



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

28		
29	1 Table of Contents.....	2
30	1 Introduction.....	8
31	1.1 Terminology.....	8
32	1.2 Conventions.....	8
33	2 WSDL Information Model Overview.....	10
34	2.1 Element <service>.....	10
35	2.2 Element <port>.....	10
36	2.3 Element <binding>.....	10
37	2.4 Element <portType>.....	10
38	2.5 Element <operation>.....	10
39	2.6 Element <message>.....	10
40	2.7 Element <types>.....	11
41	3 ebXML Registry Overview.....	12
42	3.1 Overview of [ebRIM].....	12
43	3.1.1 RegistryObject.....	13
44	3.1.2 Object Identification.....	13
45	3.1.3 Object Naming and Description.....	14
46	3.1.4 Object Attributes.....	14
47	3.1.4.1 Slot Attributes.....	14
48	3.1.5 Object Classification.....	15
49	3.1.6 Object Association.....	15
50	3.1.7 Object References To Web Content.....	16
51	3.1.8 Object Packaging.....	16
52	3.1.9 Service Description.....	16
53	3.2 Overview of [ebRS].....	16
54	4 Mapping WSDL Information Model to [ebRIM].....	17
55	4.1 wsdl:service → rim:Service Mapping.....	17
56	4.1.1 Attribute id.....	17
57	4.1.2 Element Name.....	18
58	4.1.3 Element Description.....	18
59	4.1.4 Elements Classification.....	18
60	4.1.5 Elements ServiceBinding.....	19
61	4.2 wsdl:port → rim:ServiceBinding Mapping.....	19
62	4.2.1 Attribute id.....	19
63	4.2.2 Element Name.....	19
64	4.2.3 Element Description.....	20
65	4.2.4 Attribute accessURI.....	20
66	4.2.5 Attribute service.....	20
67	4.2.6 Element Classification.....	20
68	4.3 wsdl:binding → rim:ExtrinsicObject Mapping.....	21

69	4.3.1 Attribute objectType.....	21
70	4.3.2 Attribute id.....	21
71	4.3.3 Element Name.....	21
72	4.3.4 Element Description.....	22
73	4.3.5 Element Classification.....	22
74	4.4 wsdl:portType → rim:ExtrinsicObject Mapping.....	23
75	4.4.1 Attribute objectType.....	23
76	4.4.2 Attribute id.....	23
77	4.4.3 Element Name.....	23
78	4.4.4 Element Description.....	24
79	4.5 wsdl:port «wsdl:binding to rim:Association Mapping.....	24
80	4.5.1 Attribute id.....	24
81	4.5.2 Attribute sourceObject.....	24
82	4.5.3 Attribute targetObject.....	24
83	4.5.4 Attribute associationType.....	24
84	4.6 wsdl:binding «wsdl:portType Association Mapping.....	25
85	4.6.1 Attribute id.....	25
86	4.6.2 Attribute sourceObject.....	25
87	4.6.3 Attribute targetObject.....	25
88	4.6.4 Attribute associationType.....	25
89	5 Publishing Profile.....	26
90	5.1 Structure of WSDL Documents.....	26
91	5.1.1 XxInterfaces.wsdl.....	26
92	5.1.2 XxBindings.wsdl.....	26
93	5.1.3 XxServices.wsdl.....	26
94	6 Validation Service Profile.....	27
95	6.1 Invocation Control File.....	27
96	6.2 Business Rules.....	27
97	7 Cataloging Service Profile.....	28
98	7.1 Invocation Control File.....	28
99	7.2 Input Metadata.....	28
100	7.3 Input Content.....	28
101	7.4 Output Metadata.....	29
102	7.4.1 Changes to Input Metadata.....	29
103	7.4.2 wsdl:service → rim:Service.....	29
104	7.4.3 wsdl:port → rim:ServiceBinding.....	29
105	7.4.4 wsdl:binding → rim:ExtrinsicObject.....	29
106	7.4.5 wsdl:portType → rim:ExtrinsicObject.....	29
107	7.4.6 wsdl:port « wsdl:binding Association.....	29
108	7.4.7 wsdl:binding « wsdl:portType Association.....	29
109	8 Discovery Profile.....	31
110	8.1 Overview.....	31

111	8.1.1 Discovery Query Patterns.....	32
112	8.1.2 Parameter \$name.....	32
113	8.1.3 Parameter \$description.....	32
114	8.1.4 Parameter \$targetNamespace.....	32
115	8.1.5 Parameter \$importedNamespace.....	32
116	8.1.6 Example of WSDL Document Discovery Query.....	33
117	8.2 PortType Discovery Query.....	33
118	8.2.1 Parameter \$portType.name.....	33
119	8.2.2 Parameter \$portType.description.....	33
120	8.2.3 Parameter \$portType.targetNamespace.....	33
121	8.2.4 Parameter \$portType.schemaNamespaces.....	33
122	8.2.5 Example of WSDL PortType Discovery Query.....	33
123	8.3 WSDL Binding Discovery Query.....	34
124	8.3.1 Parameter \$binding.name.....	34
125	8.3.2 Parameter \$binding.description.....	34
126	8.3.3 Parameter \$binding.targetNamespace.....	34
127	8.3.4 Parameter \$binding.protocolType.....	34
128	8.3.5 Parameter \$binding.transportType.....	34
129	8.3.6 Parameter \$binding.soapStyle.....	34
130	8.3.7 Parameter \$considerPortType.....	34
131	8.3.8 Parameter \$portType.name.....	34
132	8.3.9 Parameter \$portType.description.....	35
133	8.3.10 Parameter \$portType.targetNamespace.....	35
134	8.3.11 Parameter \$portType.schemaNamespace.....	35
135	8.3.12 Example of WSDL Binding Discovery Query.....	35
136	8.4 WSDL Port Discovery Query.....	35
137	8.4.1 Parameter \$port.name.....	35
138	8.4.2 Parameter \$port.description.....	35
139	8.4.3 Parameter \$port.targetNamespace.....	36
140	8.4.4 Parameter \$port.accessURI.....	36
141	8.4.5 Parameter \$considerBinding.....	36
142	8.4.6 Parameter \$binding.name.....	36
143	8.4.7 Parameter \$binding.description.....	36
144	8.4.8 Parameter \$binding.targetNamespace.....	36
145	8.4.9 Parameter \$binding.protocolType.....	36
146	8.4.10 Parameter \$binding.transportType.....	36
147	8.4.11 Parameter \$binding.soapStyle.....	36
148	8.4.12 Parameter \$considerPortType.....	36
149	8.4.13 Parameter \$portType.name.....	37
150	8.4.14 Parameter \$portType.description.....	37
151	8.4.15 Parameter \$portType.targetNamespace.....	37
152	8.4.16 Parameter \$portType.schemaNamespace.....	37

153	8.4.17 Example of WSDL Port Discovery Query.....	37
154	8.5 WSDL Service Discovery Query.....	37
155	8.5.1 Parameter \$service.name.....	37
156	8.5.2 Parameter \$service.description.....	37
157	8.5.3 Parameter \$service.targetNamespace.....	38
158	8.5.4 Parameter \$considerPort.....	38
159	8.5.5 Parameter \$port.name.....	38
160	8.5.6 Parameter \$port.description.....	38
161	8.5.7 Parameter \$port.targetNamespace.....	38
162	8.5.8 Parameter \$port.accessURI.....	38
163	8.5.9 Parameter \$considerBinding.....	38
164	8.5.10 Parameter \$binding.name.....	38
165	8.5.11 Parameter \$binding.description.....	38
166	8.5.12 Parameter \$binding.targetNamespace.....	38
167	8.5.13 Parameter \$binding.protocolType.....	38
168	8.5.14 Parameter \$binding.transportType.....	39
169	8.5.15 Parameter \$binding.soapStyle.....	39
170	8.5.16 Parameter \$considerPortType.....	39
171	8.5.17 Parameter \$portType.name.....	39
172	8.5.18 Parameter \$portType.description.....	39
173	8.5.19 Parameter \$portType.targetNamespace.....	39
174	8.5.20 Parameter \$portType.schemaNamespace.....	39
175	8.5.21 Example of WSDL Service Discovery Query.....	39
176	9 Event Notification Profile.....	41
177	9.1 Subscribing to a WSDL Document.....	41
178	9.2 Subscribing to PortType changes.....	41
179	9.3 Subscribing to Binding changes.....	42
180	9.4 Subscribing to Port changes.....	42
181	9.5 Subscribing to Service changes.....	42
182	10 Security Profile.....	43
183	10.1 SubjectRole Profile.....	43
184	10.2 SubjectGroup Profile.....	43
185	10.3 AccessControlPolicy Profile.....	43
186	11 Canonical Metadata Definitions.....	44
187	11.1 ObjectType Extensions.....	44
188	11.2 Canonical ClassificationSchemes.....	44
189	11.3 Canonical Queries.....	46
190	11.3.1 WSDL Document Discovery Query.....	46
191	11.3.2 WSDL PortType Discovery Query.....	46
192	11.3.3 WSDL Binding Discovery Query.....	47
193	11.3.4 WSDL Port Discovery Query.....	48
194	11.3.5 WSDL Service Discovery Query.....	49

195 12 References.....52

196 12.1 Normative References.....52

197 12.2 Informative References.....52

198

Illustration Index

Figure 1: ebXML Registry Information Model, High Level Public View.....	12
Figure 2: ebXML Registry Information Model, Inheritance View.....	13

Index of Tables

200

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 artifacts.

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].

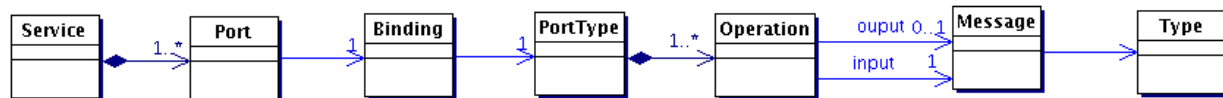


Illustration 1: WSDL Information Model

The WSDL information model is a layered model. At the lowest level is the abstract PortType. The Binding is the next level and it 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 description 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 describes 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.

294 **2.7 Element <types>**

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

3 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.

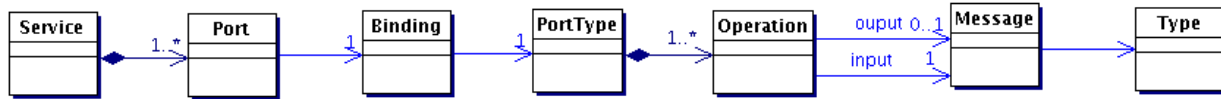


Illustration 2: WSDL Information Model

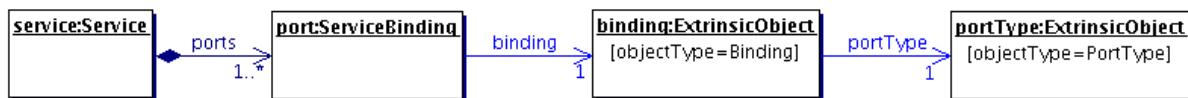


Illustration 3: Mapping of WSDL Information Model to ebRIM

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.

3.1 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

3.1.1 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

3.1.2 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

3.1.3 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

3.1.4 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

3.1.5 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

3.2 wsdl:port → rim:ServiceBinding Mapping

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

3.2.1 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

3.2.2 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

3.2.3 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

3.2.4 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

3.2.5 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.

3.2.6 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

3.3 wsdl:binding → rim:ExtrinsicObject Mapping

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

3.3.1 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

3.3.2 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

3.3.3 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

3.3.4 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

3.3.5 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

3.4 wsdl:portType → rim:ExtrinsicObject Mapping

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

3.4.1 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.

```

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

```

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

3.4.2 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.

```

600 TargetNameSpace= urn:oasis:names:tc:ebxml-
601 regrep:wsdl:registry:interfaces:3.0
602 <portType name="QueryManagerPortType"/>
603
604 <rim:ExtrinsicObject
605   id="urn:oasis:names:tc:ebxml-
606   regrep:wsdl:registry:interfaces:3.0:portType:QueryManagerPortType" ...>

```

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

3.4.3 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.

```

613 <wsdl:portType name="QueryManagerPortType">
614
615   <rim:ExtrinsicObject
616     id="urn:oasis:names:tc:ebxml-
617     regrep:wsdl:registry:interfaces:3.0:portType:QueryManagerPortType">
618     <rim:Name>
619       <rim:LocalizedString value="QueryManagerPortType"/>
620     </rim:Name>
621   </rim:ExtrinsicObject>

```

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

3.4.4 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.

```

629 <wsdl:portType name="QueryManagerPortType">
630   <wsdl:documentation>
631     portType for ebXML Registry QueryManager
632   </wsdl:documentation>
633 </wsdl:portType>
634
635   <rim:ExtrinsicObject
636     id="urn:oasis:names:tc:ebxml-
637     regrep:wsdl:registry:interfaces:3.0:portType:QueryManagerPortType">
638     <rim:Description>

```

```
639      <rim:LocalizedString value="portType for ebXML Registry  
640      QueryManager"/>  
641      </rim:Description>  
642      </rim:ExtrinsicObject>
```

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

644 **3.5 wsdl:port ↔ wsdl:binding to rim:Association Mapping**

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

646 **3.5.1 Attribute id**

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

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

649 **3.5.2 Attribute sourceObject**

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

652 **3.5.3 Attribute targetObject**

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

655 **3.5.4 Attribute associationType**

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

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

658 which is the id of the canonical "Implements" ClassificationNode within the canonical AssociationType
659 ClassificationScheme.

660 **3.6 wsdl:binding ↔ wsdl:portType Association Mapping**

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

662 **3.6.1 Attribute id**

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

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

665 **3.6.2 Attribute sourceObject**

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

668 **3.6.3 Attribute targetObject**

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

671 **3.6.4 Attribute associationType**

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

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

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

4 Publishing Profile

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

4.1 Structure of WSDL Documents

A WSDL description of a web service MAY be contained in a single file. However, to facilitate better reuse, it SHOULD be split into multiple WSDL files as described next. Examples of such suggested WSDL partitioning 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').

4.1.1 XxInterfaces.wsdl

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

4.1.2 XxBindings.wsdl

This file SHOULD contain binding elements.

4.1.3 XxServices.wsdl

This file SHOULD contain the service elements.

5 Validation Service Profile

The ebXML Registry 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].

5.1 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 Business Rules

The following business rules MUST be supported by the WSDL validation service and MUST be able to be expressed 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 Registry 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

796 mapping earlier in this document.

797 **7.4.7 wsdl:binding ↔ wsdl:portType Association**

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

8 Discovery Profile

The ebXML Registry 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 be 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.

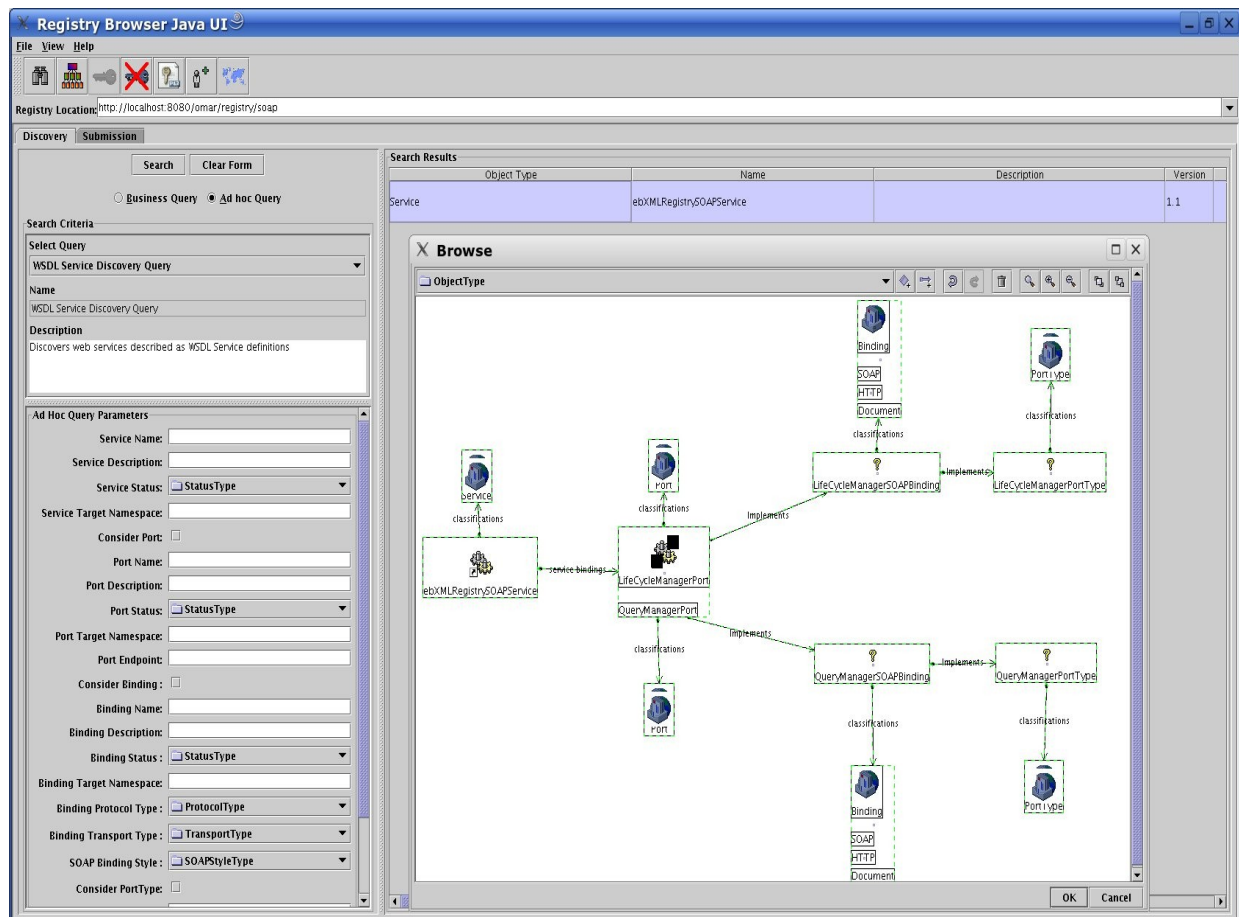


Illustration 4: Example of Parameterized Form for WSDL Service Discovery Query

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.

8.2 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.

8.2.1 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.

8.2.2 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.

8.2.3 Parameter \$targetNamespace

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

8.2.4 Parameter \$importedNamespaces

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

8.2.5 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="$targetNamespace">
      <rim:ValueList>
        <rim:Value>%oasis%</rim:Value>
      </rim:ValueList>
    </rim:Slot>
    <rim:Slot name="$importedNamespaces">
      <rim:ValueList>
        <rim:Value>%org.w3%</rim:Value>
      </rim:ValueList>
    </rim:Slot>
  </rim:AdhocQuery>
</AdhocQueryRequest>
```

Example of WSDL Document Discovery Query

8.3 PortType Discovery Query

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

8.3.1 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.

8.3.2 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.

8.3.3 Parameter \$portType.targetNamespace

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

8.3.4 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.

8.3.5 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

8.4 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.

8.4.1 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.

8.4.2 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.

8.4.3 Parameter \$binding.targetNamespace

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

8.4.4 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.

8.4.5 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.

8.4.6 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.

8.4.7 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".

8.4.8 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.

8.4.9 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.

8.4.10 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.

8.4.11 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.

8.4.12 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>
```

8.5 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.

8.5.1 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.

8.5.2 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.

8.5.3 Parameter \$port.targetNamespace

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

8.5.4 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.

8.5.5 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".

8.5.6 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.

8.5.7 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.

8.5.8 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.

8.5.9 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.

8.5.10 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.

8.5.11 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.

8.5.12 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".

8.5.13 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.

8.5.14 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.

8.5.15 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.

8.5.16 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.

8.5.17 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>
```

```

1052 <rim:Slot name="$port.accessURI">
1053   <rim:ValueList>
1054     <rim:Value>http://acme%</rim:Value>
1055   </rim:ValueList>
1056 </rim:Slot>
1057 </rim:AdhocQuery>
1058 </AdhocQueryRequest>

```

Example of WSDL Port Discovery Query

8.6 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.

8.6.1 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.

8.6.2 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.

8.6.3 Parameter \$service.targetNamespace

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

8.6.4 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".

8.6.5 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.

8.6.6 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.

8.6.7 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.

8.6.8 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.

1087 **8.6.9 Parameter \$considerBinding**

1088 This parameter's value MAY specify a string a "1" or "0" to indicate whether or not to consider the binding
1089 specific parameters that follow when processing the query. If unspecified the value defaults to "0".

1090 **8.6.10 Parameter \$binding.name**

1091 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1092 value of wsdl:binding instances that are used by the objects being discovered.

1093 **8.6.11 Parameter \$binding.description**

1094 This parameter's value MAY specify a string containing a pattern to match against the content of the
1095 wsdl:documentation element within wsdl:binding instances that are used by the objects being discovered.

1096 **8.6.12 Parameter \$binding.targetNamespace**

1097 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1098 of wsdl:binding instances that are used by the objects being discovered.

1099 **8.6.13 Parameter \$binding.protocolType**

1100 This parameter's value MAY specify a string containing a pattern to match against the id attribute value of
1101 the ClassificationNode representing the protocolType supported by wsdl:binding instances that are used
1102 by the objects being discovered.

1103 **8.6.14 Parameter \$binding.transportType**

1104 This parameter's value MAY specify a string containing a pattern to match against the id attribute value of
1105 the ClassificationNode representing the transportType supported by wsdl:binding instances that are used
1106 by the objects being discovered.

1107 **8.6.15 Parameter \$binding.soapStyle**

1108 This parameter's value MAY specify a string containing a pattern to match against the id attribute value of
1109 the ClassificationNode representing the soap style of wsdl:binding instances that are used by the objects
1110 being discovered.

1111 **8.6.16 Parameter \$considerPortType**

1112 This parameter's value MAY specify a string a "1" or "0" to indicate whether or not to consider the
1113 portType specific parameters that follow when processing the query. If unspecified the value defaults to
1114 "0".

1115 **8.6.17 Parameter \$portType.name**

1116 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1117 value of wsdl:portType instances that are used by the objects being discovered.

1118 **8.6.18 Parameter \$portType.description**

1119 This parameter's value MAY specify a string containing a pattern to match against the content of the
1120 wsdl:documentation element within wsdl:portType instances that are used by the objects being
1121 discovered.

8.6.19 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.

8.6.20 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.

8.6.21 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 criteria. 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 Subscriptions 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 1: 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 documents 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 2: 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 3: 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 4: 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 5: 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 AccessControlPolicy 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??

Need to specify external link mapping in mapping chapter??

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 6: Example of Subscription to WSDL Ports

1314

1315 11.2 Canonical ClassificationSchemes

1316 The following new canonical ClassificationSchemes are described by this profile:

```

1317     <rim:ClassificationScheme lid="urn:oasis:names:tc:ebxml-
1318     regrep:profile:ws:classificationScheme:ProtocolType"
1319     id="urn:oasis:names:tc:ebxml-
1320     regrep:profile:ws:classificationScheme:ProtocolType" isInternal="true"
1321     nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode">
1322         <rim:Name>
1323             <rim:LocalizedString charset="UTF-8" value="label.ProtocolType"/>
1324         </rim:Name>
1325         <rim:Description>
1326             <rim:LocalizedString charset="UTF-8"
1327 value="label.ProtocolType.desc"/>
1328         </rim:Description>
1329         <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1330     regrep:profile:ws:ProtocolType:SOAP" code="SOAP"
1331     id="urn:oasis:names:tc:ebxml-regrep:profile:ws:ProtocolType:SOAP"/>
1332         <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1333     regrep:profile:ws:ProtocolType:AS2" code="AS2"
1334     id="urn:oasis:names:tc:ebxml-regrep:profile:ws:ProtocolType:AS2"/>
1335         <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1336     regrep:profile:ws:ProtocolType:Atom" code="Atom"
1337     id="urn:oasis:names:tc:ebxml-regrep:profile:ws:ProtocolType:Atom"/>
1338     </rim:ClassificationScheme>
1339
1340     <rim:ClassificationScheme lid="urn:oasis:names:tc:ebxml-
1341     regrep:profile:ws:classificationScheme:TransportType"
1342     id="urn:oasis:names:tc:ebxml-
1343     regrep:profile:ws:classificationScheme:TransportType" isInternal="true"
1344     nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode">
1345         <rim:Name>
1346             <rim:LocalizedString charset="UTF-8" value="label.TransportType"/>
1347     >
1348         </rim:Name>
1349         <rim:Description>
1350             <rim:LocalizedString charset="UTF-8"
1351 value="label.TransportType.desc"/>
1352         </rim:Description>
1353         <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1354     regrep:profile:ws:TransportType:HTTP" code="HTTP"
1355     id="urn:oasis:names:tc:ebxml-regrep:profile:ws:TransportType:HTTP"/>
1356         <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1357     regrep:profile:ws:TransportType:MOM" code="MOM"
1358     id="urn:oasis:names:tc:ebxml-regrep:profile:ws:TransportType:MOM"/>
1359         <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1360     regrep:profile:ws:TransportType:BEEP" code="BEEP"
1361     id="urn:oasis:names:tc:ebxml-regrep:profile:ws:TransportType:BEEP"/>
1362     </rim:ClassificationScheme>
1363
1364     <rim:ClassificationScheme lid="urn:oasis:names:tc:ebxml-
1365     regrep:profile:ws:classificationScheme:SOAPStyleType"
1366     id="urn:oasis:names:tc:ebxml-
1367     regrep:profile:ws:classificationScheme:SOAPStyleType" isInternal="true"
1368     nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode">
1369         <rim:Name>
1370             <rim:LocalizedString charset="UTF-8" value="label.SOAPStyleType"/>
1371     >
1372         </rim:Name>
1373         <rim:Description>

```

```

1374         <rim:LocalizedString charset="UTF-8" value="label.SOAPType.desc"/>
1375     >
1376         </rim:Description>
1377         <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1378 regrep:profile:wsPStyleType:RPC" code="RPC" id="urn:oasis:names:tc:ebxml-
1379 regrep:profile:wsPStyleType:RPC"/>
1380         <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1381 regrep:profile:wsPStyleType:Document" code="Document"
1382 id="urn:oasis:names:tc:ebxml-regrep:profile:wsPStyleType:Document"/>
1383     </rim:ClassificationScheme>

```

Listing 7: 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

```

1389     <!--
1390     Parameterized Adhoc Query for WSDL Discovery query.
1391     Finds by Name, Description, ComponentType, ResourceType
1392     -->
1393     <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
1394 regrep:profile:ws:query:WSDLDiscoveryQuery" id="urn:oasis:names:tc:ebxml-
1395 regrep:profile:ws:query:WSDLDiscoveryQuery">
1396         <rim:Name>
1397             <rim:LocalizedString value="label.WSDLDiscoveryQuery"/>
1398         </rim:Name>
1399         <rim:Description>
1400             <rim:LocalizedString value="label.WSDLDiscoveryQuery.desc"/>
1401         </rim:Description>
1402         <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1403 regrep:QueryLanguage:SQL-92">
1404             SELECT DISTINCT ro.* FROM RegistryObject ro, Name_ nm, Description d,
1405             Slot tns, Slot ins
1406             WHERE (1=1)
1407                 AND (ro.objectType = 'urn:oasis:names:tc:ebxml-
1408 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL')
1409                 AND (nm.parent = ro.id AND UPPER ( nm.value ) LIKE UPPER ( '$name' ) )
1410                 AND (d.parent = ro.id AND UPPER ( d.value ) LIKE UPPER
1411 ( '$description' ) )
1412                 AND (ro.id = tns.parent
1413                 AND tns.name_ = 'urn:oasis:names:tc:ebxml-
1414 regrep:profile:ws:wSDL:targetNamespace')
1415                 AND tns.value LIKE '$targetNamespace')
1416                 AND (ro.id = ins.parent
1417                 AND ins.name_ = 'urn:oasis:names:tc:ebxml-
1418 regrep:profile:ws:wSDL:importedNamespaces')
1419                 AND ins.value LIKE '$importedNamespaces')
1420
1421         </rim:QueryExpression>
1422     </rim:AdhocQuery>

```

Listing 8: WSDL Document Discovery Query

11.3.2 WSDL PortType Discovery Query

```

1425     <!--
1426     Parameterized Adhoc Query for WSDL PortType Discovery query.
1427     -->

```

```

1428     <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
1429 regrep:profile:ws:query:PortTypeDiscoveryQuery"
1430 id="urn:oasis:names:tc:ebxml-
1431 regrep:profile:ws:query:PortTypeDiscoveryQuery">
1432     <rim:Name>
1433         <rim:LocalizedString value="label.PortTypeDiscoveryQuery"/>
1434     </rim:Name>
1435     <rim:Description>
1436         <rim:LocalizedString value="label.PortTypeDiscoveryQuery.desc"/>
1437     </rim:Description>
1438     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1439 regrep:QueryLanguage:SQL-92">
1440 SELECT DISTINCT portType.* FROM ExtrinsicObject portType, Name_
1441 portTypeName, Description portTypeDesc, Slot portTypeTNS
1442 WHERE
1443     portType.objectType = 'urn:oasis:names:tc:ebxml-
1444 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType'
1445     AND (portTypeName.parent = portType.id AND UPPER ( portTypeName.value )
1446     LIKE UPPER ( '$portType.name' ) )
1447     AND (portTypeDesc.parent = portType.id AND UPPER ( portTypeDesc.value )
1448     LIKE UPPER ( '$portType.description' ) )
1449     AND (portType.id = s.parent
1450     AND portTypeTNS.name_ = 'urn:oasis:names:tc:ebxml-
1451 regrep:profile:ws:wSDL:targetNamespace'
1452     AND portTypeTNS.value LIKE '$portType.targetNamespace')
1453     </rim:QueryExpression>
1454 </rim:AdhocQuery>

```

Listing 9: WSDL PortType Discovery Query

11.3.3 WSDL Binding Discovery Query

```

1457     <!--
1458     Parameterized Adhoc Query for WSDL Binding Discovery query.
1459     -->
1460     <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
1461 regrep:profile:ws:query:BindingDiscoveryQuery"
1462 id="urn:oasis:names:tc:ebxml-
1463 regrep:profile:ws:query:BindingDiscoveryQuery">
1464     <rim:Name>
1465         <rim:LocalizedString value="label.BindingDiscoveryQuery"/>
1466     </rim:Name>
1467     <rim:Description>
1468         <rim:LocalizedString value="label.BindingDiscoveryQuery.desc"/>
1469     </rim:Description>
1470     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1471 regrep:QueryLanguage:SQL-92">
1472 SELECT DISTINCT binding.* FROM
1473     ExtrinsicObject binding, Name_ bindingName, Description bindingDesc,
1474     Slot bindingTNS,
1475     Association implements
1476 WHERE
1477     binding.objectType = 'urn:oasis:names:tc:ebxml-
1478 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding'
1479     AND (bindingName.parent = binding.id AND UPPER ( bindingName.value )
1480     LIKE UPPER ( '$binding.name' ) )
1481     AND (bindingDesc.parent = binding.id AND UPPER ( bindingDesc.value )
1482     LIKE UPPER ( '$binding.description' ) )
1483     AND (binding.id = s.parent
1484     AND bindingTNS.name_ = 'urn:oasis:names:tc:ebxml-
1485 regrep:profile:ws:wSDL:targetNamespace'
1486     AND bindingTNS.value LIKE '$binding.targetNamespace')
1487     AND (binding.id IN ( SELECT classifiedObject FROM Classification WHERE
1488     classificationNode IN ( SELECT id

```

```

1489 FROM ClassificationNode WHERE path LIKE '$binding.protocolType' ) )
1490 AND (binding.id IN ( SELECT classifiedObject FROM Classification WHERE
1491 classificationNode IN ( SELECT id
1492 FROM ClassificationNode WHERE path LIKE '$binding.transportType%' ) ) )
1493 AND (binding.id IN ( SELECT classifiedObject FROM Classification WHERE
1494 classificationNode IN ( SELECT id
1495 FROM ClassificationNode WHERE path LIKE '$binding.soapStyleType%' ) ) )
1496
1497 AND ($considerPortType = 0 OR (
1498     implements.sourceObject=binding.id AND
1499     implements.associationType='urn:oasis:names:tc:ebxml-
1500     regrep:AssociationType:Implements' AND implements.targetObject IN
1501     (
1502         SELECT DISTINCT portType.id from ExtrinsicObject portType, Name_
1503         portTypeName, Description portTypeDesc, Slot portTypeTNS
1504         WHERE
1505             portType.objectType = 'urn:oasis:names:tc:ebxml-
1506             regrep:Object:RegistryObject:ExtrinsicObject:WSDL:PortType'
1507             AND (portTypeName.parent = portType.id AND UPPER
1508             ( portTypeName.value ) LIKE UPPER ( '$portType.name' ) )
1509             AND (portTypeDesc.parent = portType.id AND UPPER
1510             ( portTypeDesc.value ) LIKE UPPER ( '$portType.description' ) )
1511             AND (portType.id = s.parent
1512             AND portTypeTNS.name_ = 'urn:oasis:names:tc:ebxml-
1513             regrep:profile:ws:wsdl:targetNamespace'
1514             AND portTypeTNS.value LIKE '$portType.targetNamespace')
1515         ) )
1516     )
1517     </rim:QueryExpression>
1518 </rim:AdhocQuery>

```

Listing 10: WSDL Binding Discovery Query

11.3.4 WSDL Port Discovery Query

```

1521 <!--
1522 Parameterized Adhoc Query for WSDL Port Discovery query.
1523 -->
1524 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
1525 regrep:profile:ws:query:PortDiscoveryQuery" id="urn:oasis:names:tc:ebxml-
1526 regrep:profile:ws:query:PortDiscoveryQuery">
1527     <rim:Name>
1528         <rim:LocalizedString value="label.PortDiscoveryQuery"/>
1529     </rim:Name>
1530     <rim:Description>
1531         <rim:LocalizedString value="label.PortDiscoveryQuery.desc"/>
1532     </rim:Description>
1533     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1534 regrep:QueryLanguage:SQL-92">
1535 SELECT DISTINCT port.* FROM
1536     ServiceBinding port, Name_ portName, Description portDesc, Slot
1537     portTNS,
1538     Association implements
1539 WHERE
1540     (port.id IN ( SELECT classifiedObject FROM Classification WHERE
1541 classificationNode = 'urn:oasis:names:tc:ebxml-
1542 regrep:Object:RegistryObject:ExtrinsicObject:WSDL:Port' ) )
1543
1544     AND (portName.parent = port.id AND UPPER ( portName.value ) LIKE UPPER (
1545 '$port.name' ) )
1546     AND (portDesc.parent = port.id AND UPPER ( portDesc.value ) LIKE UPPER (
1547 '$port.description' ) )
1548     AND (port.id = s.parent

```

```

1549     AND portTNS.name_ = 'urn:oasis:names:tc:ebxml-
1550 regrep:profile:ws:wsdl:targetNamespace'
1551     AND portTNS.value LIKE '$port.targetNamespace')
1552     AND (port.accessURI LIKE '$port.accessURI')
1553
1554     AND ($considerBinding = 0 OR (
1555         implements.sourceObject=port.id AND
1556         implements.associationType='urn:oasis:names:tc:ebxml-
1557 regrep:AssociationType:Implements' AND implements.targetObject IN
1558         (
1559             SELECT DISTINCT binding.id FROM
1560             ExtrinsicObject binding, Name_ bindingName, Description
1561             bindingDesc, Slot bindingTNS,
1562             Association implements
1563             WHERE
1564             binding.objectType = 'urn:oasis:names:tc:ebxml-
1565 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding'
1566             AND (bindingName.parent = binding.id AND UPPER ( bindingName.value )
1567             LIKE UPPER ( '$binding.name' ) )
1568             AND (bindingDesc.parent = binding.id AND UPPER ( bindingDesc.value )
1569             LIKE UPPER ( '$binding.description' ) )
1570             AND (binding.id = s.parent
1571             AND bindingTNS.name_ = 'urn:oasis:names:tc:ebxml-
1572 regrep:profile:ws:wsdl:targetNamespace'
1573             AND bindingTNS.value LIKE '$binding.targetNamespace')
1574             AND (binding.id IN ( SELECT classifiedObject FROM Classification
1575             WHERE classificationNode IN ( SELECT id
1576             FROM ClassificationNode WHERE path LIKE
1577             '$binding.protocolType' ) ) )
1578             AND (binding.id IN ( SELECT classifiedObject FROM Classification
1579             WHERE classificationNode IN ( SELECT id
1580             FROM ClassificationNode WHERE path LIKE '$binding.transportType
1581             %' ) ) )
1582             AND (binding.id IN ( SELECT classifiedObject FROM Classification
1583             WHERE classificationNode IN ( SELECT id
1584             FROM ClassificationNode WHERE path LIKE '$binding.soapStyleType
1585             %' ) ) )
1586
1587         AND ($considerPortType = 0 OR (
1588             implements.sourceObject=binding.id AND
1589             implements.associationType='urn:oasis:names:tc:ebxml-
1590 regrep:AssociationType:Implements' AND implements.targetObject IN
1591             (
1592                 SELECT DISTINCT portType.id from ExtrinsicObject portType, Name_
1593                 portTypeName, Description portTypeDesc, Slot portTypeTNS
1594                 WHERE
1595                 portType.objectType = 'urn:oasis:names:tc:ebxml-
1596 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType'
1597                 AND (portTypeName.parent = portType.id AND UPPER
1598                 ( portTypeName.value ) LIKE UPPER ( '$portType.name' ) )
1599                 AND (portTypeDesc.parent = portType.id AND UPPER
1600                 ( portTypeDesc.value ) LIKE UPPER ( '$portType.description' ) )
1601                 AND (portType.id = s.parent
1602                 AND portTypeTNS.name_ = 'urn:oasis:names:tc:ebxml-
1603 regrep:profile:ws:wsdl:targetNamespace'
1604                 AND portTypeTNS.value LIKE '$portType.targetNamespace')
1605             )
1606             ) )
1607         ) )
1608     )
1609     </rim:QueryExpression>
1610 </rim:AdhocQuery>

```

Listing 11: 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 ( '$serviceName' ) )
  AND (serviceDesc.parent = service.id AND UPPER ( serviceDesc.value )
LIKE UPPER ( '$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
```



```

1675         ExtrinsicObject binding, Name_ bindingName, Description
1676 bindingDesc, Slot bindingTNS,
1677         Association implements
1678         WHERE
1679         binding.objectType = 'urn:oasis:names:tc:ebxml-
1680 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding'
1681         AND (bindingName.parent = binding.id AND UPPER
1682 ( bindingName.value ) LIKE UPPER ( '$binding.name' ) )
1683         AND (bindingDesc.parent = binding.id AND UPPER
1684 ( bindingDesc.value ) LIKE UPPER ( '$binding.description' ) )
1685         AND (binding.id = s.parent
1686         AND bindingTNS.name_ = 'urn:oasis:names:tc:ebxml-
1687 regrep:profile:ws:wsdl:targetNamespace'
1688         AND bindingTNS.value LIKE '$binding.targetNamespace')
1689         AND (binding.id IN ( SELECT classifiedObject FROM Classification
1690 WHERE classificationNode IN ( SELECT id
1691 FROM ClassificationNode WHERE path LIKE
1692 '$binding.protocolType' ) ) )
1693         AND (binding.id IN ( SELECT classifiedObject FROM Classification
1694 WHERE classificationNode IN ( SELECT id
1695 FROM ClassificationNode WHERE path LIKE '$binding.transportType
1696 %' ) ) )
1697         AND (binding.id IN ( SELECT classifiedObject FROM Classification
1698 WHERE classificationNode IN ( SELECT id
1699 FROM ClassificationNode WHERE path LIKE '$binding.soapStyleType
1700 %' ) ) )
1701
1702         AND ($considerBinding = 0 OR (
1703         implements.sourceObject=binding.id AND
1704 implements.associationType='urn:oasis:names:tc:ebxml-
1705 regrep:AssociationType:Implements' AND implements.targetObject IN
1706 (
1707         SELECT DISTINCT portType.id from ExtrinsicObject portType,
1708 Name_ portTypeName, Description portTypeDesc, Slot portTypeTNS
1709         WHERE
1710         portType.objectType = 'urn:oasis:names:tc:ebxml-
1711 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType'
1712         AND (portTypeName.parent = portType.id AND UPPER
1713 ( portTypeName.value ) LIKE UPPER ( '$portType.name' ) )
1714         AND (portTypeDesc.parent = portType.id AND UPPER
1715 ( portTypeDesc.value ) LIKE UPPER ( '$portType.description' ) )
1716         AND (portType.id = s.parent
1717         AND portTypeTNS.name_ = 'urn:oasis:names:tc:ebxml-
1718 regrep:profile:ws:wsdl:targetNamespace'
1719         AND portTypeTNS.value LIKE '$portType.targetNamespace')
1720         )
1721         ))
1722     )
1723 ))
1724 ))
1725 )
1726 </rim:QueryExpression>
1727 </rim:AdhocQuery>

```

Listing 12: 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>