ebXML Registry profile for Web Services

Version 1.0 Draft 3

Draft OASIS Profile, 21 September, 2005

Document identifier:

regrep-ws-profile-1.0

Location:

http://www.oasis-open.org/committees/regrep/documents/profile/regrep-ws-profile-1.0-draft-1.pdf

Editors:

Name	Affiliation
Farrukh Najmi	Sun Microsystems
Joseph Chiusano	Booz Allen Hamilton

Contributors:

Name	Affiliation
Paul Sterk	Sun Microsystems
Tony Graham	Sun Microsystems
Nikola Stojanovic	RosettaNet

Abstract:

This document defines the ebXML Registry profile for publish, management, governance discovery and reuse of Web Service artifacts.

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/).

1 Table of Contents

Illustration Index

199

Index of Tables

1 Introduction

This chapter provides an introduction to the rest of this document.

This document normatively defines the ebXML Registry profile for publish, management, governance discovery and reuse of Web Service artifacts. It defines standard extensions and restrictions of the features of ebXML Registry standard specialized for Web Services atrifacts.

The document is organized as follows:

- Chapter 1 provides an introduction to the rest of this document.
- Chapter 2 provides an overview of the Web Services information model.
- Chapter 3 provides an overview of the ebXML Registry standard.
- Chapter 4 specifies the mapping between WSDL information model and ebXML Registry information model.
- Chapter 5 specifies the profile for supporting the publishing of Web Services artifacts.
- Chapter 6 specifies the profile for supporting the validation of Web Services artifacts using business rules.
- Chapter 7 specifies the profile for supporting the cataloging of Web Services artifacts.
- Chapter 8 specifies the profile for the discovery of Web Services artifacts.
- Chapter 9 specifies the profile for subscription to and notification of events related to Web Services artifacts.
- Chapter 10 specifies the profile for securing access to Web Services artifacts.
- Chapter 11 specifies the definition of canonical metadata defined by this profile.
- Chapter 12 provides normative and informative references that are used within or relevant to this document.

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 IETF RFC 2119 [RFC2119].

The term *"repository item"* is used to refer to content (e.g., an XML document or a DTD) that resides in a repository for storage and safekeeping. Each repository item is described by a RegistryObject instance. The RegistryObject catalogs the RepositoryItem with metadata.

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 WSDL Information Model Overview

This chapter provides an overview of the source information model for web services description within an ebXML Registry. For more information see [WSDL-OVW] and [WSDL].

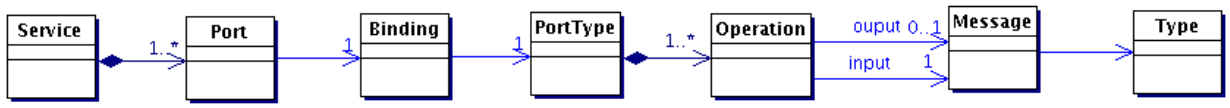

Illustration 1: WSDL Information Model

The WSDL information model is a layered model. At the lowest level is the abstract PortType The Binding is the next level and its provides a protocol binding for the abstract PortType. The concrete Port is the next level which provides an actual implementation of the abstract PortType within the protocol binding specified by the Binding. Finally, the Service encapsulated one more Ports to provide an implementation of a web service complete with all its concrete protocol specific interfaces.

2.1 Element <service>

A WSDL descripton contains one or more service elements that describe a web service. A service element contains one or more port elements which define concrete interfaces exposed by the service.

2.2 Element <port>

Each port element contains information necessary to invoke the concrete service interface described by the port (typically a URL end-point). Each port element contains a reference to a binding element.

2.3 Element <binding>

The binding element describes the binding of the service interface to a specific on-the-wire protocol (typically SOAP). A binding element contains a reference to a portType element.

2.4 Element <portType>

A portType element describes an abstract service interface. A portType element contains definition of one or more operation elements.

2.5 Element <operation>

An operation element defines an operation method supported by the parent portType. It contains 1 input message (sent as request to server) and 0 or 1 output messages (sent as response by server) supported by the operation.

2.6 Element <message>

A message element decsribes a message that is communicated by an operational method supported by the abstract service interface described by the parent portType. A message may reference data types defined within XML Schema imported in the types element.

2.7 Element <types>

The types element describes the data types used by messages excanged between the client of the web service and the server implementing the web service. This element typically is used to import XML Schema type definitions for use within the WSDL description.

3 ebXML Registry Overview

This chapter provides an overview of ebXML Registry Information Model [ebRIM] and an overview of the specific domain and/or application.

The [ebRIM] is the target for the mapping patterns defined by this document and.

The specific domain is the source information model.

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

3.1 Overview of [ebRIM]

This section is provided in the « Deployment Profile Template for ebXML V3 specs » and can be removed in a specific profile.

Normally only specifics topics needs to be developed here (but the profile editor can prefer to leave it)

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

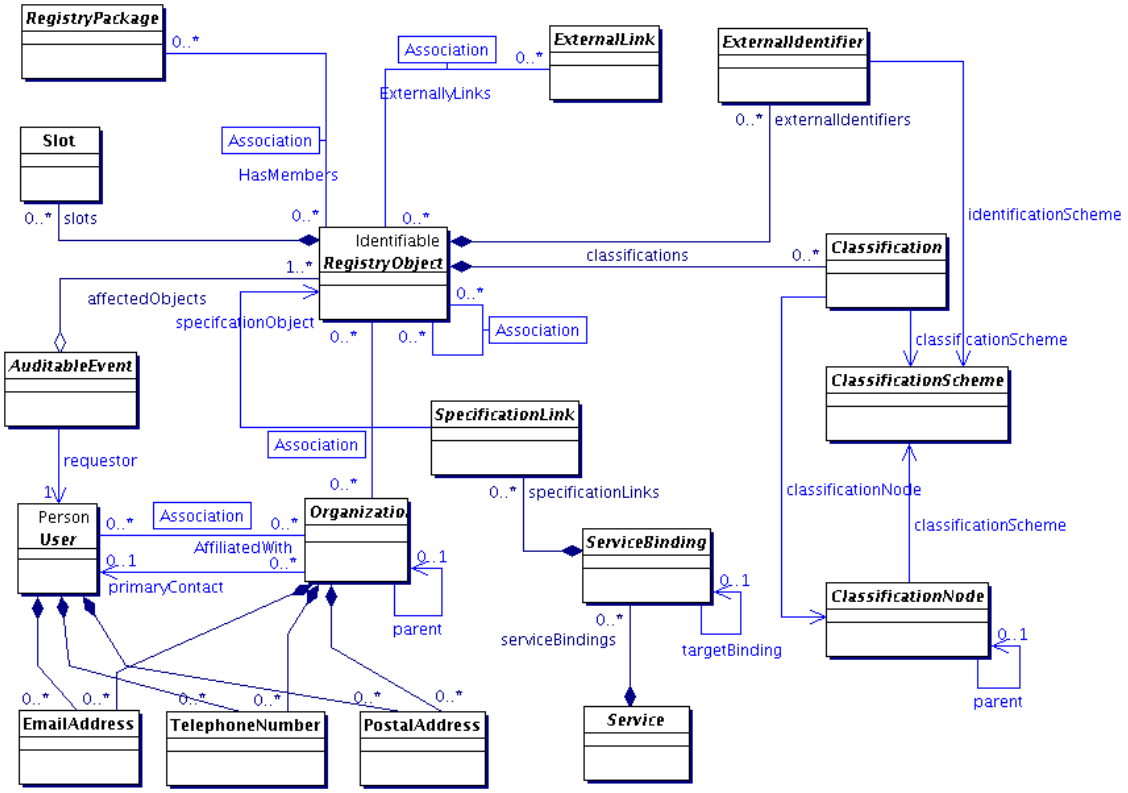

Figure 1: ebXML Registry Information Model, High Level Public View

The ebXML registry defines a Registry Information Model [ebRIM] that specifies the standard metadata that may be submitted to the registry. Figure 1 presents the UML class diagram representing the Registry Information Model. Figure 2, shows the inheritance relationships in among the classes of the ebXML Registry Information Model.

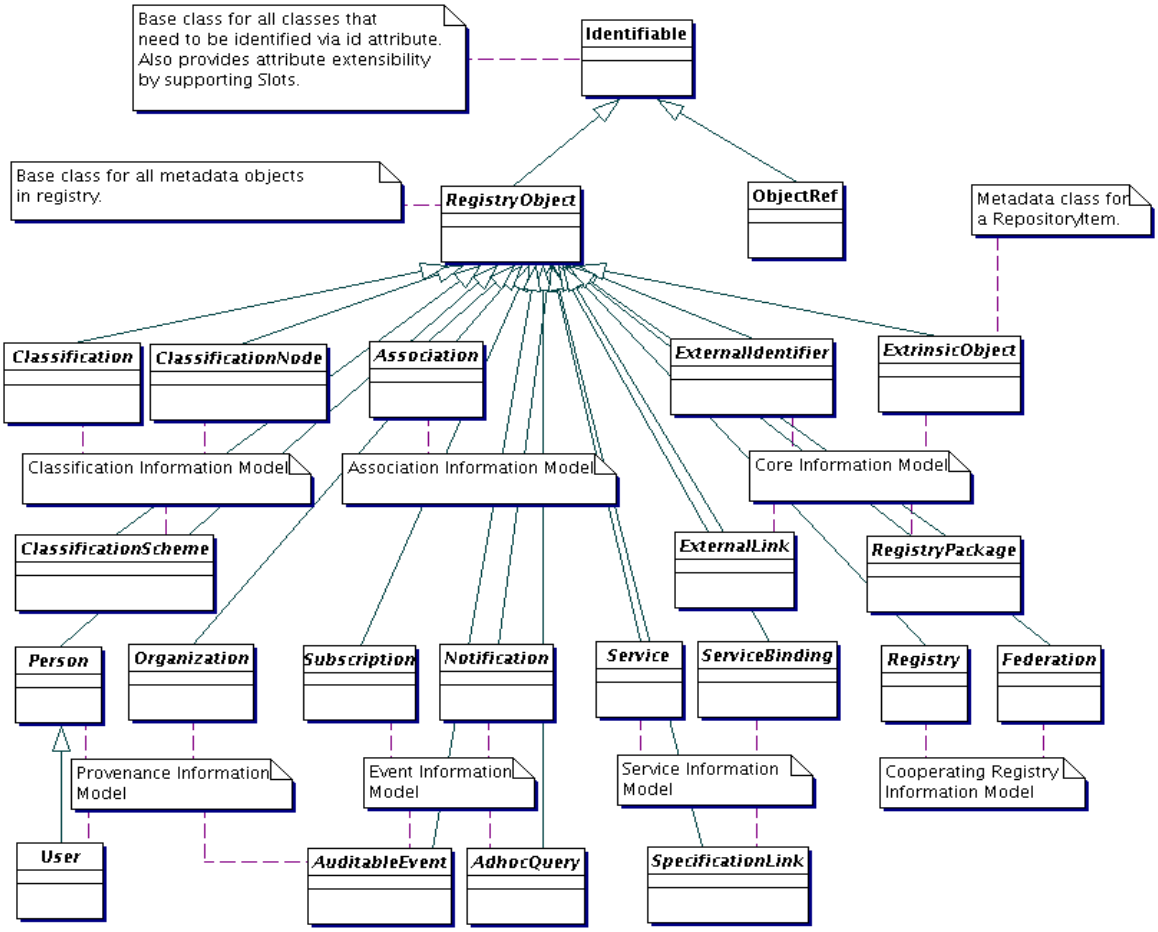

Figure 2: ebXML Registry Information Model, Inheritance View

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

3.1.1 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.

3.1.2 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

The id attribute value MAY potentially be human friendly.

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

Listing 2: 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:

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

Listing 3: 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 4: Example of ExternalIdentifier

3.1.3 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 5: Example of Name and Description

3.1.4 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.

3.1.4.1 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 6: Example of a Dynamic Attribute Using Slot

3.1.5 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 7: Example of Object Classification

3.1.6 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 8: Example of Object Association

3.1.7 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 9: Example of Reference to Web Content Using ExternalLink

3.1.8 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 10: Example of Object Packaging Using RegistryPackages

3.1.9 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.

3.2 Overview of [ebRS]

The [ebRS] specification defines the interfaces supported by an ebXML Registry and their bindings to protocols such as SOAP and HTTP.

4 Mapping WSDL Information Model to [ebRIM]

This chapter provides an overview of the mapping between the WSDL information model and [ebRIM].

The following figures provide a pictorial overview of the type mapping between the two models. Following both figures from left to right, there is a one-to-one correspondence between the two models. The mapping of types between the two models stops at the WSDL PortType.

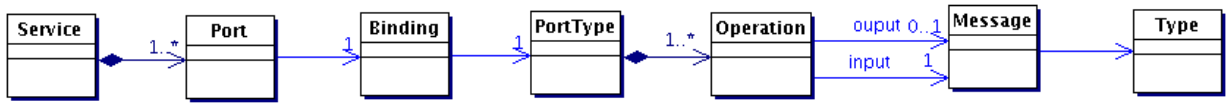

Illustration 2: WSDL Information Model

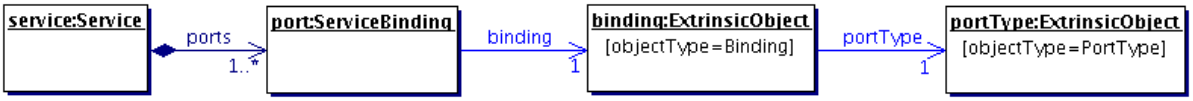

Illustration 3: Mapping of WSDL Information Model to ebRIM

It is important to note that although the mapping described in this section is complex, this complexity is hidden from the publisher of the WSDL document because the mapping is automatically created when WSDL is published to an ebXML Registry as described in chapter 7 on Cataloging Service Profile.

4.1 wsdl:service → rim:Service Mapping

A wsdl:service MUST be mapped to a rim:Service as described in this section.

```
  <service name="ebXMLRegistrySOAPService">
    <port binding="bindings:QueryManagerSOAPBinding"
name="QueryManagerPort">
      <soap:address location="http://your.server.com/soap"/>
    </port>
    <port binding="bindings:LifeCycleManagerSOAPBinding"
name="LifeCycleManagerPort">
      <soap:address location="http://your.server.com/soap"/>
    </port>
  </service>
```

Example wsdl:service

4.1.1 Attribute id

The id attribute value of the rim:Service MUST have as prefix the targetNamespace for the wsdl:service element, followed by a suffix of ":service:<service name>" where <service name> MUST be the value of the name attribute of the wsdl:service element.

```
targetNamespace: urn:acmeinc:ebxml:registry:3.0:services:wsdl

WSDL fragment:
<wsdl:service name="ebXMLRegistrySOAPService">

<rim:Service
id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:service:
ebXMLRegistrySOAPService">
```

Example of rim:Service id Attribute Mapping

4.1.2 Element Name

The name element of the rim:Service MUST be set according to the value of the name attribute within the wsdl:service element. The locale and charset of the name attribute in the rim:Service MUST be unspecified since it is unspecified within the WSDL.

```
WSDL fragment:
<wsdl:service name="ebXMLRegistrySOAPService">

<rim:Service
id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:Service:
ebXMLRegistrySOAPService">
  <rim:Name>
    <rim:LocalizedString value="ebXMLRegistrySOAPService"/>
  </rim:Name>
</rim:Service>
```

Example of rim:Service name Attribute Mapping

4.1.3 Element Description

The description element of the rim:Service MUST be set according to the content of the wsdl:documentation element, if specified, within the wsdl:service element. The locale attribute of the LocalizedString in description element MUST be set to the xml:lang attribute of the wsdl:documentation element if it is specified.

```
WSDL fragment:
<wsdl:service name="ebXMLRegistrySOAPService">
  <wsdl:documentation>
  An implementation of ebXML Registry 3.0
  <wsdl:documentation>
</wsdl:service>

<rim:Service
id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:Service:
ebXMLRegistrySOAPService">
  <rim:Description>
    <rim:LocalizedString value="An implementation of ebXML Registry
3.0"/>
  </rim:Description>
</rim:Service>
```

Example of rim:Service description Attribute Mapping

4.1.4 Elements Classification

The rim:Service for a wsdl:service contains the following composed Classification instances. The ClassificationSchemes are defined in chapter 11.

ClassificationScheme	Description	Required
ObjectType	Classifies the rim:Service for wsdl:service by the Service ClassificationNode child of the WSDL ClassificationNode in the ObjectType ClassificationScheme. This identifies the rim:Service as a wsdl:service instance.	Yes

```
<rim:Service
id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:Service:
ebXMLRegistrySOAPService">
```

```
  <Classification classificationScheme="urn:oasis:names:tc:ebxml-
regrep:ObjectType" classificationNode="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Service"/>

</rim:Service>
```

Example of rim:Service classifications Attribute Mapping for wsdl:service

4.1.5 Elements ServiceBinding

The rim:Service MUST contain a Collection of composed rim:ServiceBinding instances to represent the wsdl:port instances as described in the section on wsdl:port to rim:ServiceBinding mapping.

```
<rim:Service
id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:Service:
ebXMLRegistrySOAPService">
  <rim:ServiceBinding .../>
  <rim:ServiceBinding .../>
</rim:Service>
```

Example of wsdl:port → rim:ServiceBinding Mapping

4.2 wsdl:port → rim:ServiceBinding Mapping

A wsdl:port MUST be mapped to a rim:ServiceBinding as described in this section.

4.2.1 Attribute id

The id attribute value of the rim:ServiceBinding MUST have as prefix the targetNamespace for the the wsdl:port element, followed by a suffix of ":port:<port name>" where <port name> MUST be the value of the name attribute of the wsdl:port element.

```
<wsdl:port binding="bindings:QueryManagerSOAPBinding
  name="QueryManagerPort">

<rim:ServiceBinding
  id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:port:
QueryManagerPort">
```

Example of rim:ServiceBinding id Attribute Mapping for wsdl:port

4.2.2 Element Name

The name element of the rim:ServiceBinding MUST be set according to the value of the name attribute within the wsdl:port element. The locale and charset of the LocalizedString for the name element in the rim:Service MUST be unspecified since it is unspecified within WSDL.

```
<wsdl:port binding="bindings:QueryManagerSOAPBinding
  name="QueryManagerPort">

<rim:ServiceBinding
  id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:port:
QueryManagerPort">
  <rim:Name>
    <rim:LocalizedString value="QueryManagerPort"/>
  </rim:Name>
</rim:ServiceBinding>
```

Example of rim:ServiceBinding name Attribute Mapping for wsdl:port

4.2.3 Element Description

The description element of the rim:ServiceBinding MUST be set according to the content of the wsdl:documentation element within the wsdl:port element, if specified. The locale attribute of the LocalizedString in description element MUST be set to the xml:lang attribute of the wsdl:documentation element if it is specified.

```
<wsdl:port binding="bindings:QueryManagerSOAPBinding
  name="QueryManagerPort">
  <wsdl:documentation>
  SOAP Binding implementation of ebXML Registry QueryManager
  <wsdl:documentation>
</wsdl:port>

<rim:ServiceBinding
  id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:port:
QueryManagerPort">
  <rim:Description>
    <rim:LocalizedString value="SOAP Binding implementation of ebXML
Registry QueryManager"/>
  </rim:Description>
</rim:ServiceBinding>
```

Example of rim:ServiceBinding description Attribute Mapping for wsdl:port

4.2.4 Attribute accessURI

The accessURI attribute value of the rim:ServiceBinding MUST be set to the endpoint URI within the protocol specific element within the wsdl:port element that provides the endpoint address. In case of SOAP binding this MUST be specified in the soap:address element.

```
<wsdl:port binding="bindings:QueryManagerSOAPBinding
  name="QueryManagerPort">
  <soap:address location="http://your.server.com/soap"/>
</wsdl:port>

<rim:ServiceBinding
  id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:port:
QueryManagerPort" accessURI="http://your.server.com/soap">
</rim:ServiceBinding>
```

Example of rim:ServiceBinding accessURI Attribute Mapping for wsdl:port

4.2.5 Attribute service

The service attribute value of the rim:ServiceBinding must contain the value of the id attribute of the parent rim:Service that represents the parent wsdl:service.

4.2.6 Element Classification

The Classification elements of the rim:ServiceBinding MUST contain the following composed Classification instances. The ClassificationSchemes are defined in chapter 11.

ClassificationScheme	Description	Required
ObjectType	Classifies the rim:ServiceBinding for wsdl:port by the Port ClassificationNode child of the WSDL ClassificationNode in the ObjectType ClassificationScheme. This identifies the rim:ServiceBinding as a wsdl:port instance.	Yes

```
<wsdl:port binding="bindings:QueryManagerSOAPBinding
  name="QueryManagerPort">
  <soap:address location="http://your.server.com/soap"/>
</wsdl:port>

<rim:ServiceBinding
  id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:port:
QueryManagerPort" accessURI="http://your.server.com/soap">

  <Classification classificationScheme="urn:oasis:names:tc:ebxml-
regrep:ObjectType" classificationNode="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Port"/>

</rim:ServiceBinding>
```

Example of rim:ServiceBinding classifications Attribute Mapping for wsdl:port

4.3 wsdl:binding → rim:ExtrinsicObject Mapping

A wsdl:binding instance MUST be mapped to a rim:ExtrinsicObject instance as described in this section.

4.3.1 Attribute objectType

The objectType attribute value of the rim:ExtrinsicObject MUST be `urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding` which is the id of the Binding ClassificationNode child of the WSDL ClassificationNode described in chapter 11. This identifies the ExtrinsicObject to represent a wsdl:binding instance.

```
<rim:ExtrinsicObject
  objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding" ...>
```

Example of rim:ExtrinsicObject objectType Attribute Mapping for wsdl:binding

4.3.2 Attribute id

The id attribute value of the rim:ExtrinsicObject MUST have as prefix the targetNamespace for the the wsdl:binding element, followed by a suffix of ":binding:<binding name>" where <binding name> MUST be the value of the name attribute of the wsdl:binding element.

```
<binding name="QueryManagerSOAPBinding"
type="interfaces:QueryManagerPortType">

<rim:ExtrinsicObject
  id="urn:acmeinc:ebxml:registry:3.0:ExtrinsicObject:wsdl:binding:
QueryManagerSOAPBinding">
```

Example of rim:ExtrinsicObject id Attribute Mapping for wsdl:binding

4.3.3 Element Name

The name element of the rim:ExtrinsicObject MUST be set according to the value of the name attribute within the wsdl:binding element. The locale and charset of the LocalizedString for the name element MUST be unspecified since it is unspecified within WSDL.

```
<binding name="QueryManagerSOAPBinding"
type="interfaces:QueryManagerPortType">

<rim:ExtrinsicObject
```

```
  id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:binding:
QueryManagerSOAPBinding">
  <rim:Name>
    <rim:LocalizedString value="QueryManagerSOAPBinding"/>
  </rim:Name>
</rim:ExtrinsicObject>
```

Example of rim:ExtrinsicObject name element mapping for wsdl:binding

4.3.4 Element Description

The description element of the rim:ExtrinsicObject MUST be set according to the content of the wsdl:documentation element within the wsdl:binding element, if specified. The locale attribute of the LocalizedString in description element MUST be set to the xml:lang attribute of the wsdl:documentation element if it is specified.

```
<binding name="QueryManagerSOAPBinding"
type="interfaces:QueryManagerPortType">
<wsdl:port binding="bindings:QueryManagerSOAPBinding
  name="QueryManagerPort">
  <wsdl:documentation>
  SOAP Binding for ebXML Registry QueryManager
  <wsdl:documentation>
</wsdl:binding>

<rim:ExtrinsicObject
  <rim:Description>
    <rim:LocalizedString value="SOAP Binding for ebXML Registry
QueryManager"/>
  </rim:Description>
</rim:ExtrinsicObject>
```

Example of rim:ExtrinsicObject description element Mapping for wsdl:binding

4.3.5 Element Classification

The rim: ExtrinsicObject MUST contain the following composed Classification instances. The ClassificationSchemes are defined in chapter 11.

ClassificationScheme	Description	Required
ProtocolType	Classifies the rim:ExtrinsicObject for wsdl:binding by the type of protocol binding (e.g. SOAP) it supports.	Yes
TransportType	Classifies the rim:ExtrinsicObject for wsdl:binding by the type of transport binding (e.g. HTTP) it supports.	Yes
SOAPStyle	Classifies the rim:ExtrinsicObject for wsdl:binding by the type of SOAP style (e.g. Document) it supports.	Yes if ProtocolType is SOAP. No otherwise.

```
<binding name="QueryManagerSOAPBinding"
  type="interfaces:QueryManagerPortType">
  <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
...
</binding>

<rim:ExtrinsicObject
```

```
  <Classification classificationScheme="urn:oasis:names:tc:ebxml-
regrep:profile:ws:classificationScheme:ProtocolType"
classificationNode="urn:oasis:names:tc:ebxml-
regrep:profile:ws:ProtocolType:SOAP"/>

  <Classification classificationScheme="urn:oasis:names:tc:ebxml-
regrep:profile:ws:classificationScheme:TransportType"
classificationNode="urn:oasis:names:tc:ebxml-
regrep:profile:ws:TransportType:HTTP"/>

  <Classification classificationScheme="urn:oasis:names:tc:ebxml-
regrep:profile:ws:classificationScheme:SOAPStyle"
classificationNode="urn:oasis:names:tc:ebxml-
regrep:profile:ws:SOAPStyle:Document"/>

</rim:ExtrinsicObject>
```

Example of rim:ExtrinsicObject classifications element mapping for wsdl:binding

4.4 wsdl:portType → rim:ExtrinsicObject Mapping

A wsdl:portType instance MUST be mapped to a rim:ExtrinsicObject instance as described in this section.

4.4.1 Attribute objectType

The objectType attribute value of the rim:ExtrinsicObject MUST be `urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType` which is the id of the PortType ClassificationNode child of the WSDL ClassificationNode described in chapter 11. This identifies the ExtrinsicObject to represent a wsdl:portType instance.

```
<rim:ExtrinsicObject
  objectType="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType" ...>
```

Example of rim:ExtrinsicObject objectType Attribute Mapping for wsdl:portType

4.4.2 Attribute id

The id attribute value of the rim:ExtrinsicObject MUST have as prefix the targetNamespace of the wsdl:portType element, followed by a suffix of ":portType:<portType name>" where <portType name> MUST be the value of the name attribute of the <wsdl:portType> element.

```
TargetNameSpace= urn:oasis:names:tc:ebxml-
regrep:wsdl:registry:interfaces:3.0
<portType name="QueryManagerPortType"/>

<rim:ExtrinsicObject
  id="urn:oasis:names:tc:ebxml-
regrep:wsdl:registry:interfaces:3.0:portType:QueryManagerPortType" ...>
```

Example of rim:ExtrinsicObject id Attribute Mapping for wsdl:portType

4.4.3 Element Name

The name element of the rim:ExtrinsicObject MUST be set according to the value of the name attribute within the wsdl:portType element. The locale and charset of the name attribute in the rim:Service MUST be unspecified since it is unspecified within WSDL.

```
<wsdl:portType name="QueryManagerPortType">

<rim:ExtrinsicObject
```

```
  id="urn:oasis:names:tc:ebxml-
regrep:wsdl:registry:interfaces:3.0:portType:QueryManagerPortType">
  <rim:Name>
    <rim:LocalizedString value="QueryManagerPortType"/>
  </rim:Name>
</rim:ExtrinsicObject>
```

Example of rim:ExtrinsicObject name Attribute Mapping for wsdl:portType

4.4.4 Element Description

The description element of the rim: ExtrinsicObject MUST be set according to the content of the wsdl:documentation element within the wsdl:port element, if specified. The locale attribute of the LocalizedString in description element MUST be set to the xml:lang attribute of the wsdl:documentation element if it is specified.

```
<wsdl:portType name="QueryManagerPortType">
  <wsdl:documentation>
  portType for ebXML Registry QueryManager
  <wsdl:documentation>
</wsdl:portType>

<rim:ExtrinsicObject
  id="urn:oasis:names:tc:ebxml-
regrep:wsdl:registry:interfaces:3.0:portType:QueryManagerPortType">
  <rim:Description>
    <rim:LocalizedString value="portType for ebXML Registry
QueryManager"/>
  </rim:Description>
</rim:ExtrinsicObject>
```

Example of rim:ExtrinsicObject description Attribute Mapping for wsdl:portType

4.5 wsdl:port ↔wsdl:binding to rim:Association Mapping

This Association associates a wsdl:port to its wsdl:binding. It is specified as follows:

4.5.1 Attribute id

The id attribute value of the rim:Association MUST have the following pattern:

<id of rim:ServiceBinding for wsdl:port>:Implements:<id of rim:ExtrinsicObject for wsdl:binding>

4.5.2 Attribute sourceObject

The sourceObject attribute value of the rim:Association MUST contain as value the id of the rim:ServiceBinding for wsdl:port.

4.5.3 Attribute targetObject

The targetObject attribute value of the rim:Association MUST contain as value the id of the rim:ExtrinsicObject for wsdl:binding.

4.5.4 Attribute associationType

The associationType attribute value of the rim:Association MUST contain as value:

`urn:oasis:names:tc:ebxml-regrep:AssociationType:Implements`

which is the id of the canonical “Implements” ClassificationNode within the canonical AssociationType ClassificationScheme.

4.6 wsdl:binding ↔wsdl:portType Association Mapping

This Association associates a wsdl:binding to its wsdl:portType. It is specified as follows:

4.6.1 Attribute id

The id attribute value of the rim:Association MUST have the following pattern:

<id of rim:ExtrinsicObject for wsdl:binding>:Implements:<id of rim:ExtrinsicObject for wsdl:portType>

4.6.2 Attribute sourceObject

The sourceObject attribute value of the rim:Association MUST contain as value the id of the rim:ExtrinsicObject for wsdl:binding.

4.6.3 Attribute targetObject

The targetObject attribute value of the rim:Association MUST contain as value the id of the rim:ExtrinsicObject for wsdl:portType.

4.6.4 Attribute associationType

The associationType attribute value of the rim:Association MUST contain as value:

`urn:oasis:names:tc:ebxml-regrep:AssociationType:Implements`

which is the id of the canonical “Implements” ClassificationNode within the canonical AssociationType ClassificationScheme.

5 Publishing Profile

This chapter profiles how Web Services artifacts MUST be published to an ebXML Registry implementing the WS Profile.

5.1 Structure of WSDL Documents

A WSDL description of a web service MAY be contained in a single file. However, to faciliate better resuse, it SHOULD be split into multiple WSDL files as described next. Examples of such suggested WSDL partioning are illustrated by ebXML Registry 3.0 WSDL documents. 'Xx' is a place holder for the specific type of WSDL being defined (e.g. 'ebXML Registry').

5.1.1 XxInterfaces.wsdl

This file should contain types, message and portType (includes operations) element definitions.

5.1.2 XxBindings.wsdl

This file SHOULD contain binding elements.

5.1.3 XxServices.wsdl

This file SHOULD contain the service elements.

6 Validation Service Profile

The ebXML Regitsry provides the ability for a content validation service to be configured for any type of content. The purpose of validation service is to enforce conformance to business rules or policies governing content published to the registry in a content specific manner. Content validation is a key feature for enabling SOA governance within ebXML Registry.

A WSDL document, when published to an ebXML Registry implementing the WS Profile, MUST be validated as specified in this section using a Content Validation Service as defined by [ebRS].

6.1 Invocation Control File

The WSDL validation service MUST support an invocation control file that declaratively specifies the business rules for validating WSDL documents upon publishing. It MUST NOT require programming in order to support the required Business Rules defined in the next section.

6.2 Business Rules

The following business rules MUST be supported by the WSDL validation service and MUST be expressable declaratively within the Invocation Control File:

- Ability to specify that published WSDL documents MUST restrict <wsdl:binding> elements to only use a subset of the ClassificationNodes specified within the WSDLBindingType ClassificationScheme defined by this profile. For example it MUST be possible to express the rule that a binding element MUST only use SOAP binding.
- Ability to specify that published WSDL documents MUST restrict <soap:binding> style attribute to a subset of the ClassificationNodes specified within the SOAPBindingStyle ClassificationScheme defined by this profile. For example it MUST be possible to express the rule that a SOAP binding MUST only use "document" style.
- Ability to specify that published WSDL documents MUST restrict <soap:binding> transport attribute to a subset of the ClassificationNodes specified within the TransportType ClassificationScheme defined by this profile. For example it MUST be possible to express the rule that a SOAP binding MUST only use "http" transport.

The WSDL validation service MAY support any other business rules in addition to those listed above.

7 Cataloging Service Profile

The ebXML Regitsry provides the ability for a content cataloging service to be configured for any type of content. The cataloging service serves the following purposes:

- Automates the mapping from the source information model (in this case WSDL) to ebRIM. This hides the complexity of the mapping from the WSDL publisher and eliminates the need for any special UI tools to be provided by the registry implementor for publishing WSDL documents.
- Selectively converts content into ebRIM compatible metadata when the content is cataloged after being published. The generated metadata enables the selected content to be used as parameter(s) in content specific parameterized queries.

This section describes the cataloging service for cataloging WSDL content.

A WSDL document, when published to an ebXML Registry implementing the WS Profile, MUST be cataloged as specified in this section using a WSDL Content Cataloging Service as defined by [ebRS].

7.1 Invocation Control File

The WSDL cataloging service MAY optionally support an invocation control file that declaratively specifies the transforms necessary to catalog published WSDL documents.

7.2 Input Metadata

The WSDL cataloging service MUST be pre-configured to be automatically invoked when the following types of metadata are published, as defined by the [ebRS] specifications.

These are the only types of metadata that MAY describe a WSDL document being published:

- An ExtrinsicObject whose ObjectType references the canonical WSDL ClassificationNode specified in chapter 11. The ExtrinsicObject MUST have a WSDL document as its RepositoryItem.
- An ExternalLink whose ObjectType references the canonical WSDL ClassificationNode specified in chapter 11. In case of ExternalLink the WSDL document MUST be resolvable via a URL described by the value of the externalURI attribute of the ExternalLink. Recall that, in the ExternalLink case the WSDL document is not be stored in the repository.

```
<rim:ExtrinsicObject id="urn:acmeinc:ebxml:registry:3.0:services:wsdl">
...
<rim:ExtrinsicObject>
```

Example of ExtrinsicObject Input Metadata

```
<rim:ExternalLink
  id="urn:acmeinc:ebxml:registry:3.0:services:wsdl"
  externalURI="http://www.acme.com/wsdl/ebXMLRegistryService.wsdl"
  >
...
<rim:ExternalLink>
```

Example of ExternalLink Input Metadata

7.3 Input Content

The WSDL cataloging service expects a WSDL document as its input content. The input content MUST be processed by the WSDL cataloging service regardless of whether it is a RepositoryItem for an ExtrinsicObject or whether it is content external to repository that is referenced by an ExternalLink. The input WSDL file may contain imports of other WSDL files. The WSDL cataloging service MUST implicitly process WSDL documents that have been imported within the explicitly submitted WSDL document as

defined in ??.

7.4 Output Metadata

This section describes the metadata produced by the WSDL cataloging service produces as output.

7.4.1 Changes to Input Metadata

The WSDL cataloging service MUST make the following changes to the input ExtrinsicObject or ExternalLink metadata.

Slot importedNameSpaces

A Slot `importedNamespaces` MUST be added to the input metadata to describe the XML namespaces that are imported by the input WSDL. The slotName MUST be `urn:oasis:names:tc:ebxml-regrep:profile:ws:wsdl:importedNameSpaces`. The value of this slot MUST be a collection of URNs where each URN identifies the URN of a namespace that is imported by the input WSDL.

Slot targetNamespace

A Slot `targetNamespace` MUST be added to the input metadata to describe the target XML namespace for the input WSDL. The slotName MUST be `urn:oasis:names:tc:ebxml-regrep:profile:ws:wsdl:targetNameSpace`. The value of this slot MUST be a a URN where that identifies the URN of a targetNamespace for the input WSDL.

7.4.2 wsdl:service → rim:Service

The WSDL Cataloging service MUST automatically produce a rim:Service instance for each wsdl:service element within the input WSDL or its imports, as specified in the wsdl:service → rim:Service mapping earlier in this document.

7.4.3 wsdl:port → rim:ServiceBinding

The WSDL Cataloging service MUST automatically produce an rim:ServiceBinding instance for each wsdl:port element within the input WSDL or its imports, as specified in the wsdl:port → rim:ServiceBinding mapping earlier in this document.

7.4.4 wsdl:binding → rim:ExtrinsicObject

The WSDL Cataloging service MUST automatically produce an rim:ExtrinsicObject instance for each wsdl:binding element within the input WSDL or its imports, as specified in the wsdl:binding → rim:ExtrinsicObject mapping earlier in this document.

7.4.5 wsdl:portType → rim:ExtrinsicObject

The WSDL Cataloging service MUST automatically produce an rim:ExtrinsicObject instance for each wsdl:portType element within the input WSDL or its imports, as specified in the wsdl:portType → rim:ExtrinsicObject mapping earlier in this document.

7.4.6 wsdl:port ↔ wsdl:binding Association

The WSDL Cataloging service MUST automatically produce rim:Association instances for each wsdl:port element within the input WSDL or its imports, as specified in the wsdl:port → wsdl:binding Association mapping earlier in this document.

7.4.7 wsdl:binding ↔ wsdl:portType Association

The WSDL Cataloging service MUST automatically produce rim:Association instances for each wsdl:binding element within the input WSDL or its imports, as specified in the wsdl:binding →

1015 wsdl:portType Association mapping earlier in this document.

8 Discovery Profile

The ebXML Regitsry provides the ability for a user defined parameterized queries to be configured for each type of content. The queries may be as complex or simple as the discovery use case requires. The complexity of the parameterized queries may hidden from the registry client by storing them within the ebXML Registry as instances of the AdhocQuery class, and being invoked by simply providing their parameters. Query parameters are often pattern strings that may contain wildcard characters '%' (matches any number of characters) and '_' (matches exactly one character) as described by [ebRS].

An ebXML Registry SHOULD provide a graphical user interface that displays any configured parameterized query as a form which contains an appropriate field for entering each query parameter.

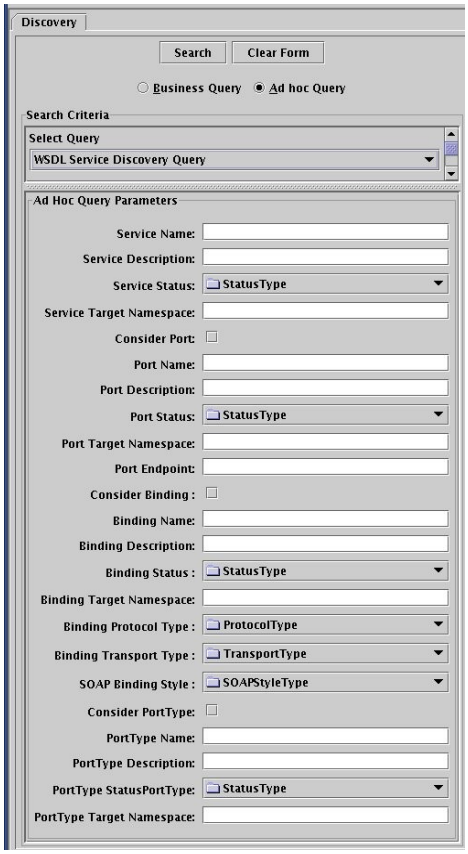

Illustration 4: Example of Parameterized Form for WSDL Service Discovery Query

This chapter defines the queries that MUST be support by an ebXML Registry implementing the WS Profile for discovering WSDL content. An implementation MAY also support additional discovery queries for WSDL content.

8.1 Overview

Refer to the layered architecture of WSDL information model described in chapter 2.

Discovery is the process that enables discovering higher level objects based on search criteria that predicates upon attributes of the type of object being discovered as well. as the attributes of lower level objects in the model.

In contrast, drill-down is the process that enables explore a higher level object and determining the lower level objects that were used in building the higher level object.

[ebRIM] provides support for drill-down use cases in a generic manner out-of-box. For example, [ebRIM] provides:

- A generic ability to explore all rim:ServiceBindings used within a rim:Service
- A generic ability to explore all Association involving a ServiceBinding or any RegistryObject

The queries defined in this chapter are therefor focused on discovery rather than drill-down. Because ebXML Registry supports a flexible and extensible query capability, this chapter defines a single discovery query for each discoverable type within the WSDL information model. Each query may be as complex as necessary. However, the complexity is hidden from the client using parameterized queries stored in the Registry as instances of the AdhocQuery type, in the same manner as any other RegistryObject.

In the subsequent section each query is described simply in terms of its supported parameters that serve as its search criteria. The actual AdhocQuery instances are much more complex in comparison but they are not exposed to the client making the query. Details on these queries are specified canonically in chapter 11.

8.1.1 Discovery Query Patterns

The discovery queries are specified from the bottom of the layered WSDL information model to the top (portType, binding, port and service). The query for each layer specifies parameters specific to it as well as parameters specific to each of the lower layers that it builds upon. Thus the number of parameters increase as queries are defined for higher level types in the model. This is key to being able to discover higher level objects based on attributes of the lower level objects that they build upon.

There are many parameters supported for the discovery query for each higher level type in the model. However, it is often the case that discovery may not require parameters specific to all lower level types. To facilitate pruning of the discovery query for unwanted predicates related to lower level types there is a special parameter name $considerXXX where XXX represents a lower level type within the model. If the value of this parameter is set to “0” then all parameter values specific to lower level type MUST are ignored by the discovery query.

WSDL Document Discovery Query

The WSDL Document discovery query MUST be implemented by an ebXML Registry implementing this profile. It allows the discovery of WSDL documents using zero or more of the parameters described next.

8.1.2 Parameter $name

This parameter's value MAY specify a string containing a pattern to match against the name attribute value of RegistryObjects that have objectType of WSDL.

8.1.3 Parameter $description

This parameter's value MAY specify a string containing a pattern to match against the description attribute value of RegistryObjects that have objectType of WSDL.

8.1.4 Parameter $targetNamespace

This parameter's value MAY specify a string containing a pattern to match against the targetNamespace of a WSDL document.

8.1.5 Parameter $importedNamespace

This parameter's value MAY specify a string containing a pattern to match against the namespaces imported by a WSDL document.

8.1.6 Example of WSDL Document Discovery Query

The following example illustrates how to find all WSDL documents that have a targetNamespace containing the string “oasis” and that import a namespace with string “org.w3”. Note that additional supported parameters MAY also be specified if needed.

```
<AdhocQueryRequest>
  <rim:AdhocQuery id="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:WSDLDiscoveryQuery">
    <rim:Slot name="$targetNamespace">
      <rim:ValueList>
        <rim:Value>%oasis%</rim:Value>
      </rim:ValueList>
    </rim:Slot>
    <rim:Slot name="$importedNamespace">
      <rim:ValueList>
        <rim:Value>%org.w3%</rim:Value>
      </rim:ValueList>
    </rim:Slot>
  </rim:AdhocQuery>
</AdhocQueryRequest>
```

Example of WSDL Document Discovery Query

8.2 PortType Discovery Query

The WSDL PortType discovery query allows the discovery of wsdl:portType instances using zero or more of the parameters described next.

8.2.1 Parameter $portType.name

This parameter's value MAY specify a string containing a pattern to match against the name attribute value of wsdl:portType instances.

8.2.2 Parameter $portType.description

This parameter's value MAY specify a string containing a pattern to match against the content of the wsdl:documentation element within wsdl:portType instances.

8.2.3 Parameter $portType.targetNamespace

This parameter's value MAY specify a string containing a pattern to match against the targetNamespace of wsdl:portType instances.

8.2.4 Parameter $portType.schemaNamespaces

This parameter's value MAY specify a string containing a pattern to match against the XML schema namespaces used within the wsdl:message instances used within the wsdl:operation instances used within the wsdl:portType instances.

8.2.5 Example of WSDL PortType Discovery Query

The following example illustrates how to find all wsdl:portType instances that have a name containing the string “QueryManager” and have import an XML Schema with namespace containing the string “ebxml-regrep”. Note that additional supported parameters MAY also be specified if needed.

```
<AdhocQueryRequest>
  <rim:AdhocQuery id="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:WSDLPortTypeDiscoveryQuery">
    <rim:Slot name="$portType.name">
```

```
      <rim:ValueList>
        <rim:Value>%QueryManager%</rim:Value>
      </rim:ValueList>
    </rim:Slot>
    <rim:Slot name="$portType.schemaNamespaces">
      <rim:ValueList>
        <rim:Value>%ebxml-regrep%</rim:Value>
      </rim:ValueList>
    </rim:Slot>
  </rim:AdhocQuery>
</AdhocQueryRequest>
```

Example of WSDL PortType Discovery Query

8.3 WSDL Binding Discovery Query

The WSDL Binding discovery query allows the discovery of wsdl:binding instances using zero or more of the parameters described next.

8.3.1 Parameter $binding.name

This parameter's value MAY specify a string containing a pattern to match against the name attribute value of wsdl:binding instances.

8.3.2 Parameter $binding.description

This parameter's value MAY specify a string containing a pattern to match against the content of the wsdl:documentation element within wsdl:binding instances.

8.3.3 Parameter $binding.targetNamespace

This parameter's value MAY specify a string containing a pattern to match against the targetNamespace of wsdl:binding instances.

8.3.4 Parameter $binding.protocolType

This parameter's value MAY specify a string containing a pattern to match against the id attribute value of the ClassificationNode representing the protocolType supported by wsdl:binding instances.

8.3.5 Parameter $binding.transportType

This parameter's value MAY specify a string containing a pattern to match against the id attribute value of the ClassificationNode representing the transportType supported by wsdl:binding instances.

8.3.6 Parameter $binding.soapStyle

This parameter's value MAY specify a string containing a pattern to match against the id attribute value of the ClassificationNode representing the soap style of wsdl:binding instances.

8.3.7 Parameter $considerPortType

This parameter's value MAY specify a string a “1” or “0” to indicate whether or not to consider the portType specific parameters that follow when processing the query. If unspecified the value defaults to “0”.

8.3.8 Parameter $portType.name

This parameter's value MAY specify a string containing a pattern to match against the name attribute value of wsdl:portType instances that are used by the objects being discovered.

8.3.9 Parameter $portType.description

This parameter's value MAY specify a string containing a pattern to match against the content of the wsdl:documentation element within wsdl:portType instances that are used by the objects being discovered.

8.3.10 Parameter $portType.targetNamespace

This parameter's value MAY specify a string containing a pattern to match against the targetNamespace of wsdl:portType instances that are used by the objects being discovered.

8.3.11 Parameter $portType.schemaNamespace

This parameter's value MAY specify a string containing a pattern to match against the XML schema namespaces used within the wsdl:message instances used within the wsdl:operation instances used within the wsdl:portType instances that are used by the objects being discovered.

8.3.12 Example of WSDL Binding Discovery Query

The following example illustrates how to find all wsdl:binding instances that have a name containing the string “QueryManager” and have a binding that supports SOAP protocol using HTTP transport. Note that additional supported parameters MAY also be specified if needed.

```
<AdhocQueryRequest>
  <rim:AdhocQuery id="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:WSDLBindingDiscoveryQuery">
    <rim:Slot name="$binding.name">
      <rim:ValueList>
        <rim:Value>%QueryManager%</rim:Value>
      </rim:ValueList>
    </rim:Slot>
    <rim:Slot name="$binding.protocolType">
      <rim:ValueList>
        <rim:Value>urn:oasis:names:tc:ebxml-
regrep:profile:ws:ProtocolType:SOAP</rim:Value>
      </rim:ValueList>
    </rim:Slot>
  </rim:AdhocQuery>
</AdhocQueryRequest>
```

Example of WSDL Port Discovery Query

8.4 WSDL Port Discovery Query

The WSDL Port discovery query allows the discovery of wsdl:port instances using zero or more of the parameters described next.

8.4.1 Parameter $port.name

This parameter's value MAY specify a string containing a pattern to match against the name attribute value of wsdl:port instances.

8.4.2 Parameter $port.description

This parameter's value MAY specify a string containing a pattern to match against the content of the wsdl:documentation element within wsdl:port instances.

8.4.3 Parameter $port.targetNamespace

This parameter's value MAY specify a string containing a pattern to match against the targetNamespace of wsdl:port instances.

8.4.4 Parameter $port.accessURI

This parameter's value MAY specify a string containing a pattern to match against the accessURI for the endpoint defined for the wsdl:port instances.

8.4.5 Parameter $considerBinding

This parameter's value MAY specify a string a “1” or “0” to indicate whether or not to consider the binding specific parameters that follow when processing the query. If unspecified the value defaults to “0”.

8.4.6 Parameter $binding.name

This parameter's value MAY specify a string containing a pattern to match against the name attribute value of wsdl:binding instances that are used by the objects being discovered.

8.4.7 Parameter $binding.description

This parameter's value MAY specify a string containing a pattern to match against the content of the wsdl:documentation element within wsdl:binding instances that are used by the objects being discovered.

8.4.8 Parameter $binding.targetNamespace

This parameter's value MAY specify a string containing a pattern to match against the targetNamespace of wsdl:binding instances that are used by the objects being discovered.

8.4.9 Parameter $binding.protocolType

This parameter's value MAY specify a string containing a pattern to match against the id attribute value of the ClassificationNode representing the protocolType supported by wsdl:binding instances that are used by the objects being discovered.

8.4.10 Parameter $binding.transportType

This parameter's value MAY specify a string containing a pattern to match against the id attribute value of the ClassificationNode representing the transportType supported by wsdl:binding instances that are used by the objects being discovered.

8.4.11 Parameter $binding.soapStyle

This parameter's value MAY specify a string containing a pattern to match against the id attribute value of the ClassificationNode representing the soap style of wsdl:binding instances that are used by the objects being discovered.

8.4.12 Parameter $considerPortType

This parameter's value MAY specify a string a “1” or “0” to indicate whether or not to consider the portType specific parameters that follow when processing the query. If unspecified the value defaults to “0”.

8.4.13 Parameter $portType.name

This parameter's value MAY specify a string containing a pattern to match against the name attribute value of wsdl:portType instances that are used by the objects being discovered.

8.4.14 Parameter $portType.description

This parameter's value MAY specify a string containing a pattern to match against the content of the wsdl:documentation element within wsdl:portType instances that are used by the objects being discovered.

8.4.15 Parameter $portType.targetNamespace

This parameter's value MAY specify a string containing a pattern to match against the targetNamespace of wsdl:portType instances that are used by the objects being discovered.

8.4.16 Parameter $portType.schemaNamespace

This parameter's value MAY specify a string containing a pattern to match against the XML schema namespaces used within the wsdl:message instances used within the wsdl:operation instances used within the wsdl:portType instances that are used by the objects being discovered.

8.4.17 Example of WSDL Port Discovery Query

The following example illustrates how to find all wsdl:port instances that have a name containing the string “QueryManager” and have a binding that supports SOAP protocol using HTTP transport. Note that additional supported parameters MAY also be specified if needed.

```
<AdhocQueryRequest>
  <rim:AdhocQuery id="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:WSDLPortDiscoveryQuery">
    <rim:Slot name="$port.name">
      <rim:ValueList>
        <rim:Value>%QueryManager%</rim:Value>
      </rim:ValueList>
    </rim:Slot>
    <rim:Slot name="$port.accessURI">
      <rim:ValueList>
        <rim:Value>http://acme%</rim:Value>
      </rim:ValueList>
    </rim:Slot>
  </rim:AdhocQuery>
</AdhocQueryRequest>
```

Example of WSDL Port Discovery Query

8.5 WSDL Service Discovery Query

The WSDL Service discovery query allows the discovery of wsdl:service instances using zero or more of the parameters described next.

8.5.1 Parameter $service.name

This parameter's value MAY specify a string containing a pattern to match against the name attribute value of wsdl:service instances.

8.5.2 Parameter $service.description

This parameter's value MAY specify a string containing a pattern to match against the content of the wsdl:documentation element within wsdl:service instances.

8.5.3 Parameter $service.targetNamespace

This parameter's value MAY specify a string containing a pattern to match against the targetNamespace of wsdl:service instances.

8.5.4 Parameter $considerPort

This parameter's value MAY specify a string a “1” or “0” to indicate whether or not to consider the port specific parameters that follow when processing the query. If unspecified the value defaults to “0”.

8.5.5 Parameter $port.name

This parameter's value MAY specify a string containing a pattern to match against the name attribute value of wsdl:port instances that are used by the objects being discovered.

8.5.6 Parameter $port.description

This parameter's value MAY specify a string containing a pattern to match against the content of the wsdl:documentation element within wsdl:port instances that are used by the objects being discovered.

8.5.7 Parameter $port.targetNamespace

This parameter's value MAY specify a string containing a pattern to match against the targetNamespace of wsdl:port instances that are used by the objects being discovered.

8.5.8 Parameter $port.accessURI

This parameter's value MAY specify a string containing a pattern to match against the accessURI for the endpoint defined for the wsdl:port instances that are used by the objects being discovered.

8.5.9 Parameter $considerBinding

This parameter's value MAY specify a string a “1” or “0” to indicate whether or not to consider the binding specific parameters that follow when processing the query. If unspecified the value defaults to “0”.

8.5.10 Parameter $binding.name

This parameter's value MAY specify a string containing a pattern to match against the name attribute value of wsdl:binding instances that are used by the objects being discovered.

8.5.11 Parameter $binding.description

This parameter's value MAY specify a string containing a pattern to match against the content of the wsdl:documentation element within wsdl:binding instances that are used by the objects being discovered.

8.5.12 Parameter $binding.targetNamespace

This parameter's value MAY specify a string containing a pattern to match against the targetNamespace of wsdl:binding instances that are used by the objects being discovered.

8.5.13 Parameter $binding.protocolType

This parameter's value MAY specify a string containing a pattern to match against the id attribute value of the ClassificationNode representing the protocolType supported by wsdl:binding instances that are used by the objects being discovered.

8.5.14 Parameter $binding.transportType

This parameter's value MAY specify a string containing a pattern to match against the id attribute value of the ClassificationNode representing the transportType supported by wsdl:binding instances that are used by the objects being discovered.

8.5.15 Parameter $binding.soapStyle

This parameter's value MAY specify a string containing a pattern to match against the id attribute value of the ClassificationNode representing the soap style of wsdl:binding instances that are used by the objects being discovered.

8.5.16 Parameter $considerPortType

This parameter's value MAY specify a string a “1” or “0” to indicate whether or not to consider the portType specific parameters that follow when processing the query. If unspecified the value defaults to “0”.

8.5.17 Parameter $portType.name

This parameter's value MAY specify a string containing a pattern to match against the name attribute value of wsdl:portType instances that are used by the objects being discovered.

8.5.18 Parameter $portType.description

This parameter's value MAY specify a string containing a pattern to match against the content of the wsdl:documentation element within wsdl:portType instances that are used by the objects being discovered.

8.5.19 Parameter $portType.targetNamespace

This parameter's value MAY specify a string containing a pattern to match against the targetNamespace of wsdl:portType instances that are used by the objects being discovered.

8.5.20 Parameter $portType.schemaNamespace

This parameter's value MAY specify a string containing a pattern to match against the XML schema namespaces used within the wsdl:message instances used within the wsdl:operation instances used within the wsdl:portType instances that are used by the objects being discovered.

8.5.21 Example of WSDL Service Discovery Query

The following example illustrates how to find all wsdl:service instances that have a name containing the string “QueryManager” and have a targetNamespace containing the string “ebXML”. Note that additional supported parameters MAY also be specified if needed.

```
<AdhocQueryRequest>
  <rim:AdhocQuery id="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:WSDLServiceDiscoveryQuery">
    <rim:Slot name="$name">
      <rim:ValueList>
        <rim:Value>%QueryManager%</rim:Value>
      </rim:ValueList>
    </rim:Slot>
    <rim:Slot name="$targetNamespace">
      <rim:ValueList>
        <rim:Value>%ebXML%</rim:Value>
      </rim:ValueList>
    </rim:Slot>
```

```
  </rim:AdhocQuery>
</AdhocQueryRequest>
```

Example of WSDL Service Discovery Query

9 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. The event matching criteria is specified using a stored parameterized query similar to the discovery queries described in previous chapter.

This chapter specifies the template Subscriptons that MUST be implemented by an ebXML Registry implementing the Web Services profile. To subscribe to instances of types defined within the WSDL information model a subscription can simply reuse the discovery queries defined in the previous chapter or define more specific queries.

9.1 Subscribing to a WSDL Document

A client may publish a rim:Subscription instance using the WSDL Document discovery query as the selector in order to receive notification of changes to WSDL documents that they have an interest in.

```
<rim:Subscription id=${WSDL_SUBSCRIPTION_ID}
  selector="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:WSDLDiscoveryQuery">
  <rim:Slot name="$name">
    <rim:ValueList>
      <rim:Value>%ebXML%</rim:Value>
    </rim:ValueList>
  </rim:Slot>
  <rim:Slot name="$targetNamespace">
    <rim:ValueList>
      <rim:Value>%oasis%</rim:Value>
    </rim:ValueList>
  </rim:Slot>

  <!-- email address endPoint for receiving notification via email -->
  <rim:NotifyAction notificationOption="urn:oasis:names:tc:ebxml-
regrep:NotificationOptionType:ObjectRefs"
    endPoint="mailto:farrukh.najmi@sun.com"/>

  <!—Web Service endPoint for receiving notification via SOAP -->
  <rim:NotifyAction notificationOption="urn:oasis:names:tc:ebxml-
regrep:NotificationOptionType:Objects"
    endPoint="urn:acme:wsdlChangeListenerService"/>
</rim:Subscription>
```

Listing 11: Example of Subscription to WSDL Documents

The above example show how to create a subscription for WSDL Documents where name contains the string “ebXML” and targetNamespace contains the string “oasis”. Light-weight notifications containing references to WSDL dcouments are configured to be sent to an email address while heavy-weight notifications containing actual WSDL documents are configured to be sent to a listener service using the SOAP protocol.

9.2 Subscribing to PortType changes

A client may publish a rim:Subscription instance using the WSDL PortType discovery query as the selector in order to receive notification of changes to WSDL PortTypes that they have an interest in.

```
<rim:Subscription id=${WSDL_SUBSCRIPTION_ID}
  selector="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:PortTypeDiscoveryQuery">

...
</rim:Subscription>
```

Listing 12: Example of Subscription to WSDL PortTypes

9.3 Subscribing to Binding changes

A client may publish a rim:Subscription instance using the WSDL Binding discovery query as the selector in order to receive notification of changes to WSDL PortBindings that they have an interest in.

```
<rim:Subscription id=${WSDL_SUBSCRIPTION_ID}
  selector="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:BindingDiscoveryQuery">

...
</rim:Subscription>
```

Listing 13: Example of Subscription to WSDL Bindings

9.4 Subscribing to Port changes

A client may publish a rim:Subscription instance using the WSDL Port discovery query as the selector in order to receive notification of changes to WSDL Ports that they have an interest in.

```
<rim:Subscription id=${WSDL_SUBSCRIPTION_ID}
  selector="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:PortDiscoveryQuery">

...
</rim:Subscription>
```

Listing 14: Example of Subscription to WSDL Ports

9.5 Subscribing to Service changes

A client may publish a rim:Subscription instance using the WSDL Service discovery query as the selector in order to receive notification of changes to WSDL Services that they have an interest in.

```
<rim:Subscription id=${WSDL_SUBSCRIPTION_ID}
  selector="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:ServiceDiscoveryQuery">

...
</rim:Subscription>
```

Listing 15: Example of Subscription to WSDL Ports

10 Security Profile

This chapter specifies security aspects governing the publish, management and discovery of web services within ebXML Registry.

10.1 SubjectRole Profile

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.

TBD??

10.2 SubjectGroup Profile

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.

TBD??

10.3 AcessControlPolicy 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.

...

TBD??

11 Canonical Metadata Definitions

This chapter specifies the canonical metadata defined by this profile.

11.1 ObjectType Extensions

The following new extensions to the canonical ObjectType ClassificationScheme are described by this profile:

```
    <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject"
lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL"  code="WSDL"
id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL">
      <rim:Name>
        <rim:LocalizedString charset="UTF-8" value="label.WSDL"/>
      </rim:Name>
      <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL"
lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType"
code="PortType" id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType">
        <rim:Name>
          <rim:LocalizedString charset="UTF-8" value="label.PortType"/>
        </rim:Name>
      </rim:ClassificationNode>
      <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL"
lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding"
code="Binding" id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding">
        <rim:Name>
          <rim:LocalizedString charset="UTF-8" value="label.Binding"/>
        </rim:Name>
      </rim:ClassificationNode>
      <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL"
lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Port"  code="Port"
id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Port">
        <rim:Name>
          <rim:LocalizedString charset="UTF-8" value="label.Port"/>
        </rim:Name>
      </rim:ClassificationNode>
      <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL"
lid="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Service"
code="Service" id="urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Service">
        <rim:Name>
          <rim:LocalizedString charset="UTF-8" value="label.Service"/>
        </rim:Name>
      </rim:ClassificationNode>
    </rim:ClassificationNode>
```

Listing 16: Example of Subscription to WSDL Ports

11.2 Canonical ClassificationSchemes

The following new canonical ClassificationSchemes are described by this profile:

```
    <rim:ClassificationScheme lid="urn:oasis:names:tc:ebxml-
regrep:profile:ws:classificationScheme:ProtocolType"
id="urn:oasis:names:tc:ebxml-
regrep:profile:ws:classificationScheme:ProtocolType" isInternal="true"
nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode">
      <rim:Name>
        <rim:LocalizedString charset="UTF-8"
value="label.ProtocolType"/>
      </rim:Name>
      <rim:Description>
        <rim:LocalizedString charset="UTF-8"
value="label.ProtocolType.desc"/>
      </rim:Description>
      <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
regrep:profile:ws:ProtocolType:SOAP" code="SOAP"
id="urn:oasis:names:tc:ebxml-regrep:profile:ws:ProtocolType:SOAP"/>
      <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
regrep:profile:ws:ProtocolType:AS2" code="AS2"
id="urn:oasis:names:tc:ebxml-regrep:profile:ws:ProtocolType:AS2"/>
      <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
regrep:profile:ws:ProtocolType:Atom" code="Atom"
id="urn:oasis:names:tc:ebxml-regrep:profile:ws:ProtocolType:Atom"/>
    </rim:ClassificationScheme>

    <rim:ClassificationScheme lid="urn:oasis:names:tc:ebxml-
regrep:profile:ws:classificationScheme:TransportType"
id="urn:oasis:names:tc:ebxml-
regrep:profile:ws:classificationScheme:TransportType" isInternal="true"
nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode">
      <rim:Name>
        <rim:LocalizedString charset="UTF-8"
value="label.TransportType"/>
      </rim:Name>
      <rim:Description>
        <rim:LocalizedString charset="UTF-8"
value="label.TransportType.desc"/>
      </rim:Description>
      <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
regrep:profile:ws:TransportType:HTTP" code="HTTP"
id="urn:oasis:names:tc:ebxml-regrep:profile:ws:TransportType:HTTP"/>
      <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
regrep:profile:ws:TransportType:MOM" code="MOM"
id="urn:oasis:names:tc:ebxml-regrep:profile:ws:TransportType:MOM"/>
      <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
regrep:profile:ws:TransportType:BEEP" code="BEEP"
id="urn:oasis:names:tc:ebxml-regrep:profile:ws:TransportType:BEEP"/>
    </rim:ClassificationScheme>

    <rim:ClassificationScheme lid="urn:oasis:names:tc:ebxml-
regrep:profile:ws:classificationScheme:SOAPStyleType"
id="urn:oasis:names:tc:ebxml-
regrep:profile:ws:classificationScheme:SOAPStyleType" isInternal="true"
nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode">
      <rim:Name>
        <rim:LocalizedString charset="UTF-8"
value="label.SOAPStyleType"/>
      </rim:Name>
      <rim:Description>
        <rim:LocalizedString charset="UTF-8"
value="label.SOAPType.desc"/>
      </rim:Description>
      <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
regrep:profile:wsPStyleType:RPC" code="RPC"
id="urn:oasis:names:tc:ebxml-regrep:profile:wsPStyleType:RPC"/>
      <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
regrep:profile:wsPStyleType:Document" code="Document"
id="urn:oasis:names:tc:ebxml-regrep:profile:wsPStyleType:Document"/>
    </rim:ClassificationScheme>
```

Listing 17: Example of Subscription to WSDL Ports

11.3 Canonical Queries

The following new canonical queries are described by this profile. Note that while these queries are complex, the complexity is hidden from clients by exposing only the query parameters to them.

11.3.1 WSDL Document Discovery Query

```
    <!--
    Parameterized Adhoc Query for WSDL Discovery query.
    Finds by Name, Description, ComponentType, ResourceType
    -->
    <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:WSDLDiscoveryQuery"
id="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:WSDLDiscoveryQuery">
      <rim:Name>
        <rim:LocalizedString value="label.WSDLDiscoveryQuery"/>
      </rim:Name>
      <rim:Description>
        <rim:LocalizedString value="label.WSDLDiscoveryQuery.desc"/>
      </rim:Description>
      <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
regrep:QueryLanguage:SQL-92">
SELECT DISTINCT ro.* FROM RegistryObject ro, Name_ nm, Description d,
Slot tns, Slot ins
WHERE (1=1)
 AND (ro.objectType = ''urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL'')
 AND (nm.parent = ro.id AND UPPER ( nm.value ) LIKE UPPER ( ''$name''
 ) )
 AND (d.parent = ro.id AND UPPER ( d.value ) LIKE UPPER
( ''$description'' ) )
 AND (ro.id = tns.parent
  AND tns.name_ = ''urn:oasis:names:tc:ebxml-
regrep:profile:ws:wsdl:targetNamespace''
  AND tns.value LIKE ''$targetNamespace'')
 AND (ro.id = ins.parent
  AND ins.name_ = ''urn:oasis:names:tc:ebxml-
regrep:profile:ws:wsdl:importedNamespaces''
  AND ins.value LIKE ''$importedNamespaces'')

        </rim:QueryExpression>
    </rim:AdhocQuery>
```

Listing 18: WSDL Document Discovery Query

11.3.2 WSDL PortType Discovery Query

```
    <!--
    Parameterized Adhoc Query for WSDL PortType Discovery query.
    -->
    <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:PortTypeDiscoveryQuery"
id="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:PortTypeDiscoveryQuery">
      <rim:Name>
        <rim:LocalizedString value="label.PortTypeDiscoveryQuery"/>
      </rim:Name>
      <rim:Description>
        <rim:LocalizedString value="label.PortTypeDiscoveryQuery.desc"/>
      </rim:Description>
      <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
regrep:QueryLanguage:SQL-92">
SELECT DISTINCT portType.* FROM ExtrinsicObject portType, Name_
portTypeName, Description portTypeDesc, Slot portTypeTNS
WHERE
 portType.objectType = ''urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType''
 AND (portTypeName.parent = portType.id AND UPPER ( portTypeName.value )
LIKE UPPER ( ''$portType.name'' ) )
```

```
 AND (portTypeDesc.parent = portType.id AND UPPER ( portTypeDesc.value )
LIKE UPPER ( ''$portType.description'' ) )
 AND (portType.id = s.parent
  AND portTypeTNS.name_ = ''urn:oasis:names:tc:ebxml-
regrep:profile:ws:wsdl:targetNamespace''
  AND portTypeTNS.value LIKE ''$portType.targetNamespace'')
        </rim:QueryExpression>
    </rim:AdhocQuery>
```

Listing 19: WSDL PortType Discovery Query

11.3.3 WSDL Binding Discovery Query

```
    <!--
    Parameterized Adhoc Query for WSDL Binding Discovery query.
    -->
    <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:BindingDiscoveryQuery"
id="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:BindingDiscoveryQuery">
      <rim:Name>
        <rim:LocalizedString value="label.BindingDiscoveryQuery"/>
      </rim:Name>
      <rim:Description>
        <rim:LocalizedString value="label.BindingDiscoveryQuery.desc"/>
      </rim:Description>
      <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
regrep:QueryLanguage:SQL-92">
SELECT DISTINCT binding.* FROM
  ExtrinsicObject binding, Name_ bindingName, Description bindingDesc,
Slot bindingTNS,
  Association implements
WHERE
 binding.objectType = ''urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding''
 AND (bindingName.parent = binding.id AND UPPER ( bindingName.value )
LIKE UPPER ( ''$binding.name'' ) )
 AND (bindingDesc.parent = binding.id AND UPPER ( bindingDesc.value )
LIKE UPPER ( ''$binding.description'' ) )
 AND (binding.id = s.parent
  AND bindingTNS.name_ = ''urn:oasis:names:tc:ebxml-
regrep:profile:ws:wsdl:targetNamespace''
  AND bindingTNS.value LIKE ''$binding.targetNamespace'')
 AND (binding.id IN ( SELECT classifiedObject FROM Classification WHERE
classificationNode IN (  SELECT id
 FROM ClassificationNode WHERE path LIKE ''$binding.protocolType'' ) ))
 AND (binding.id IN ( SELECT classifiedObject FROM Classification WHERE
classificationNode IN (  SELECT id
 FROM ClassificationNode WHERE path LIKE
''$binding.transportType%'' ) ))
 AND (binding.id IN ( SELECT classifiedObject FROM Classification WHERE
classificationNode IN (  SELECT id
 FROM ClassificationNode WHERE path LIKE
''$binding.soapStyleType%'' ) ))

 AND ($considerPortType = 0 OR (
   implements.sourceObject=binding.id AND
implements.associationType=''urn:oasis:names:tc:ebxml-
regrep:AssociationType:Implements'' AND implements.targetObject IN
   (
    SELECT DISTINCT portType.id from ExtrinsicObject portType, Name_
portTypeName, Description portTypeDesc, Slot portTypeTNS
    WHERE
     portType.objectType = ''urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType''
     AND (portTypeName.parent = portType.id AND UPPER
( portTypeName.value ) LIKE UPPER ( ''$portType.name'' ) )
     AND (portTypeDesc.parent = portType.id AND UPPER
( portTypeDesc.value ) LIKE UPPER ( ''$portType.description'' ) )
     AND (portType.id = s.parent
      AND portTypeTNS.name_ = ''urn:oasis:names:tc:ebxml-
regrep:profile:ws:wsdl:targetNamespace''
```

```
      AND portTypeTNS.value LIKE ''$portType.targetNamespace'')
   ))
 )
       </rim:QueryExpression>
    </rim:AdhocQuery>
```

Listing 20: WSDL Binding Discovery Query

11.3.4 WSDL Port Discovery Query

```
    <!--
    Parameterized Adhoc Query for WSDL Port Discovery query.
    -->
    <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:PortDiscoveryQuery"
id="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:PortDiscoveryQuery">
      <rim:Name>
        <rim:LocalizedString value="label.PortDiscoveryQuery"/>
      </rim:Name>
      <rim:Description>
        <rim:LocalizedString value="label.PortDiscoveryQuery.desc"/>
      </rim:Description>
      <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
regrep:QueryLanguage:SQL-92">
SELECT DISTINCT port.* FROM
  ServiceBinding port, Name_ portName, Description portDesc, Slot
portTNS,
  Association implements
WHERE
 (port.id IN ( SELECT classifiedObject FROM Classification WHERE
classificationNode = ''urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Port'' ) )

 AND (portName.parent = port.id AND UPPER ( portName.value ) LIKE UPPER
( ''$port.name'' ) )
 AND (portDesc.parent = port.id AND UPPER ( portDesc.value ) LIKE UPPER
( ''$port.description'' ) )
 AND (port.id = s.parent
  AND portTNS.name_ = ''urn:oasis:names:tc:ebxml-
regrep:profile:ws:wsdl:targetNamespace''
  AND portTNS.value LIKE ''$port.targetNamespace'')
 AND (port.accessURI LIKE ''$port.accessURI'')

 AND ($considerBinding = 0 OR (
   implements.sourceObject=port.id AND
implements.associationType=''urn:oasis:names:tc:ebxml-
regrep:AssociationType:Implements'' AND implements.targetObject IN
   (
    SELECT DISTINCT binding.id FROM
      ExtrinsicObject binding, Name_ bindingName, Description
bindingDesc, Slot bindingTNS,
      Association implements
    WHERE
     binding.objectType = ''urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding''
     AND (bindingName.parent = binding.id AND UPPER
( bindingName.value ) LIKE UPPER ( ''$binding.name'' ) )
     AND (bindingDesc.parent = binding.id AND UPPER
( bindingDesc.value ) LIKE UPPER ( ''$binding.description'' ) )
     AND (binding.id = s.parent
      AND bindingTNS.name_ = ''urn:oasis:names:tc:ebxml-
regrep:profile:ws:wsdl:targetNamespace''
      AND bindingTNS.value LIKE ''$binding.targetNamespace'')
     AND (binding.id IN ( SELECT classifiedObject FROM Classification
WHERE classificationNode IN (  SELECT id
     FROM ClassificationNode WHERE path LIKE ''$binding.protocolType'' )
))
     AND (binding.id IN ( SELECT classifiedObject FROM Classification
WHERE classificationNode IN (  SELECT id
     FROM ClassificationNode WHERE path LIKE ''$binding.transportType%''
) ))
```

```
     AND (binding.id IN ( SELECT classifiedObject FROM Classification
WHERE classificationNode IN (  SELECT id
     FROM ClassificationNode WHERE path LIKE ''$binding.soapStyleType%''
) ))

     AND ($considerPortType = 0 OR (
       implements.sourceObject=binding.id AND
implements.associationType=''urn:oasis:names:tc:ebxml-
regrep:AssociationType:Implements'' AND implements.targetObject IN
       (
        SELECT DISTINCT portType.id from ExtrinsicObject portType, Name_
portTypeName, Description portTypeDesc, Slot portTypeTNS
        WHERE
         portType.objectType = ''urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType''
         AND (portTypeName.parent = portType.id AND UPPER
( portTypeName.value ) LIKE UPPER ( ''$portType.name'' ) )
         AND (portTypeDesc.parent = portType.id AND UPPER
( portTypeDesc.value ) LIKE UPPER ( ''$portType.description'' ) )
         AND (portType.id = s.parent
          AND portTypeTNS.name_ = ''urn:oasis:names:tc:ebxml-
regrep:profile:ws:wsdl:targetNamespace''
          AND portTypeTNS.value LIKE ''$portType.targetNamespace'')
       )
     ))
   ))
 )
        </rim:QueryExpression>
    </rim:AdhocQuery>
```

Listing 21: WSDL Port Discovery Query

11.3.5 WSDL Service Discovery Query

```
    <!--
    Parameterized Adhoc Query for WSDL Service Discovery query.
    -->
    <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:ServiceDiscoveryQuery"
id="urn:oasis:names:tc:ebxml-
regrep:profile:ws:query:ServiceDiscoveryQuery">
      <rim:Name>
        <rim:LocalizedString value="label.ServiceDiscoveryQuery"/>
      </rim:Name>
      <rim:Description>
        <rim:LocalizedString value="label.ServiceDiscoveryQuery.desc"/>
      </rim:Description>
      <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
regrep:QueryLanguage:SQL-92">
SELECT DISTINCT service.* FROM
  Service service, Name_ serviceName, Description serviceDesc, Slot
serviceTNS,
  ServiceBinding port
WHERE
 (service.id IN ( SELECT classifiedObject FROM Classification WHERE
classificationNode = ''urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Port'' ) )

 AND (serviceName.parent = service.id AND UPPER ( serviceName.value )
LIKE UPPER ( ''$service.name'' ) )
 AND (serviceDesc.parent = service.id AND UPPER ( serviceDesc.value )
LIKE UPPER ( ''$service.description'' ) )
 AND (service.id = s.parent
  AND serviceTNS.name_ = ''urn:oasis:names:tc:ebxml-
regrep:profile:ws:wsdl:targetNamespace''
  AND serviceTNS.value LIKE ''$service.targetNamespace'')

 AND ($considerPort = 0 OR (
  service.id = port.service AND port.id IN
  (

    SELECT DISTINCT port.id FROM
```

```
      ServiceBinding port, Name_ portName, Description portDesc, Slot
portTNS,
      Association implements
    WHERE
     (port.id IN ( SELECT classifiedObject FROM Classification WHERE
classificationNode = ''urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Port'' ) )

     AND (portName.parent = port.id AND UPPER ( portName.value ) LIKE
UPPER ( ''$port.name'' ) )
     AND (portDesc.parent = port.id AND UPPER ( portDesc.value ) LIKE
UPPER ( ''$port.description'' ) )
     AND (port.id = s.parent
      AND portTNS.name_ = ''urn:oasis:names:tc:ebxml-
regrep:profile:ws:wsdl:targetNamespace''
      AND portTNS.value LIKE ''$port.targetNamespace'')
     AND (port.accessURI LIKE ''$port.accessURI'')

     AND ($considerBinding = 0 OR (
       implements.sourceObject=port.id AND
implements.associationType=''urn:oasis:names:tc:ebxml-
regrep:AssociationType:Implements'' AND implements.targetObject IN
       (
        SELECT DISTINCT binding.id FROM
          ExtrinsicObject binding, Name_ bindingName, Description
bindingDesc, Slot bindingTNS,
          Association implements
        WHERE
         binding.objectType = ''urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding''
         AND (bindingName.parent = binding.id AND UPPER
( bindingName.value ) LIKE UPPER ( ''$binding.name'' ) )
         AND (bindingDesc.parent = binding.id AND UPPER
( bindingDesc.value ) LIKE UPPER ( ''$binding.description'' ) )
         AND (binding.id = s.parent
          AND bindingTNS.name_ = ''urn:oasis:names:tc:ebxml-
regrep:profile:ws:wsdl:targetNamespace''
          AND bindingTNS.value LIKE ''$binding.targetNamespace'')
         AND (binding.id IN ( SELECT classifiedObject FROM
Classification WHERE classificationNode IN (  SELECT id
         FROM ClassificationNode WHERE path LIKE
''$binding.protocolType'' ) ))
         AND (binding.id IN ( SELECT classifiedObject FROM
Classification WHERE classificationNode IN (  SELECT id
         FROM ClassificationNode WHERE path LIKE
''$binding.transportType%'' ) ))
         AND (binding.id IN ( SELECT classifiedObject FROM
Classification WHERE classificationNode IN (  SELECT id
         FROM ClassificationNode WHERE path LIKE
''$binding.soapStyleType%'' ) ))

         AND ($considerBinding = 0 OR (
           implements.sourceObject=binding.id AND
implements.associationType=''urn:oasis:names:tc:ebxml-
regrep:AssociationType:Implements'' AND implements.targetObject IN
           (
            SELECT DISTINCT portType.id from ExtrinsicObject portType,
Name_ portTypeName, Description portTypeDesc, Slot portTypeTNS
            WHERE
             portType.objectType = ''urn:oasis:names:tc:ebxml-
regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType''
             AND (portTypeName.parent = portType.id AND UPPER
( portTypeName.value ) LIKE UPPER ( ''$portType.name'' ) )
             AND (portTypeDesc.parent = portType.id AND UPPER
( portTypeDesc.value ) LIKE UPPER ( ''$portType.description'' ) )
             AND (portType.id = s.parent
              AND portTypeTNS.name_ = ''urn:oasis:names:tc:ebxml-
regrep:profile:ws:wsdl:targetNamespace''
              AND portTypeTNS.value LIKE ''$portType.targetNamespace'')
           )
         ))
       )
```

```
      ))
   ))
)
         </rim:QueryExpression>
      </rim:AdhocQuery>
```

Listing 22: WSDL Service Discovery Query

12 References

12.1 Normative References

[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

[ebRR-DPT] Deployment Profile Template For ebXML Registry 3.0 OASIS Specifications V_0.1.2

[ebMS-DPT] Deployment Profile Template For OASIS Specification ebXML Message Service 2.0

[WSDL] WSDL Specification
http://www.w3.org/TR/wsdl

12.2 Informative References

Appendix Anformative

[WSDL-OVW] WSDL Essentials
http://www.developer.com/services/article.php/1602051

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