassificationSchemes....29
4.1 Use Cases for ClassificationSchemes....29
4.2 Canonical ClassificationSchemes....29
4.3 Extending ClassificationSchemes....30
4.3.1 Use Cases for Extending ClassificationSchemes....30
4.4 Defining New ClassificationSchemes....30
4.4.1 Use Cases for Defining New ClassificationSchemes....30
5 Defining Content Management Services....31
5.1 Defining Content Validation Services....31
5.2 Defining Content Cataloging Services....31
6 Defining Domain Specific Queries....32
6.1 Identifying Common Discovery Use Cases....32
7 Using the Event Notification Feature....33
7.1 Use Cases for Event Notification....33
7.2 Creating Subscriptions for Events....33
8 Defining Access Control....36
8.1 Subject Role Extension....36
8.2 Subject Group Extension....36
8.2.1 Defining Custom Access Control Policies....36
9 Known Issues....37

Illustration Index

Figure 1: Person Information Model: A Sample Domain Specific Model...7
Figure 2: Person Information Model: Inheritance View....................................... 8
Figure 3: ebXML Registry Information Model, High Level Public View..9
Figure 4: ebXML Registry Information Model, Inheritance View..10
Figure 5: ObjectType ClassificationScheme: Before and After Extension for LifeEvent 17
Figure 6: ObjectType ClassificationScheme: Before and After Extension For Spouse.........................19
Figure 7: Sample Association instance between a Husband and Wife pair..20
Figure 8: Attribute Mapping Algorithm Flowchart.. 22
Figure 9: Geography ClassificationScheme Example..29

1 Introduction

This document is a tutorial on how to effectively customize and use an ebXML Registry for specific domains and applications. The document includes a standard methodology for mapping a domain specific information model to the ebXML Registry Information Model.

As more and more organization are adopting ebXML Registry standard they are faced with the recurring need to map between their domain specific information model to the ebXML Registry Information Model [ebRIM] in order to use the registry to manage their domain specific artifacts. Currently this mapping is being done in an ad hoc manner.

This technical note provides the necessary guidelines, design patterns and algorithms to customize an ebXML Registry for a specific domain. Specifically, it enables a consistent mapping from domain specific information models to ebXML Registry Information Model.

It is not the purpose of this document to educate the reader on ebXML Registry [ebRIM], [ebRS], information modeling or the Unified Modeling Language [UML]. The reader of this document should have a good understanding of the ebXML Registry specifications and the UML 1.5 specification.

1.1 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].

1.2 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:

- **UML Diagrams**

  UML diagrams are used as a way to concisely describe information models in a standard way. They are not intended to convey any specific *Implementation* or methodology requirements.

- **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">
```

1. 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>
```

2 Overview

This chapter provides an overview of ebXML Registry Information Model [ebRIM] and the sample domain specific Person Information Model (PIM). The PIM is the source information model for the mapping patterns defined by this document. The [ebRIM] is the target for the mapping patterns defined by this document.

The information presented is informative and is not intended to replace the normative information defined by ebXML Registry and UML specifications.

2.1 Overview of UML

This document will not provide an overview of UML. The reader SHOULD review UML tutorials [TUT] to get a rapid understanding of [UML]. The reader MAY refer to [UML] if a deeper understanding is needed.

Although UML defines many different types of diagrams the focus of this document is the UML Class diagram. The reader SHOULD familiarize themselves with the UML Class Diagram notation using [TUT] and [UML].

2.2 Overview of Person Information Model

Throughout this document we use a sample domain specific information model called Person Information Model (PIM). This document will demonstrate the mapping principals described using the PIM as source model and [ebRIM] as the target model for the mapping.

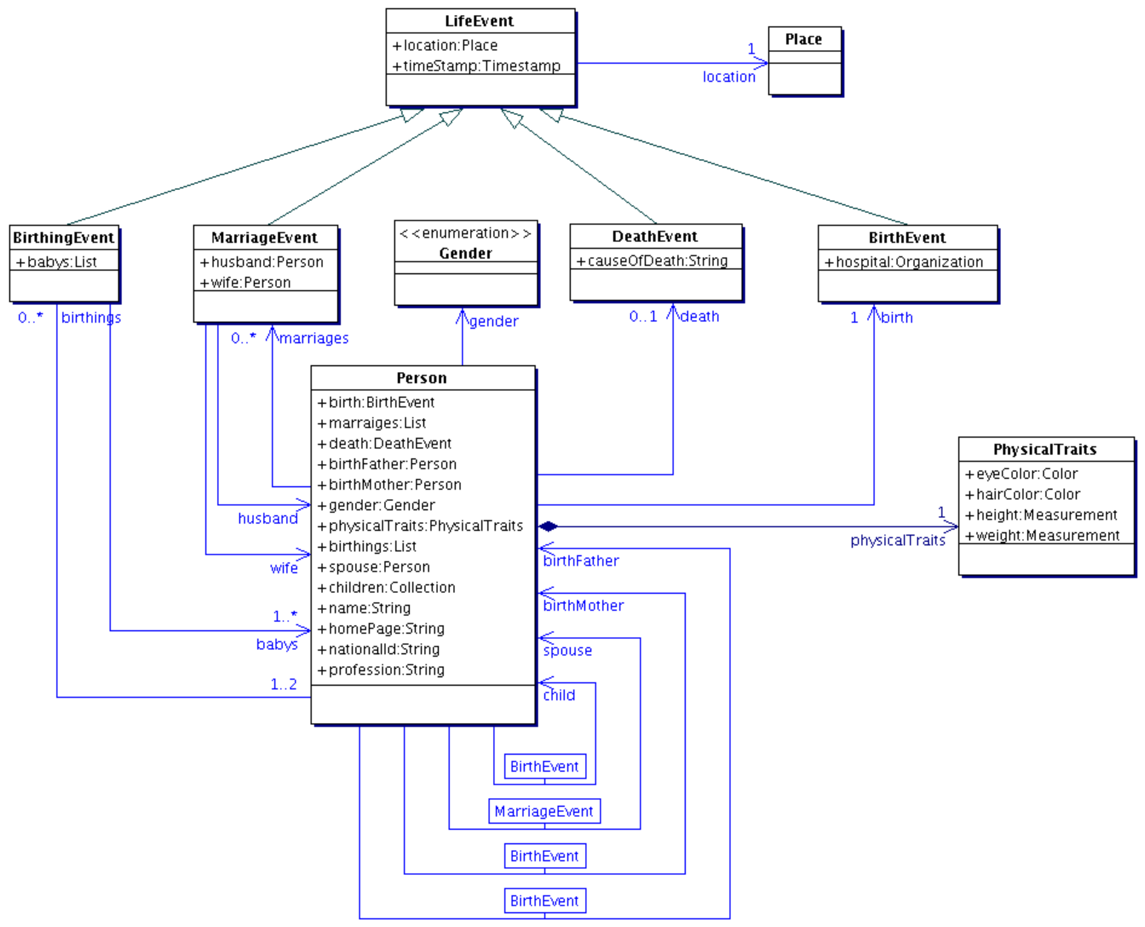


Figure 1: Person Information Model: A Sample Domain Specific Model

Figure 1 shows the UML Class diagram for the Person Information Model. The model shows that:

1. A Person has several LifeEvents:
    - o BirthEvent: Marks the birth of the associated Person
    - o MarriageEvent: Marks a marriage of the associated Person
    - o BirthingEvent: Marks a delivery of one or more babies where the associated person is a parent.
    - o 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.

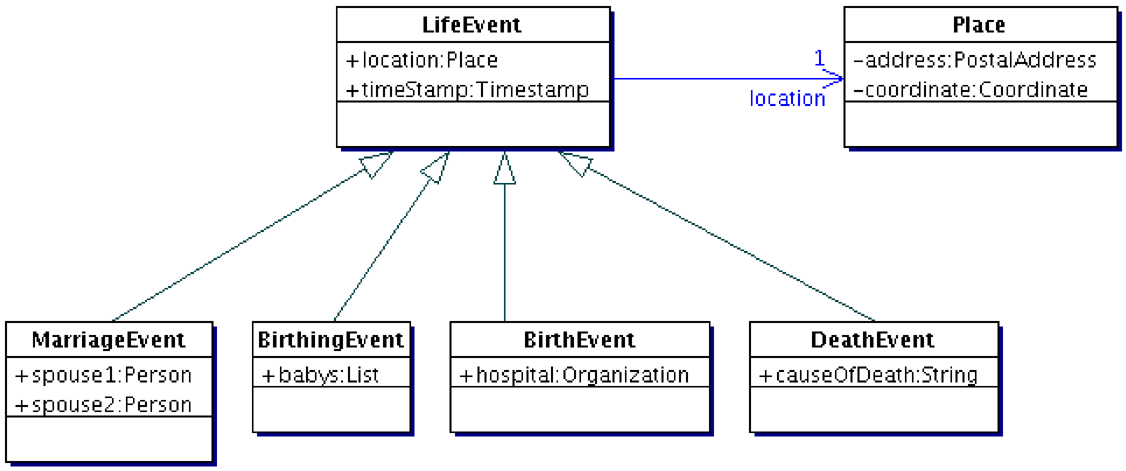


Figure 2: Person Information Model: Inheritance View

Figure 2 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.

2.3 Overview of ebXML Registry Information Model

This section summarizes the ebXML Registry Information Model [ebRIM]. This model is the target of the mapping defined in this document. The reader SHOULD read [CMRR] for a more detailed overview of ebXML Registry as a whole

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 3 presents the UML class diagram representing the Registry Information Model. Figure 4, shows the inheritance relationships in among the classes of the ebXML Registry Information Model.

Figure 4: ebXML Registry Information Model, Inheritance View

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

2.3.1 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.

2.3.2 Object Identification

A RegistryObject has a globally unique id which is a UUID based URN:

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

Listing 1: Example of 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 MAY potentially be human friendly:

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

Listing 2: 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 3: Example of ExternalIdentifier

2.3.3 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 acrossRegistryObjects.

```
<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 4: Example of Name and Description

2.3.4 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.

2.3.4.1 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>
          <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 5: Example of a Dynamic Attribute Using Slot

2.3.5 Object Classification

Any RegistryObject may be classified using any number of Classification instance. 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 6: Example of Object Classification

2.3.6 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).

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

Listing 7: Example of Object Association

2.3.7 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 8: Example of Reference to Web Content Using ExternalLink

2.3.8 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 9: Example of Object Packaging Using RegistryPackages

2.3.9 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.

3 Mapping a Domain Specific Model to ebRIM

This chapter identifies several common mapping patterns that are encountered when a domain specific information model is mapped to [ebRIM]. For each such pattern we define a consistent heuristic or algorithm to perform the mapping. The goal is to make it easier for domain experts to utilize the ebXML Registry for their domain and to have consistency across all domain-specific uses of ebXML Registry.

A source model may be in many different formats such as Java, XML, SQL and so on.

[UML] is a standard for information model description and therefore this document assumes the source information model is described in UML. [UML] terminology and notation is consistently used throughout this chapter and this document.

It should be understood that the mappings produced by applying the heuristics and algorithms described in this document will be only as good as the input UML model (this is the old garbage-in, garbage-out principal). A person applying these mapping patterns (the mapper) MAY choose to deviate from these patterns to compensate for special situations in the input UML model. Any mapping pattern not covered by this document MAY be addressed in an ad hoc manner by the mapping. Suggestions for improvements to the mapping should be sent to the Editors listed on the title page of this document.

3.1 Class Mapping

This section defines how a class in the source model is mapped to a class in [ebRIM]. Mapping of attributes of the source class will be discussed in section 3.6.

A class in the source model is mapped to [ebRIM] using the following algorithm:

1. **Direct Class Mapping To Rim:** First determine if there is a class in ebRIM that closely matches the class in the source model. For example the Person class in PIM matches closely to the Person class in [ebRIM]. Thus it is preferred that the Person class in PIM is mapped to the Person class in [ebRIM].
2. **Mapping To ExtrinsicObject Sub-Class:** If no class in [ebRIM] is a good match then define a new sub-class of ExtrinsicObject class in [ebRIM] and map the source class to the new sub-class. See section 3.1.1 on how to define a new sub-class of ExtrinsicObject. For example the various LifeEvent classes in PIM SHOULD be mapped to sub-classes of ExtrinsicObject where the class names match the various LifeEvent class names.

3.1.1 Defining a Sub-Class of ExtrinsicObject

This section provides the steps to define a new sub-class of ExtrinsicObject class.

To define a sub-class of ExtrinsicObject you MUST extend the canonical ObjectType ClassificationScheme and add a new ClassificationNode as a child or descendent of the canonical ClassificationNode for ExtrinsicObject in the ObjectType ClassificationScheme.

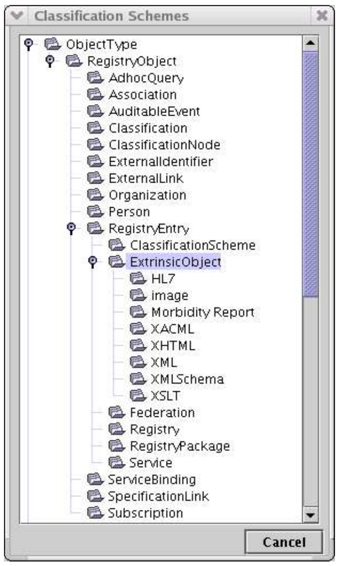
For example to extend the ObjectType ClassificationScheme for the LifeEvent classes in PIM the following ClassificationNode hierarchy MUST be submitted to the ebXML Registry via a SubmitObjectsRequest.

Note that:

- The id attribute values SHOULD have actual id values. See 9 for generating unique 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>
```

Listing 10: Example of Adding LifeEvent Classes to ObjectType ClassificationScheme



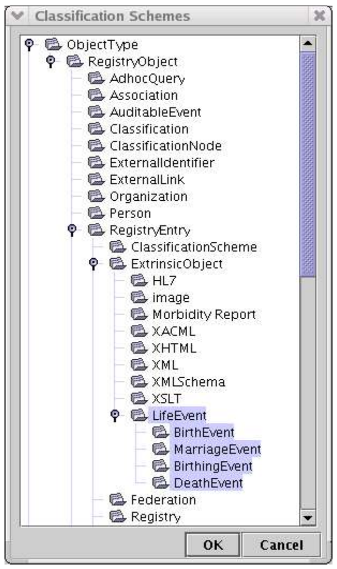


Figure 5: ObjectType ClassificationScheme: Before and After Extension for LifeEvent

3.2 Interface Mapping

Interfaces are classes that only have methods and have no attributes (they may contain constant attributes). They should be mapped in a manner similar to Class mapping. The only difference is that Interface methods that follow the getter method design pattern MAY be mapped to corresponding attributes.

For example, if the Person class in PIM model was an interface that had a method called getAge(), then that method MAY be mapped to an age attribute in the corresponding [ebRIM] class.

3.3 Inheritance Mapping

A class in the source model may have a generalization or inheritance relationship with another class in the model. For example, the BirthEvent, MarriageEvent, BirthingEvent and DeathEvent classes have an inheritance relationship with the LifeEvent class in PIM.

Such inheritance relationships SHOULD be reflected in the mapping to [ebRIM] by defining a corresponding inheritance relationship among the ClassificationNodes defined when extending the ObjectType scheme. This has already been illustrated in section 3.1.1 and Figure 5.

3.3.1 Mapping of Multiple Inheritance

A special case is “multiple inheritance” where the source model has multiple base classes for the same derived class. There is no direct support for multiple inheritance in [ebRIM]. In case the source model has a derived class with multiple base classes, the mapping SHOULD choose one base class to map as the base ClassificationNode in the ObjectType ClassificationScheme. The remaining base classes SHOULD be mapped as ClassificationNodes in the ObjectType ClassificationScheme and should be associated with the derived class using an Association whose associationType is the id for the canonical ClassificationNode “Extends” or “Implements” within the canonical AssociationType ClassificationScheme.

3.4 Method Mapping:

There is no support for mapping methods from a source model to [ebRIM]. Methods that follow a getter method MAY be mapped to an attribute as defined in section 3.3.

3.5 Association Mapping

A UML Association in the source model SHOULD be mapped to an [ebRIM] Association.

3.5.1 Navigability / Direction Mapping

Associations in UML MAY be directed or undirected. Associations in [ebRIM] are always implicitly directed from the sourceObject to the targetObject of an Association.

Directed UML associations MUST map the Class at the arrowhead end as targetObject and the Class at the other as sourceObject. In case of Undirected UML associations the mapper MAY specify the mapping of the Classes at each end to sourceObject or targetObject using their best judgement.

3.5.2 Role Name / Association Name Mapping

UML defines for an association, an association name as well as two role names (one for each end of the association).

The role name in the UML mapping at the targetObject end of the association, if present, SHOULD be mapped to the associationType. If the role name at the targetObject end (target role name) is not present then the association name SHOULD be mapped to the associationType.

In addition, the target role name (or UML association name) MAY also be mapped to the Association name in ebRIM.

1.1.1.1 Defining a New Association Type

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

To define a Association Type you 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 5 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="${SPOUSE_NODE_ID}"
     parent="urn:uuid:6902675f-2f18-44b8-888b-
c91db8b96b4d">
  <rim:Name>
    <rim:LocalizedString charset="UTF-8"
value="Spouse"/>
  </rim:Name>
  <rim:ClassificationNode code="Husband"
     id="${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 11: Example of Adding Spouse Association Types

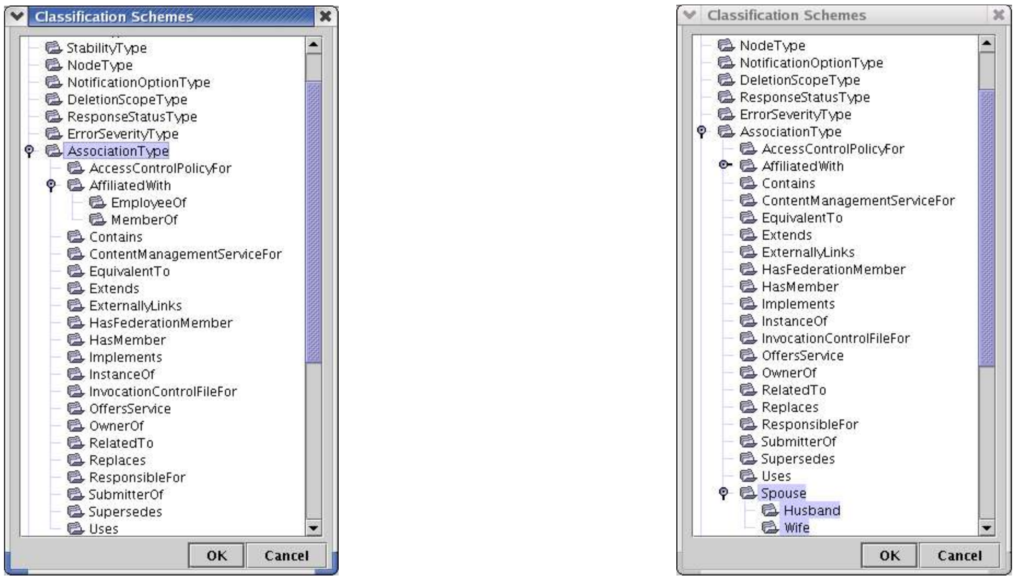


Figure 6: ObjectType ClassificationScheme: Before and After Extension For Spouse

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.

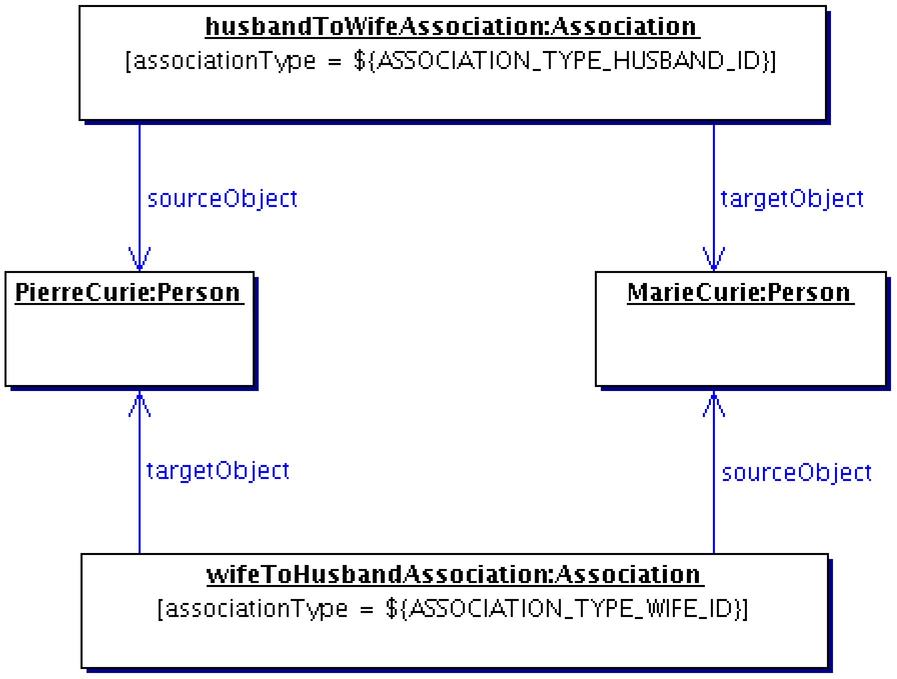

husbandToWifeAssociation:Association
[associationType = ${ASSOCIATION_TYPE_HUSBAND_ID}]
sourceObject
targetObject
PierreCurie:Person
MarieCurie:Person
targetObject
sourceObject
wifeToHusbandAssociation:Association
[associationType = ${ASSOCIATION_TYPE_WIFE_ID}]

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

3.5.4 Aggregation Mapping

A UML Aggregation maps to multiple [ebRIM] Associations in a manner consistent with earlier sections.

Give example here later??

3.5.5 Composition Mapping

When a UML Class (Container) wholly contains another class (Contained) then the UML Association between the two is called a UML Composition. The Composition Association is denoted with a filled diamond at the source end of the Association.

An example of composition in PIM is where the Person class is the container while the PhysicalTraits class is the contained class.

A composition association in UML is mapped [ebRIM] as follow:

1. The container class and the contained class map to [ebRIM] as defined by section 3.1.
2. The composition Association maps to a Slot instance that is defined for the container RegistryObject.
3. The composition Slot MUST have as the value of its “name” attribute,
    a. The target role name from the UML Association, or if that is not present
    b. The name of the UML Association
4. The composition Slot MUST have as the value of its “slotType” attribute, the logical lid of the canonical DataType “ObjectRef”. This value is:

    `urn:oasis:names:tc:ebxml- regrep:DataType:ObjectRef`
5. The composition Slot MUST have as the value of its “values” attribute, a list of String where each String MUST be the value of the id attribute of an object that is composed or contained by

the container RegistryObject

Note that the ebXML Registry does not enforce the semantics of composition Associations. Specifically, deleting a container object does not automatically delete contained objects.

The following example shows how the composition association between a Person instance and a PhysicalTraits instance in PIM maps to [ebRIM].

```
<--The ExtrinsicObject of objectType Person for Person
PierreCurie -->
<rim:ExtrinsicObject id="${PIERRECURIE_PERSON_ID}"
mimeType="text/xml"
      objectType="${OBJECT_TYPE_PERSON_ID}">
      <rim:Slot name="physicalTraits"
            slotType="urn:oasis:names:tc:ebxml-
regrep:DataType:ObjectRef ">
            <rim:ValueList>
                  <rim:Value>$
{PIERRECURIE_PHYSICAL_TRAITS_ID}</rim:Value>
            </rim:ValueList>
      </rim:Slot>
      …
</rim:ExtrinsicObject>

<--The ExtrinsicObject of objectType PhysicalTraits for
Person PierreCurie -->
<rim:ExtrinsicObject id="${PIERRECURIE_PHYS_TRAITS_ID}"
mimeType="text/xml"
            objectType="${OBJECT_TYPE_PHYS_TRAITS_ID}">
            …
</rim:ExtrinsicObject>
```

Listing 12: Example of Composition of PhsyicalTraits Instance Within Person Instance

3.5.6 N-ary Association Mapping

UML N-ary associations involving three or more Classes is not commonly used and is not covered by this document in detail. It is suggested that RegistryPackage may be considered as a mapping for such n-ary Associations.

3.5.7 XOR Associations

XOR Associations as defined by UML are not commonly used in source models. XOR Associations may be mapped to [ebRIM] Associations and it MUST be the responsibility of the mapping to enforce the XOR constraints in an application specific manner.

3.6 Attribute Mapping

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 3.1.

Figure 8 provides the flowchart for the algorithm that SHOULD be used to map attributes from the source model to [ebRIM]. Each box in right column maps to a section later in the document that describes the mapping in detail.

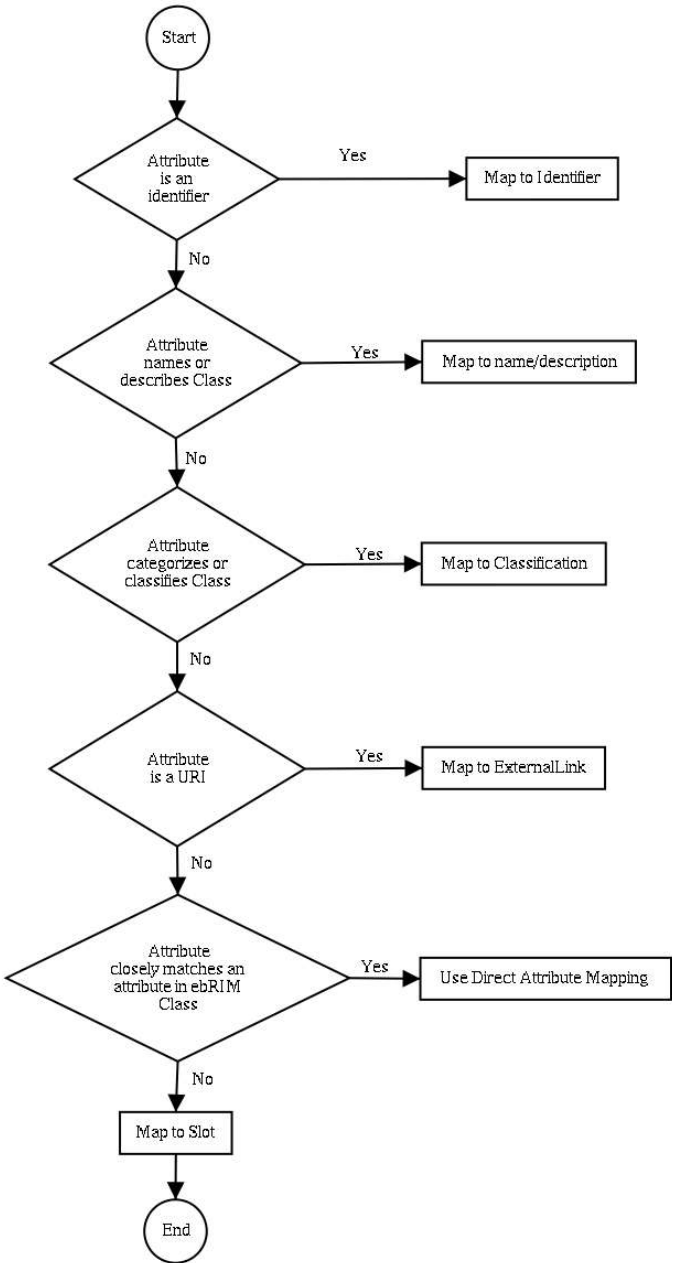

Start
Attribute is an identifier
Yes
Map to Identifier
No
Attribute names or describes Class
Yes
Map to name/description
No
Attribute categorizes or classifies Class
Yes
Map to Classification
No
Attribute is a URI
Yes
Map to ExternalLink
No
Attribute closely matches an attribute in ebRIM Class
Yes
Use Direct Attribute Mapping
No
Map to Slot
End

Figure 8: Attribute Mapping Algorithm Flowchart

3.6.1 Mapping to Identifier

Section 2.3.2 describes the various ways that a RegistryObject may be identified in [ebRIM].

3.6.1.1 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 13: 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] class. 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 class then it MUST not be mapped to the id attribute. Instead it SHOULD be mapped to the Logical id (lid) attribute as defined next.

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

3.6.1.2 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:

- SHOULD 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.

3.6.1.3 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 4.2.

```
<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 14: Example of Mapping to ExternalIdentifier

3.6.2 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 datatype 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 15: 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.

3.6.3 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 2.3.6.

For example, the rim.Person.gender attribute is of datatype 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 16: 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.

3.6.4 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 2.3.7.

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 hown 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 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 17: Example of Mapping to ExternalLink

3.6.5 Direct Mapping to ebRIM Attribute

In some cases an attribute in the source model class class may closely match an attribute in the [ebRIM] class. 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] (refered 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 datatype String while the rim.Person.telephoneNumbers attribute is of datatype TelephoneNumber where TelephoneName consists of several String attributes:

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

Thus the mapper MUST choose which rim. TelephoneNumber attribute the pim.Person.phone attribute maps to. 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.

3.6.6 Mapping to Slot

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

3.6.6.1 Mapping to rim.Slot.slotName

The source attribute name SHOULD be mapped to the rim.Slot.slotName attribute. To prevent name conflicts the mapping SHOULD 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 18: Example of Mapping pim.Person.Profession to slotName

3.6.6.2 Mapping to rim.Slot.slotType

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

For example, the datatype 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 19: Example of Mapping DataType to slotType

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

3.6.6.3 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 20: Example of Mapping Attribute value to Value

3.7 Enumerated Type Mapping

A source attribute whose datatype 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 3.6.3.

4 Using ClassificationSchemes

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 called a ClassificationScheme. The allowed values in a ClassificationScheme are represented by ClassificationNode instances within ebRIM.

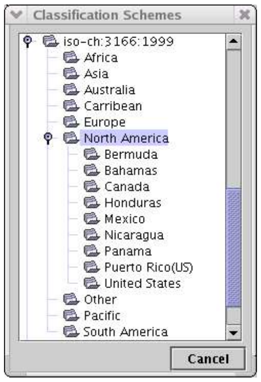


Figure 9: Geography ClassificationScheme Example

Figure 9 shows a geography ClassificationScheme. It is a hierarchical tree structure where the root of the tree "iso-ch:3166:1999" is the name of the ClassificationScheme while the rest of the nodes in the tree are ClassificationNodes.

Note that most ebXML Registry implementations [IMPL] provide a GUI tool to create and manage ClassificationSchemes graphically.

4.1 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 datatypes supported by an registry (DataType scheme).
- Used to define the Classes supported by a registry (ObjectType scheme).
- Used to define the association types supported by the registry (AssociationType scheme).
- 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).

4.2 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.

4.3 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.

4.3.1 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 10 shows an example of extending the ObjectType scheme.
- Extending the AssociationType scheme to define the association types supported by the registry. Listing 11 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.

4.4 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.

4.4.1 Use Cases for Defining New ClassificationSchemes

4.5

5 Defining Content Management Services

5.1 Defining Content Validation Services

Use of jCAM to validate XML instance docs?

5.2 Defining Content Cataloging Services

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

6 Defining Domain Specific Queries

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

6.1 Identifying Common Discovery Use Cases

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.

For the Person Information model we identify the following sample domain specific discovery use cases as likely to be commonly needed:

- Find Persons by:
  - Name
  - Gender
  - Age
  - #of Children
  - Physical trait
  - #of marriages
  - Married to specified person
  - Parent of specified person
  - Child of specified person
  - Ancestor of specified person
  - Descendent of specified person

6.1.1

7 Using the Event Notification Feature

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 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 sprocesses 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 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”.

7.2 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:QuerySt
ring>
    </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/no
tificationListener"
   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>
</SubmitObjectsRequest>
```

Listing 21: Example of Defining a Subscription for DeathEvent

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

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 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 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.

8 Defining Access Control

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.

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. SubjectRole scheme may be extended like any other scheme as defined in section 4.3.

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. SubjectGroup scheme may be extended like any other scheme as defined in section 4.3.

8.2.1 Defining Custom Access Control Policies

8.3

9 Known Issues

These generic mapping patterns should be formalized via RIM artifacts and stored in the registry.

- UML cardinality needs to be expressed generically, like for Slots, Associations, …
- Expanding RIM ObjectType hierarchy beyond ExtrinsicObject subtree
- Objective criteria for when to use ObjectRefs vs. Values, like "NameAsRole" could refer to something like RoleTaxonomy instead of using value of UML role.
- Aggregation and Composition are Association in UML. There mapping to ebRIM is inconsistent.
- Need to give example of mapping an Association class (e.g. MarriageEvent)

Appendix A - PIM to ebRIM: The Complete Mapping

Appendix B - Tips and Tricks

Appendix C - Generating Unique UUIDs

Appendix D - Assigning Logical Id

Appendix E - Organizing Object in RegistryPackages

1116 ## Appendix F - Revision History

Rev	**Date**	**By Whom**	**What**
0.1	September 22, 2004	Farrukh Najmi, Nikola Stojanovic	Initial version with core mapping pattern for input from CCTS mappers.
0.2	September 23, 2004	Farrukh Najmi, Nikola Stojanovic	Minor bug fixes.
0.3	September 24, 2004	Farrukh Najmi, Nikola Stojanovic	Added some content to chapters 4-8.
0.3	September 29, 2004	Farrukh Najmi, Nikola Stojanovic	Minor fixes based upon feedback from initial reviewers.
0.5	Avril 15, 2005	Ivan Bedini	Updated to version [ebRIM] v3.0 Changed file format

1117

Appendix G - References

Appendix H - Normative

[ebRIM] ebXML Registry Information Model version 3.0
http://www.oasis-open.org/committees/regrep/documents/3.0/specs/ebRIM.pdf

[ebRS] ebXML Registry Services Specification version 3.0
http://www.oasisopen.org/committees/regrep/documents/3.0/specs/ebRS.pdf

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

Appendix I Informative

[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
Need other implementations listed here??

[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