



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/>).

28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71

1 Table of Contents

1 Table of Contents.....	2
1 Introduction.....	8
1.1 Terminology.....	8
1.2 Conventions.....	8
2 WSDL Information Model Overview.....	10
2.1 Element <service>.....	10
2.2 Element <port>.....	10
2.3 Element <binding>.....	10
2.4 Element <portType>.....	10
2.5 Element <operation>.....	10
2.6 Element <message>.....	10
2.7 Element <types>.....	11
3 ebXML Registry Overview.....	12
3.1 Overview of [ebRIM].....	12
3.1.1 RegistryObject.....	13
3.1.2 Object Identification.....	13
3.1.3 Object Naming and Description.....	14
3.1.4 Object Attributes.....	14
3.1.4.1 Slot Attributes.....	14
3.1.5 Object Classification.....	15
3.1.6 Object Association.....	15
3.1.7 Object References To Web Content.....	16
3.1.8 Object Packaging.....	16
3.1.9 Service Description.....	16
3.2 Overview of [ebRS].....	16
4 Mapping WSDL Information Model to [ebRIM].....	17
4.1 wsdl:service → rim:Service Mapping.....	17
4.1.1 Attribute id.....	17
4.1.2 Element Name.....	18
4.1.3 Element Description.....	18
4.1.4 Elements Classification.....	18
4.1.5 Elements ServiceBinding.....	19
4.2 wsdl:port → rim:ServiceBinding Mapping.....	19
4.2.1 Attribute id.....	19
4.2.2 Element Name.....	19
4.2.3 Element Description.....	20
4.2.4 Attribute accessURI.....	20
4.2.5 Attribute service.....	20
4.2.6 Element Classification.....	20
4.3 wsdl:binding → rim:ExtrinsicObject Mapping.....	21
4.3.1 Attribute objectType.....	21
4.3.2 Attribute id.....	21
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.schemaNamespaces.....	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

201 1 Introduction

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

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

206 The document is organized as follows:

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

223 1.1 Terminology

224 The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT,
225 RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in IETF
226 RFC 2119 [RFC2119].

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

230 1.2 Conventions

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

- 233 • UML Diagrams

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

- 236 • Identifier Placeholders

237 Listings may contain values that reference ebXML Registry objects by their id attribute. These id
238 values uniquely identify the objects within the ebXML Registry. For convenience and better
239 readability, these key values are replaced by meaningful textual variables to represent such id
240 values.

241 For example, the following placeholder refers to the unique id defined for the canonical
242 ClassificationNode that defines the Organization ObjectType defined in [ebRIM]:

243

244 `<id="{CANONICAL_OBJECT_TYPE_ID_ORGANIZATION}" >`

245 • **Constants**

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

```
250 <rim:ClassificationScheme  
251     lid="urn:oasis:names:tc:ebxml-regrep:classificationScheme:ObjectType"  
252     id="urn:uuid:3188a449-18ac-41fb-be9f-99aladca02cb">
```

253 **1. Example Values**

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

```
256  
257 <rim:Name>  
258     <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>  
259 </rim:Name>
```

260

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].

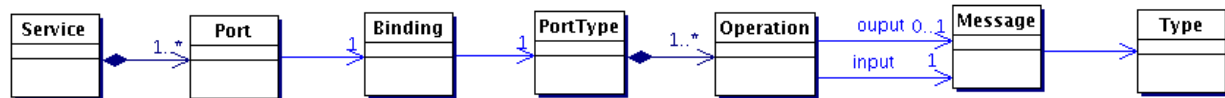


Illustration 1: WSDL Information Model

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 encapsulates one or 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.

2.7 Element <types>

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

295 description.

3 ebXML Registry Overview

296

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

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

300 The specific domain is the source information model.

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

3.1 Overview of [ebRIM]

303

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

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

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

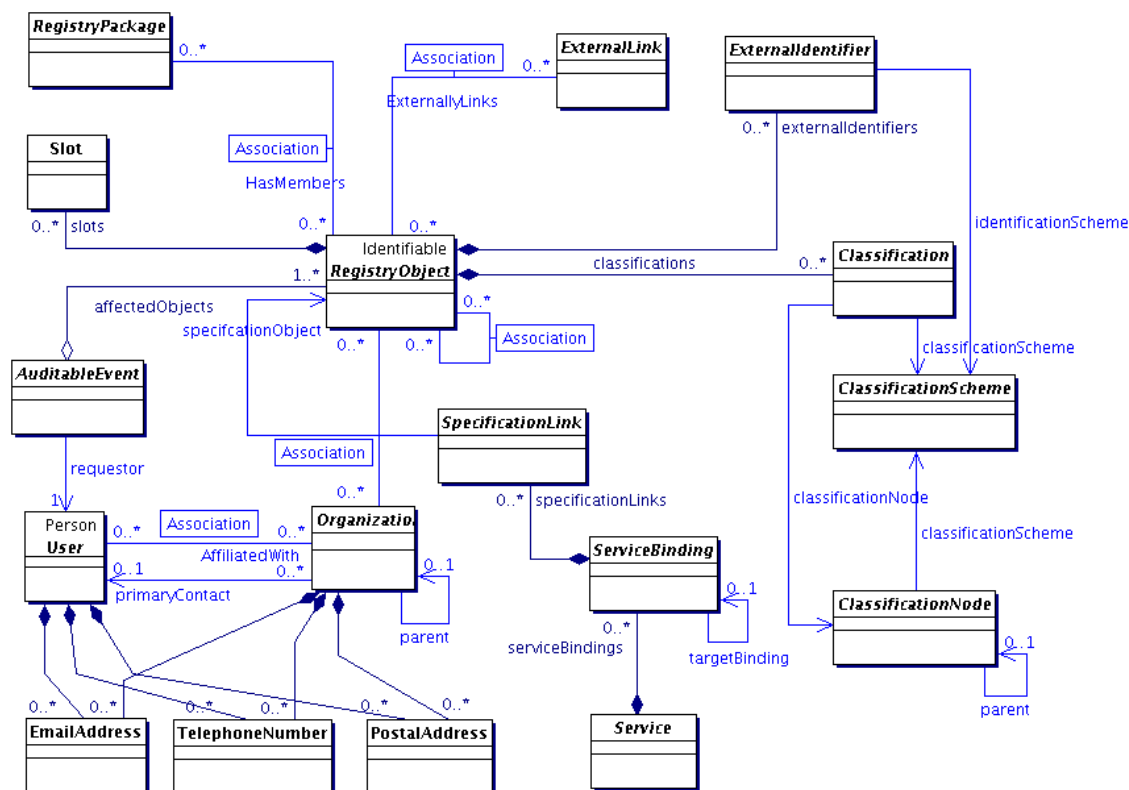


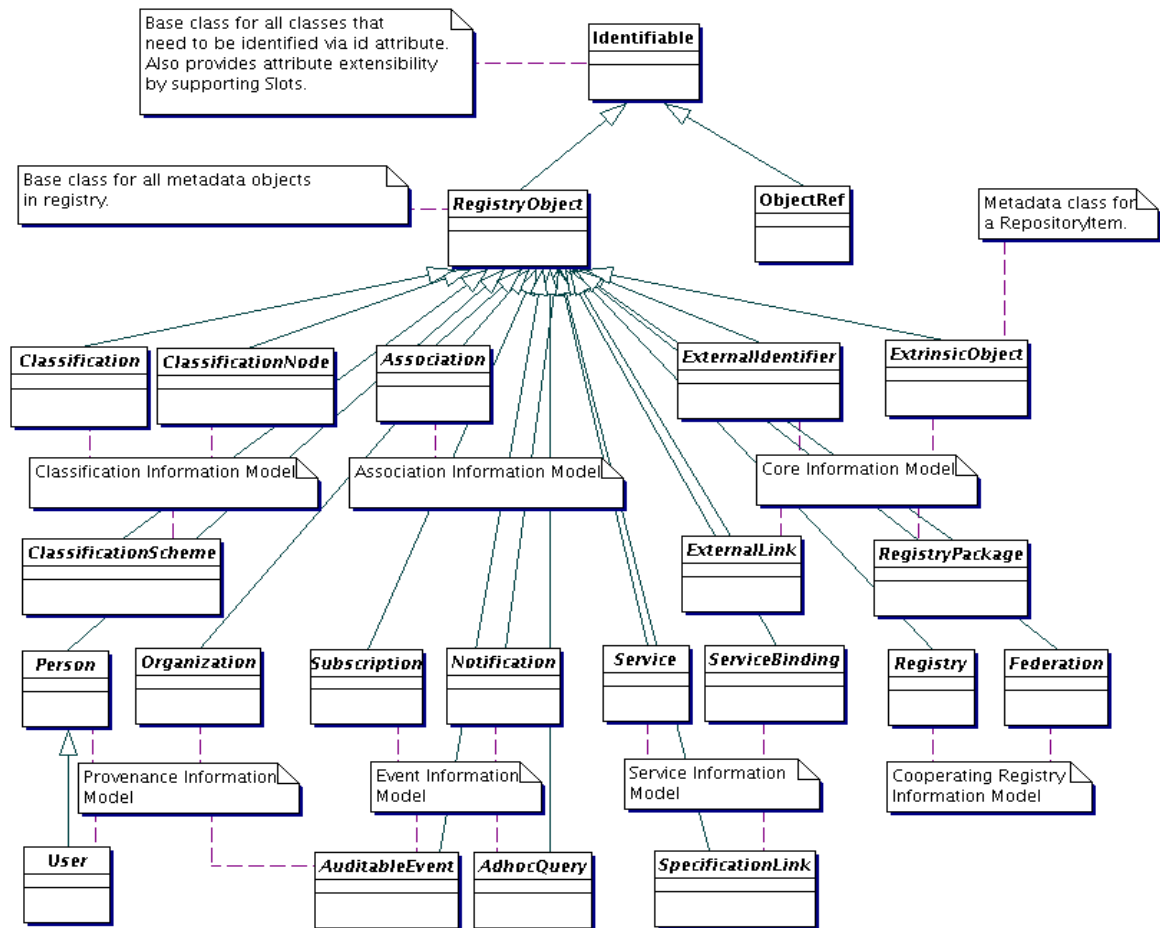
Figure 1: ebXML Registry Information Model, High Level Public View

311

312

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

317



319 **Figure 2: ebXML Registry Information Model, Inheritance View**

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

321 **3.1.1 RegistryObject**

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

326 **3.1.2 Object Identification**

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

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

329 **Listing 1: Example of id attribute**

331 The id attribute value MAY potentially be human friendly.

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

333 **Listing 2: Example of human friendly id attribute**

334 Since a RegistryObject MAY have several versions, a logical id (called lid) is also defined which is

336 unique for different logical objects. However the lid attribute value MUST be the same for all versions of
337 the same logical object. The lid attribute value is a URN that, as well for id attribute, MAY potentially be
338 human friendly:

339

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

342 **Listing 3: Example of lid Attribute**

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

345

```
346 <rim:Organization id=${ACME_ORG_ID}  
347     lid="urn:acme:ACMEOrganization">  
348       
349     <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}  
350         identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}  
351         value="ACME"/>  
352 </rim:ExternalIdentifier>  
353 </rim:Organization>
```

354
355 **Listing 4: Example of ExternalIdentifier**

356 3.1.3 Object Naming and Description

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

359

```
360 <rim:Organization id=${ACME_ORG_ID}  
361     lid="urn:acme:ACMEOrganization">  
362       
363     <rim:Name>  
364     <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>  
365     </rim:Name>  
366     <rim:Description>  
367     <rim:LocalizedString value="ACME is a provider of Java software."  
368         xml:lang="en-US"/>  
369     </rim:Description>  
370       
371     <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}  
372         identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}  
373         value="ACME"/>  
374     </rim:ExternalIdentifier>  
375 </rim:Organization>
```

376
377 **Listing 5: Example of Name and Description**

378 3.1.4 Object Attributes

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

381 3.1.4.1 Slot Attributes

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

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

388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411

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

  <rim:Slot name="urn:acme:slot:NASDAQSymbol">
    <rim:ValueList>
      <rim:Value>ACME</rim:Value>
    </rim:ValueList>
  </rim:Slot>

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

Listing 6: Example of a Dynamic Attribute Using Slot

3.1.5 Object Classification

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

417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443

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

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

Listing 7: Example of Object Classification

3.1.6 Object Association

Any RegistryObject MAY be associated with any other RegistryObject using an Association instance where one object is the sourceObject and the other is the targetObject of the Association instance. An Association instance MAY have an associationType which defines the nature of the association. There are a number of predefined Association Types that a registry must support to be [ebRIM] compliant

449 as shown in Table 1. [ebRIM] allows this list to be extensible.

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

453

```
454 <rim:Association  
455     id=${ASSOCIATION_ID}  
456     associationType=${CANONICAL_ASSOCIATION_TYPE_OFFERS_SERVICE_ID}  
457     sourceObject=${ACME_ORG_ID}  
458     targetObject=${ACME_SERVICE1_ID}/>
```

459

Listing 8: Example of Object Association

460 3.1.7 Object References To Web Content

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

466

```
467 <rim:ExternalLink externalURI="http://www.acme.com"  
468     id=${ACME_WEBSITE_EXTERNAL_ID}>  
469 <rim:Association  
470     id=${EXTERNALLYLINKS_ASSOCIATION_ID}  
471     associationType=${CANONICAL_ASSOCIATION_TYPE_EXTERNALLY_LINKS_ID}  
472     sourceObject=${ACME_WEBSITE_EXTERNAL_ID}  
473     targetObject=${ACME_ORG_ID}/>
```

474

Listing 9: Example of Reference to Web Content Using ExternalLink

475 3.1.8 Object Packaging

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

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

483

```
484 <rim:RegistryPackage  
485     id=${ACME_SERVICES_PACKAGE_ID}>  
486     <rim:RegistryObjectList>  
487         <rim:ObjectRef id=${ACME_SERVICE1_ID}>  
488             <rim:RegistryPackage  
489                 id=${ACME_PURCHASING_SERVICES_PACKAGE_ID}>  
490                 <rim:ObjectRef id=${ACME_PURCHASING_SERVICE1_ID}>  
491                 <rim:ObjectRef id=${ACME_PURCHASING_SERVICE2_ID}>  
492             </rim:RegistryPackage>  
493         <rim:RegistryPackage  
494             id=${ACME_HR_SERVICES_PACKAGE_ID}>  
495             <rim:ObjectRef id=${ACME_HR_SERVICE1_ID}>  
496             <rim:ObjectRef id=${ACME_HR_SERVICE2_ID}>  
497         </rim:RegistryPackage>  
498     </rim:RegistryObjectList>  
499 </rim:RegistryPackage>
```

500

Listing 10: Example of Object Packaging Using RegistryPackages

501 **3.1.9 Service Description**

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

505 **3.2 Overview of [ebRS]**

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

4 Mapping WSDL Information Model to [ebRIM]

508

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

510 The following figures provide a pictorial overview of the type mapping between the two models.

511 Following both figures from left to right, there is a one-to-one correspondence between the two models.

512 The mapping of types between the two models stops at the WSDL PortType.

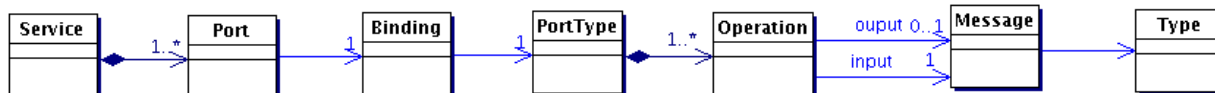


Illustration 2: WSDL Information Model

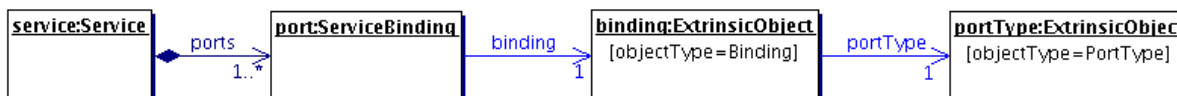


Illustration 3: Mapping of WSDL Information Model to ebRIM

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

4.1 wsdl:service → rim:Service Mapping

518

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

520

```
521 <service name="ebXMLRegistrySOAPService">
522   <port binding="bindings:QueryManagerSOAPBinding"
523   name="QueryManagerPort">
524     <soap:address location="http://your.server.com/soap"/>
525   </port>
526   <port binding="bindings:LifecycleManagerSOAPBinding"
527   name="LifecycleManagerPort">
528     <soap:address location="http://your.server.com/soap"/>
529   </port>
530 </service>
```

531

Example wsdl:service

4.1.1 Attribute id

532

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

536

```
537 targetNamespace: urn:acmeinc:ebxml:registry:3.0:services:wsdl
538
539 WSDL fragment:
540 <wsdl:service name="ebXMLRegistrySOAPService">
541
542 <rim:Service
543   id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:service:
544   ebXMLRegistrySOAPService">
```

545

Example of rim:Service id Attribute Mapping

546 **4.1.2 Element Name**

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

550

```

551 WSDL fragment:
552 <wsdl:service name="ebXMLRegistrySOAPService">
553
554 <rim:Service
555 id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:Service:
556 ebXMLRegistrySOAPService">
557   <rim:Name>
558     <rim:LocalizedString value="ebXMLRegistrySOAPService"/>
559   </rim:Name>
560 </rim:Service>
  
```

561 **Example of rim:Service name Attribute Mapping**

562 **4.1.3 Element Description**

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

567

```

568 WSDL fragment:
569 <wsdl:service name="ebXMLRegistrySOAPService">
570   <wsdl:documentation>
571     An implementation of ebXML Registry 3.0
572   </wsdl:documentation>
573 </wsdl:service>
574
575 <rim:Service
576 id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:Service:
577 ebXMLRegistrySOAPService">
578   <rim:Description>
579     <rim:LocalizedString value="An implementation of ebXML Registry
580 3.0"/>
581   </rim:Description>
582 </rim:Service>
  
```

583 **Example of rim:Service description Attribute Mapping**

584 **4.1.4 Elements Classification**

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

587

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

588

```

589 <rim:Service
  
```

```

590 id="urn:acmeinc:ebxml:registry:3.0:services:wSDL:Service:
591 ebxMLRegistrySOAPService">
592
593     <Classification classificationScheme="urn:oasis:names:tc:ebxml-
594     regrep:ObjectType" classificationNode="urn:oasis:names:tc:ebxml-
595     regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Service"/>
596
597 </rim:Service>

```

598 **Example of rim:Service classifications Attribute Mapping for wSDL:service**

599 4.1.5 Elements ServiceBinding

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

```

603 <rim:Service
604 id="urn:acmeinc:ebxml:registry:3.0:services:wSDL:Service:
605 ebxMLRegistrySOAPService">
606     <rim:ServiceBinding .../>
607     <rim:ServiceBinding .../>
608 </rim:Service>

```

609 **Example of wSDL:port → rim:ServiceBinding Mapping**

610 4.2 wSDL:port → rim:ServiceBinding Mapping

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

612 4.2.1 Attribute id

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

```

617 <wSDL:port binding="bindings:QueryManagerSOAPBinding
618     name="QueryManagerPort">
619
620 <rim:ServiceBinding
621 id="urn:acmeinc:ebxml:registry:3.0:services:wSDL:port:
622 QueryManagerPort">

```

623 **Example of rim:ServiceBinding id Attribute Mapping for wSDL:port**

624 4.2.2 Element Name

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

```

629 <wSDL:port binding="bindings:QueryManagerSOAPBinding
630     name="QueryManagerPort">
631
632 <rim:ServiceBinding
633 id="urn:acmeinc:ebxml:registry:3.0:services:wSDL:port:
634 QueryManagerPort">
635     <rim:Name>
636     <rim:LocalizedString value="QueryManagerPort"/>
637 </rim:Name>

```

638 </rim:ServiceBinding>

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

640 4.2.3 Element Description

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

645

```
646 <wsdl:port binding="bindings:QueryManagerSOAPBinding
647     name="QueryManagerPort">
648     <wsdl:documentation>
649         SOAP Binding implementation of ebXML Registry QueryManager
650     </wsdl:documentation>
651 </wsdl:port>
652
653 <rim:ServiceBinding
654     id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:port:
655     QueryManagerPort">
656     <rim:Description>
657         <rim:LocalizedString value="SOAP Binding implementation of ebXML
658         Registry QueryManager"/>
659     </rim:Description>
660 </rim:ServiceBinding>
```

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

662 4.2.4 Attribute accessURI

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

666

```
667 <wsdl:port binding="bindings:QueryManagerSOAPBinding
668     name="QueryManagerPort">
669     <soap:address location="http://your.server.com/soap"/>
670 </wsdl:port>
671
672 <rim:ServiceBinding
673     id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:port:
674     QueryManagerPort" accessURI="http://your.server.com/soap">
675 </rim:ServiceBinding>
```

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

677 4.2.5 Attribute service

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

680 4.2.6 Element Classification

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

683

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

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

```
<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>
```

700

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

701

4.3 wsdl:binding → rim:ExtrinsicObject Mapping

702

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

703

4.3.1 Attribute objectType

704

705

706

707

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.

708

709

710

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

711

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

712

4.3.2 Attribute id

713

714

715

716

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.

717

718

719

720

721

722

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

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

723

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

724 4.3.3 Element Name

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

728

```
729 <binding name="QueryManagerSOAPBinding"  
730 type="interfaces:QueryManagerPortType">  
731  
732 <rim:ExtrinsicObject  
733 id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:binding:  
734 QueryManagerSOAPBinding">  
735 <rim:Name>  
736 <rim:LocalizedString value="QueryManagerSOAPBinding"/>  
737 </rim:Name>  
738 </rim:ExtrinsicObject>
```

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

740 4.3.4 Element Description

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

745

```
746 <binding name="QueryManagerSOAPBinding"  
747 type="interfaces:QueryManagerPortType">  
748 <wsdl:port binding="bindings:QueryManagerSOAPBinding  
749 name="QueryManagerPort">  
750 <wsdl:documentation>  
751 SOAP Binding for ebXML Registry QueryManager  
752 <wsdl:documentation>  
753 </wsdl:binding>  
754  
755 <rim:ExtrinsicObject  
756 <rim:Description>  
757 <rim:LocalizedString value="SOAP Binding for ebXML Registry  
758 QueryManager"/>  
759 </rim:Description>  
760 </rim:ExtrinsicObject>
```

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

762 4.3.5 Element Classification

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

765

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.

766

767

```

768 <binding name="QueryManagerSOAPBinding"
769   type="interfaces:QueryManagerPortType">
770   <soap:binding style="document"
771   transport="http://schemas.xmlsoap.org/soap/http"/>
772   ...
773 </binding>
774
775 <rim:ExtrinsicObject
776
777   <Classification classificationScheme="urn:oasis:names:tc:ebxml-
778   regrep:profile:ws:classificationScheme:ProtocolType"
779   classificationNode="urn:oasis:names:tc:ebxml-
780   regrep:profile:ws:ProtocolType:SOAP"/>
781
782   <Classification classificationScheme="urn:oasis:names:tc:ebxml-
783   regrep:profile:ws:classificationScheme:TransportType"
784   classificationNode="urn:oasis:names:tc:ebxml-
785   regrep:profile:ws:TransportType:HTTP"/>
786
787   <Classification classificationScheme="urn:oasis:names:tc:ebxml-
788   regrep:profile:ws:classificationScheme:SOAPStyle"
789   classificationNode="urn:oasis:names:tc:ebxml-
790   regrep:profile:ws:SOAPStyle:Document"/>
791
792 </rim:ExtrinsicObject>

```

793

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

794 4.4 wsdl:portType → rim:ExtrinsicObject Mapping

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

797 4.4.1 Attribute objectType

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

```

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

```

805

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

806 4.4.2 Attribute id

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

810

```
811 TargetNameSpace= urn:oasis:names:tc:ebxml-  
812 regrep:wsdl:registry:interfaces:3.0  
813 <portType name="QueryManagerPortType"/>  
814  
815 <rim:ExtrinsicObject  
816   id="urn:oasis:names:tc:ebxml-  
817   regrep:wsdl:registry:interfaces:3.0:portType:QueryManagerPortType" ...>
```

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

819 4.4.3 Element Name

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

823

```
824 <wsdl:portType name="QueryManagerPortType">  
825  
826 <rim:ExtrinsicObject  
827   id="urn:oasis:names:tc:ebxml-  
828   regrep:wsdl:registry:interfaces:3.0:portType:QueryManagerPortType">  
829   <rim:Name>  
830     <rim:LocalizedString value="QueryManagerPortType"/>  
831   </rim:Name>  
832 </rim:ExtrinsicObject>
```

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

834 4.4.4 Element Description

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

839

```
840 <wsdl:portType name="QueryManagerPortType">  
841   <wsdl:documentation>  
842     portType for ebXML Registry QueryManager  
843   </wsdl:documentation>  
844 </wsdl:portType>  
845  
846 <rim:ExtrinsicObject  
847   id="urn:oasis:names:tc:ebxml-  
848   regrep:wsdl:registry:interfaces:3.0:portType:QueryManagerPortType">  
849   <rim:Description>  
850     <rim:LocalizedString value="portType for ebXML Registry  
851   QueryManager"/>  
852   </rim:Description>  
853 </rim:ExtrinsicObject>
```

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

855 **4.5 wsdl:port ↔wsdl:binding to rim:Association Mapping**

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

857 **4.5.1 Attribute id**

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

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

860 **4.5.2 Attribute sourceObject**

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

863 **4.5.3 Attribute targetObject**

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

866 **4.5.4 Attribute associationType**

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

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

869 which is the id of the canonical "Implements" ClassificationNode within the canonical AssociationType
870 ClassificationScheme.

871 **4.6 wsdl:binding ↔wsdl:portType Association Mapping**

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

873 **4.6.1 Attribute id**

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

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

876 **4.6.2 Attribute sourceObject**

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

879 **4.6.3 Attribute targetObject**

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

882 **4.6.4 Attribute associationType**

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

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

885 which is the id of the canonical "Implements" ClassificationNode within the canonical AssociationType
886 ClassificationScheme.

887 **5 Publishing Profile**

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

890 **5.1 Structure of WSDL Documents**

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

895 **5.1.1 XxInterfaces.wsdl**

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

897 **5.1.2 XxBindings.wsdl**

898 This file SHOULD contain binding elements.

899 **5.1.3 XxServices.wsdl**

900 This file SHOULD contain the service elements.

901

6 Validation Service Profile

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

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

908

6.1 Invocation Control File

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

912

6.2 Business Rules

913 The following business rules MUST be supported by the WSDL validation service and MUST be able to
914 be expressed declaratively within the Invocation Control File:

- 915 • Ability to specify that published WSDL documents MUST restrict <wsdl:binding> elements to
916 only use a subset of the ClassificationNodes specified within the WSDLBindingType
917 ClassificationScheme defined by this profile. For example it MUST be possible to express the
918 rule that a binding element MUST only use SOAP binding.
- 919 • Ability to specify that published WSDL documents MUST restrict <soap:binding> style attribute
920 to a subset of the ClassificationNodes specified within the SOAPBindingStyle
921 ClassificationScheme defined by this profile. For example it MUST be possible to express the
922 rule that a SOAP binding MUST only use "document" style.
- 923 • Ability to specify that published WSDL documents MUST restrict <soap:binding> transport
924 attribute to a subset of the ClassificationNodes specified within the TransportType
925 ClassificationScheme defined by this profile. For example it MUST be possible to express the
926 rule that a SOAP binding MUST only use "http" transport.

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

7 Cataloging Service Profile

928

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

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

937 This section describes the cataloging service for cataloging WSDL content.

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

7.1 Invocation Control File

940

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

7.2 Input Metadata

943

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

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

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

954

```
955 <rim:ExtrinsicObject id="urn:acmeinc:ebxml:registry:3.0:services:wSDL">  
956 ...  
957 </rim:ExtrinsicObject>
```

Example of ExtrinsicObject Input Metadata

958

```
959  
960 <rim:ExternalLink  
961 id="urn:acmeinc:ebxml:registry:3.0:services:wSDL"  
962 externalURI="http://www.acme.com/wSDL/ebXMLRegistryService.wSDL"  
963 >  
964 ...  
965 </rim:ExternalLink>
```

Example of ExternalLink Input Metadata

7.3 Input Content

967

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

973 defined in ??.

974 **7.4 Output Metadata**

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

976 **7.4.1 Changes to Input Metadata**

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

979 **Slot importedNameSpaces**

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

984 **Slot targetNamespace**

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

989 **7.4.2 wsdL:service → rim:Service**

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

993 **7.4.3 wsdL:port → rim:ServiceBinding**

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

997 **7.4.4 wsdL:binding → rim:ExtrinsicObject**

998 The WSDL Cataloging service MUST automatically produce an rim:ExtrinsicObject instance for each
999 wsdL:binding element within the input WSDL or its imports, as specified in the wsdL:binding →
1000 rim:ExtrinsicObject mapping earlier in this document.

1001 **7.4.5 wsdL:portType → rim:ExtrinsicObject**

1002 The WSDL Cataloging service MUST automatically produce an rim:ExtrinsicObject instance for each
1003 wsdL:portType element within the input WSDL or its imports, as specified in the wsdL:portType →
1004 rim:ExtrinsicObject mapping earlier in this document.

1005 **7.4.6 wsdL:port ↔ wsdL:binding Association**

1006 The WSDL Cataloging service MUST automatically produce rim:Association instances for each wsdL:port
1007 element within the input WSDL or its imports, as specified in the wsdL:port → wsdL:binding Association
1008 mapping earlier in this document.

1009 **7.4.7 wsdL:binding ↔ wsdL:portType Association**

1010 The WSDL Cataloging service MUST automatically produce rim:Association instances for each
1011 wsdL:binding element within the input WSDL or its imports, as specified in the wsdL:binding →
1012 wsdL:portType Association mapping earlier in this document.

1013

8 Discovery Profile

1014 The ebXML Registry provides the ability for a user defined parameterized queries to be configured for
1015 each type of content. The queries may be as complex or simple as the discovery use case requires. The
1016 complexity of the parameterized queries may hidden from the registry client by storing them within the
1017 ebXML Registry as instances of the AdhocQuery class, and being invoked by simply providing their
1018 parameters. Query parameters are often pattern strings that may contain wildcard characters '%'
1019 (matches any number of characters) and '_' (matches exactly one character) as described by [eBRS].

1020 An ebXML Registry SHOULD provide a graphical user interface that displays any configured
1021 parameterized query as a form which contains an appropriate field for entering each query parameter.
1022

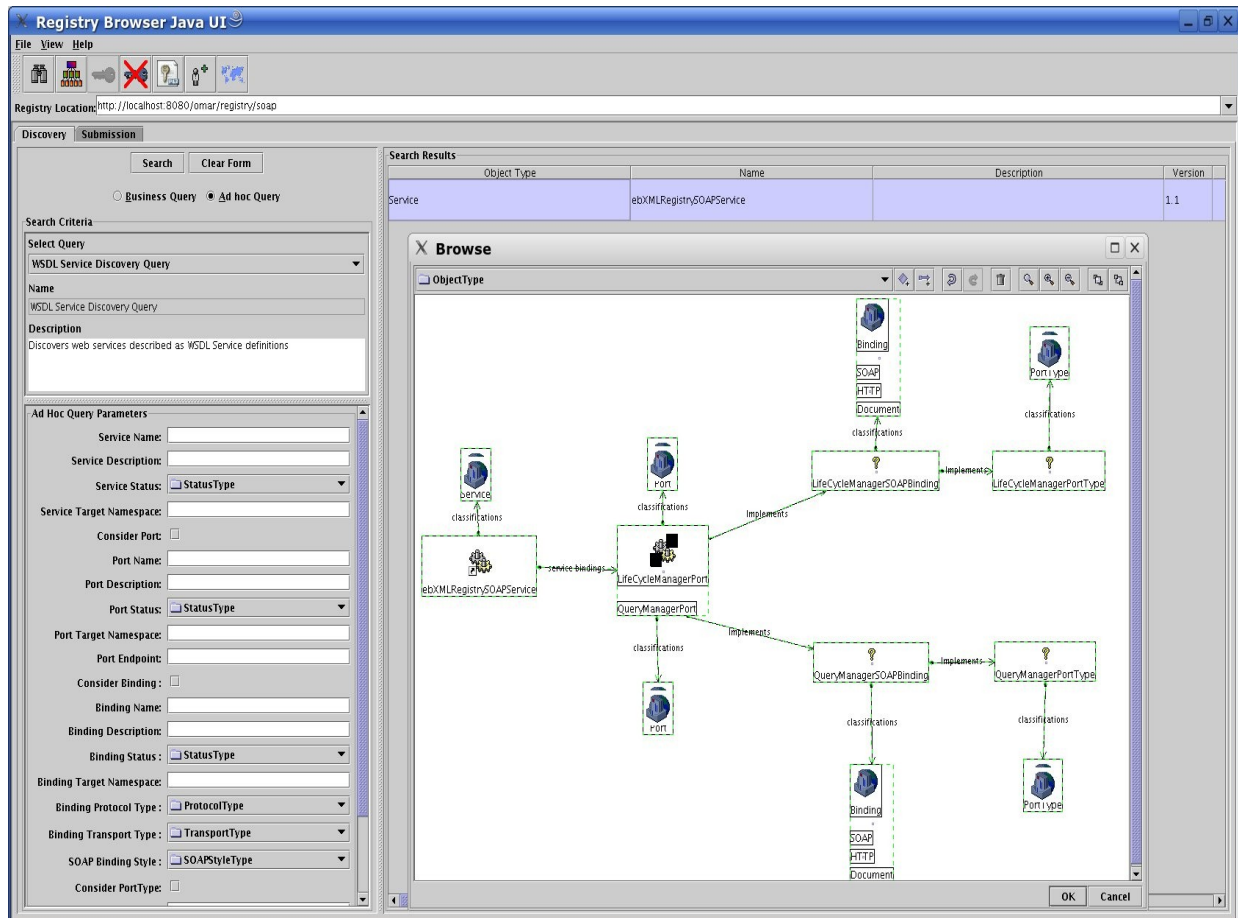


Illustration 4: Example of Parameterized Form for WSDL Service Discovery Query

1024

1025 This chapter defines the queries that MUST be support by an ebXML Registry implementing the WS
1026 Profile for discovering WSDL content. An implementation MAY also support additional discovery queries
1027 for WSDL content.

8.1 Overview

1029 Refer to the layered architecture of WSDL information model described in chapter 2.

1030 Discovery is the process that enables discovering higher level objects based on search criteria that
1031 predicates upon attributes of the type of object being discovered as well. as the attributes of lower level
1032 objects in the model.

1033 In contrast, drill-down is the process that enables explore a higher level object and determining the lower
1034 level objects that were used in building the higher level object.

1035 [ebRIM] provides support for drill-down use cases in a generic manner out-of-box. For example, [ebRIM]
1036 provides:

- 1037 • A generic ability to explore all rim:ServiceBindings used within a rim:Service
- 1038 • A generic ability to explore all Association involving a ServiceBinding or any RegistryObject

1039 The queries defined in this chapter are therefor focused on discovery rather than drill-down. Because
1040 ebXML Registry supports a flexible and extensible query capability, this chapter defines a single
1041 discovery query for each discoverable type within the WSDL information model. Each query may be as
1042 complex as necessary. However, the complexity is hidden from the client using parameterized queries
1043 stored in the Registry as instances of the AdhocQuery type, in the same manner as any other
1044 RegistryObject.

1045 In the subsequent section each query is described simply in terms of its supported parameters that serve
1046 as its search criteria. The actual AdhocQuery instances are much more complex in comparison but they
1047 are not exposed to the client making the query. Details on these queries are specified canonically in
1048 chapter 11.

1049 **8.1.1 Discovery Query Patterns**

1050 The discovery queries are specified from the bottom of the layered WSDL information model to the top
1051 (portType, binding, port and service). The query for each layer specifies parameters specific to it as well
1052 as parameters specific to each of the lower layers that it builds upon. Thus the number of parameters
1053 increase as queries are defined for higher level types in the model. This is key to being able to discover
1054 higher level objects based on attributes of the lower level objects that they build upon.

1055 There are many parameters supported for the discovery query for each higher level type in the model.
1056 However, it is often the case that discovery may not require parameters specific to all lower level types.
1057 To facilitate pruning of the discovery query for unwanted predicates related to lower level types there is a
1058 special parameter name \$considerXXX where XXX represents a lower level type within the model. If the
1059 value of this parameter is set to "0" then all parameter values specific to lower level type MUST be
1060 ignored by the discovery query.

1061 **8.2 WSDL Document Discovery Query**

1062 The WSDL Document discovery query MUST be implemented by an ebXML Registry implementing this
1063 profile. It allows the discovery of WSDL documents using zero or more of the parameters described next.

1064 **8.2.1 Parameter \$name**

1065 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1066 value of RegistryObjects that have objectType of WSDL.

1067 **8.2.2 Parameter \$description**

1068 This parameter's value MAY specify a string containing a pattern to match against the description
1069 attribute value of RegistryObjects that have objectType of WSDL.

1070 **8.2.3 Parameter \$targetNamespace**

1071 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1072 of a WSDL document.

1073 **8.2.4 Parameter \$importedNamespaces**

1074 This parameter's value MAY specify a string containing a pattern to match against the namespaces
1075 imported by a WSDL document.

1076 8.2.5 Example of WSDL Document Discovery Query

1077 The following example illustrates how to find all WSDL documents that have a targetNamespace
1078 containing the string "oasis" and that import a namespace with string "org.w3". Note that additional
1079 supported parameters MAY also be specified if needed.

1080

```
1081 <AdhocQueryRequest>  
1082   <rim:AdhocQuery id="urn:oasis:names:tc:ebxml-  
1083   regrep:profile:ws:query:WSDLDiscoveryQuery">  
1084     <rim:Slot name="$targetNamespace">  
1085       <rim:ValueList>  
1086         <rim:Value>%oasis%</rim:Value>  
1087       </rim:ValueList>  
1088     </rim:Slot>  
1089     <rim:Slot name="$importedNamespaces">  
1090       <rim:ValueList>  
1091         <rim:Value>%org.w3%</rim:Value>  
1092       </rim:ValueList>  
1093     </rim:Slot>  
1094   </rim:AdhocQuery>  
1095 </AdhocQueryRequest>
```

1096

Example of WSDL Document Discovery Query

1097 8.3 PortType Discovery Query

1098 The WSDL PortType discovery query allows the discovery of wsdl:portType instances using zero or more
1099 of the parameters described next.

1100 8.3.1 Parameter \$portType.name

1101 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1102 value of wsdl:portType instances.

1103 8.3.2 Parameter \$portType.description

1104 This parameter's value MAY specify a string containing a pattern to match against the content of the
1105 wsdl:documentation element within wsdl:portType instances.

1106 8.3.3 Parameter \$portType.targetNamespace

1107 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1108 of wsdl:portType instances.

1109 8.3.4 Parameter \$portType.schemaNamespaces

1110 This parameter's value MAY specify a string containing a pattern to match against the XML schema
1111 namespaces used within the wsdl:message instances used within the wsdl:operation instances used
1112 within the wsdl:portType instances.

1113 8.3.5 Example of WSDL PortType Discovery Query

1114 The following example illustrates how to find all wsdl:portType instances that have a name containing the
1115 string "QueryManager" and have import an XML Schema with namespace containing the string "ebxml-
1116 regrep". Note that additional supported parameters MAY also be specified if needed.

1117

```
1118 <AdhocQueryRequest>  
1119   <rim:AdhocQuery id="urn:oasis:names:tc:ebxml-  
1120   regrep:profile:ws:query:WSDLPortTypeDiscoveryQuery">
```

```
1121 <rim:Slot name="$portType.name">
1122 <rim:ValueList>
1123 <rim:Value>%QueryManager%</rim:Value>
1124 </rim:ValueList>
1125 </rim:Slot>
1126 <rim:Slot name="$portType.schemaNamespaces">
1127 <rim:ValueList>
1128 <rim:Value>%ebxml-regrep%</rim:Value>
1129 </rim:ValueList>
1130 </rim:Slot>
1131 </rim:AdhocQuery>
1132 </AdhocQueryRequest>
```

1133 **Example of WSDL PortType Discovery Query**

1134 **8.4 WSDL Binding Discovery Query**

1135 The WSDL Binding discovery query allows the discovery of wsdl:binding instances using zero or more of
1136 the parameters described next.

1137 **8.4.1 Parameter \$binding.name**

1138 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1139 value of wsdl:binding instances.

1140 **8.4.2 Parameter \$binding.description**

1141 This parameter's value MAY specify a string containing a pattern to match against the content of the
1142 wsdl:documentation element within wsdl:binding instances.

1143 **8.4.3 Parameter \$binding.targetNamespace**

1144 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1145 of wsdl:binding instances.

1146 **8.4.4 Parameter \$binding.protocolType**

1147 This parameter's value MAY specify a string containing a pattern to match against the id attribute value
1148 of the ClassificationNode representing the protocolType supported by wsdl:binding instances.

1149 **8.4.5 Parameter \$binding.transportType**

1150 This parameter's value MAY specify a string containing a pattern to match against the id attribute value
1151 of the ClassificationNode representing the transportType supported by wsdl:binding instances.

1152 **8.4.6 Parameter \$binding.soapStyle**

1153 This parameter's value MAY specify a string containing a pattern to match against the id attribute value
1154 of the ClassificationNode representing the soap style of wsdl:binding instances.

1155 **8.4.7 Parameter \$considerPortType**

1156 This parameter's value MAY specify a string a "1" or "0" to indicate whether or not to consider the
1157 portType specific parameters that follow when processing the query. If unspecified the value defaults to
1158 "0".

1159 **8.4.8 Parameter \$portType.name**

1160 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1161 value of wsdl:portType instances that are used by the objects being discovered.

1162 **8.4.9 Parameter \$portType.description**

1163 This parameter's value MAY specify a string containing a pattern to match against the content of the
1164 wsdl:documentation element within wsdl:portType instances that are used by the objects being
1165 discovered.

1166 **8.4.10 Parameter \$portType.targetNamespace**

1167 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1168 of wsdl:portType instances that are used by the objects being discovered.

1169 **8.4.11 Parameter \$portType.schemaNamespace**

1170 This parameter's value MAY specify a string containing a pattern to match against the XML schema
1171 namespaces used within the wsdl:message instances used within the wsdl:operation instances used
1172 within the wsdl:portType instances that are used by the objects being discovered.

1173 **8.4.12 Example of WSDL Binding Discovery Query**

1174 The following example illustrates how to find all wsdl:binding instances that have a name containing the
1175 string "QueryManager" and have a binding that supports SOAP protocol using HTTP transport. Note that
1176 additional supported parameters MAY also be specified if needed.

1177

```
1178 <AdhocQueryRequest>  
1179   <rim:AdhocQuery id="urn:oasis:names:tc:ebxml-  
1180   regrep:profile:ws:query:WSDLBindingDiscoveryQuery">  
1181     <rim:Slot name="$binding.name">  
1182       <rim:ValueList>  
1183         <rim:Value>%QueryManager%</rim:Value>  
1184       </rim:ValueList>  
1185     </rim:Slot>  
1186     <rim:Slot name="$binding.protocolType">  
1187       <rim:ValueList>  
1188         <rim:Value>urn:oasis:names:tc:ebxml-  
1189   regrep:profile:ws:ProtocolType:SOAP</rim:Value>  
1190       </rim:ValueList>  
1191     </rim:Slot>  
1192   </rim:AdhocQuery>  
1193 </AdhocQueryRequest>
```

1194

Example of WSDL Port Discovery Query

1195 **8.5 WSDL Port Discovery Query**

1196 The WSDL Port discovery query allows the discovery of wsdl:port instances using zero or more of the
1197 parameters described next.

1198 **8.5.1 Parameter \$port.name**

1199 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1200 value of wsdl:port instances.

1201 **8.5.2 Parameter \$port.description**

1202 This parameter's value MAY specify a string containing a pattern to match against the content of the
1203 wsdl:documentation element within wsdl:port instances.

1204 **8.5.3 Parameter \$port.targetNamespace**

1205 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1206 of wsdl:port instances.

1207 **8.5.4 Parameter \$port.accessURI**

1208 This parameter's value MAY specify a string containing a pattern to match against the accessURI for the
1209 endpoint defined for the wsdl:port instances.

1210 **8.5.5 Parameter \$considerBinding**

1211 This parameter's value MAY specify a string a "1" or "0" to indicate whether or not to consider the binding
1212 specific parameters that follow when processing the query. If unspecified the value defaults to "0".

1213 **8.5.6 Parameter \$binding.name**

1214 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1215 value of wsdl:binding instances that are used by the objects being discovered.

1216 **8.5.7 Parameter \$binding.description**

1217 This parameter's value MAY specify a string containing a pattern to match against the content of the
1218 wsdl:documentation element within wsdl:binding instances that are used by the objects being discovered.

1219 **8.5.8 Parameter \$binding.targetNamespace**

1220 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1221 of wsdl:binding instances that are used by the objects being discovered.

1222 **8.5.9 Parameter \$binding.protocolType**

1223 This parameter's value MAY specify a string containing a pattern to match against the id attribute value
1224 of the ClassificationNode representing the protocolType supported by wsdl:binding instances that are
1225 used by the objects being discovered.

1226 **8.5.10 Parameter \$binding.transportType**

1227 This parameter's value MAY specify a string containing a pattern to match against the id attribute value
1228 of the ClassificationNode representing the transportType supported by wsdl:binding instances that are
1229 used by the objects being discovered.

1230 **8.5.11 Parameter \$binding.soapStyle**

1231 This parameter's value MAY specify a string containing a pattern to match against the id attribute value
1232 of the ClassificationNode representing the soap style of wsdl:binding instances that are used by the
1233 objects being discovered.

1234 **8.5.12 Parameter \$considerPortType**

1235 This parameter's value MAY specify a string a "1" or "0" to indicate whether or not to consider the
1236 portType specific parameters that follow when processing the query. If unspecified the value defaults to
1237 "0".

1238 **8.5.13 Parameter \$portType.name**

1239 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1240 value of wsdl:portType instances that are used by the objects being discovered.

1241 **8.5.14 Parameter \$portType.description**

1242 This parameter's value MAY specify a string containing a pattern to match against the content of the
1243 wsdl:documentation element within wsdl:portType instances that are used by the objects being
1244 discovered.

1245 **8.5.15 Parameter \$portType.targetNamespace**

1246 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1247 of wsdl:portType instances that are used by the objects being discovered.

1248 **8.5.16 Parameter \$portType.schemaNamespace**

1249 This parameter's value MAY specify a string containing a pattern to match against the XML schema
1250 namespaces used within the wsdl:message instances used within the wsdl:operation instances used
1251 within the wsdl:portType instances that are used by the objects being discovered.

1252 **8.5.17 Example of WSDL Port Discovery Query**

1253 The following example illustrates how to find all wsdl:port instances that have a name containing the
1254 string "QueryManager" and have a binding that supports SOAP protocol using HTTP transport. Note that
1255 additional supported parameters MAY also be specified if needed.

1256

```
1257 <AdhocQueryRequest>  
1258   <rim:AdhocQuery id="urn:oasis:names:tc:ebxml-  
1259   regrep:profile:ws:query:WSDLPortDiscoveryQuery">  
1260     <rim:Slot name="$port.name">  
1261       <rim:ValueList>  
1262         <rim:Value>%QueryManager%</rim:Value>  
1263       </rim:ValueList>  
1264     </rim:Slot>  
1265     <rim:Slot name="$port.accessURI">  
1266       <rim:ValueList>  
1267         <rim:Value>http://acme%</rim:Value>  
1268       </rim:ValueList>  
1269     </rim:Slot>  
1270   </rim:AdhocQuery>  
1271 </AdhocQueryRequest>
```

1272

Example of WSDL Port Discovery Query

1273 **8.6 WSDL Service Discovery Query**

1274 The WSDL Service discovery query allows the discovery of wsdl:service instances using zero or more of
1275 the parameters described next.

1276 **8.6.1 Parameter \$service.name**

1277 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1278 value of wsdl:service instances.

1279 **8.6.2 Parameter \$service.description**

1280 This parameter's value MAY specify a string containing a pattern to match against the content of the
1281 wsdl:documentation element within wsdl:service instances.

1282 **8.6.3 Parameter \$service.targetNamespace**

1283 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1284 of wsdl:service instances.

1285 **8.6.4 Parameter \$considerPort**

1286 This parameter's value MAY specify a string a "1" or "0" to indicate whether or not to consider the port
1287 specific parameters that follow when processing the query. If unspecified the value defaults to "0".

1288 **8.6.5 Parameter \$port.name**

1289 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1290 value of wsdl:port instances that are used by the objects being discovered.

1291 **8.6.6 Parameter \$port.description**

1292 This parameter's value MAY specify a string containing a pattern to match against the content of the
1293 wsdl:documentation element within wsdl:port instances that are used by the objects being discovered.

1294 **8.6.7 Parameter \$port.targetNamespace**

1295 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1296 of wsdl:port instances that are used by the objects being discovered.

1297 **8.6.8 Parameter \$port.accessURI**

1298 This parameter's value MAY specify a string containing a pattern to match against the accessURI for the
1299 endpoint defined for the wsdl:port instances that are used by the objects being discovered.

1300 **8.6.9 Parameter \$considerBinding**

1301 This parameter's value MAY specify a string a "1" or "0" to indicate whether or not to consider the
1302 binding specific parameters that follow when processing the query. If unspecified the value defaults to
1303 "0".

1304 **8.6.10 Parameter \$binding.name**

1305 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1306 value of wsdl:binding instances that are used by the objects being discovered.

1307 **8.6.11 Parameter \$binding.description**

1308 This parameter's value MAY specify a string containing a pattern to match against the content of the
1309 wsdl:documentation element within wsdl:binding instances that are used by the objects being discovered.

1310 **8.6.12 Parameter \$binding.targetNamespace**

1311 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1312 of wsdl:binding instances that are used by the objects being discovered.

1313 **8.6.13 Parameter \$binding.protocolType**

1314 This parameter's value MAY specify a string containing a pattern to match against the id attribute value
1315 of the ClassificationNode representing the protocolType supported by wsdl:binding instances that are
1316 used by the objects being discovered.

1317 **8.6.14 Parameter \$binding.transportType**

1318 This parameter's value MAY specify a string containing a pattern to match against the id attribute value
1319 of the ClassificationNode representing the transportType supported by wsdl:binding instances that are
1320 used by the objects being discovered.

1321 **8.6.15 Parameter \$binding.soapStyle**

1322 This parameter's value MAY specify a string containing a pattern to match against the id attribute value
1323 of the ClassificationNode representing the soap style of wsdl:binding instances that are used by the
1324 objects being discovered.

1325 **8.6.16 Parameter \$considerPortType**

1326 This parameter's value MAY specify a string a "1" or "0" to indicate whether or not to consider the
1327 portType specific parameters that follow when processing the query. If unspecified the value defaults to
1328 "0".

1329 **8.6.17 Parameter \$portType.name**

1330 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1331 value of wsdl:portType instances that are used by the objects being discovered.

1332 **8.6.18 Parameter \$portType.description**

1333 This parameter's value MAY specify a string containing a pattern to match against the content of the
1334 wsdl:documentation element within wsdl:portType instances that are used by the objects being
1335 discovered.

1336 **8.6.19 Parameter \$portType.targetNamespace**

1337 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1338 of wsdl:portType instances that are used by the objects being discovered.

1339 **8.6.20 Parameter \$portType.schemaNamespace**

1340 This parameter's value MAY specify a string containing a pattern to match against the XML schema
1341 namespaces used within the wsdl:message instances used within the wsdl:operation instances used
1342 within the wsdl:portType instances that are used by the objects being discovered.

1343 **8.6.21 Example of WSDL Service Discovery Query**

1344 The following example illustrates how to find all wsdl:service instances that have a name containing the
1345 string "QueryManager" and have a targetNamespace containing the string "ebXML". Note that additional
1346 supported parameters MAY also be specified if needed.

1347

```
1348 <AdhocQueryRequest>  
1349   <rim:AdhocQuery id="urn:oasis:names:tc:ebxml-  
1350   regrep:profile:ws:query:WSDLServiceDiscoveryQuery">  
1351     <rim:Slot name="$name">  
1352       <rim:ValueList>  
1353         <rim:Value>%QueryManager%</rim:Value>  
1354       </rim:ValueList>  
1355     </rim:Slot>  
1356     <rim:Slot name="$targetNamespace">  
1357       <rim:ValueList>  
1358         <rim:Value>%ebXML%</rim:Value>  
1359       </rim:ValueList>  
1360     </rim:Slot>
```

1361
1362
1363

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

Example of WSDL Service Discovery Query

9 Event Notification Profile

1364

1365 The ebXML Registry provides the ability for a user or an automated service to create a subscription to
1366 events that match a specified criteria. Whenever an event matching the specified criteria occurs, the
1367 registry notifies the subscriber that the event transpired. The event matching criteria is specified using a
1368 stored parameterized query similar to the discovery queries described in previous chapter.

1369 This chapter specifies the template Subscriptions that MUST be implemented by an ebXML Registry
1370 implementing the Web Services profile. To subscribe to instances of types defined within the WSDL
1371 information model a subscription can simply reuse the discovery queries defined in the previous chapter
1372 or define more specific queries.

9.1 Subscribing to a WSDL Document

1373

1374 A client may publish a rim:Subscription instance using the WSDL Document discovery query as the
1375 selector in order to receive notification of changes to WSDL documents that they have an interest in.

1376

```
1377 <rim:Subscription id=${WSDL_SUBSCRIPTION_ID}  
1378   selector="urn:oasis:names:tc:ebxml-  
1379   regrep:profile:ws:query:WSDLDiscoveryQuery">  
1380   <rim:Slot name="$name">  
1381     <rim:ValueList>  
1382       <rim:Value>%ebXML%</rim:Value>  
1383     </rim:ValueList>  
1384   </rim:Slot>  
1385   <rim:Slot name="$targetNamespace">  
1386     <rim:ValueList>  
1387       <rim:Value>%oasis%</rim:Value>  
1388     </rim:ValueList>  
1389   </rim:Slot>  
1390  
1391   <!-- email address endPoint for receiving notification via email -->  
1392   <rim:NotifyAction notificationOption="urn:oasis:names:tc:ebxml-  
1393   regrep:NotificationOptionType:ObjectRefs"  
1394     endPoint="mailto:farrukh.najmi@sun.com"/>  
1395  
1396   <!--Web Service endPoint for receiving notification via SOAP -->  
1397   <rim:NotifyAction notificationOption="urn:oasis:names:tc:ebxml-  
1398   regrep:NotificationOptionType:Objects"  
1399     endPoint="urn:acme:wSDLChangeListenerService"/>  
1400 </rim:Subscription>
```

1401

Listing 11: Example of Subscription to WSDL Documents

1402

1403 The above example show how to create a subscription for WSDL Documents where name contains the
1404 string “ebXML” and targetNamespace contains the string “oasis”. Light-weight notifications containing
1405 references to WSDL documents are configured to be sent to an email address while heavy-weight
1406 notifications containing actual WSDL documents are configured to be sent to a listener service using the
1407 SOAP protocol.

9.2 Subscribing to PortType changes

1408

1409 A client may publish a rim:Subscription instance using the WSDL PortType discovery query as the
1410 selector in order to receive notification of changes to WSDL PortTypes that they have an interest in.

1411

```
1412 <rim:Subscription id=${WSDL_SUBSCRIPTION_ID}  
1413   selector="urn:oasis:names:tc:ebxml-  
1414   regrep:profile:ws:query:PortTypeDiscoveryQuery">
```

1415
1416
1417

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

1418 **Listing 12: Example of Subscription to WSDL PortTypes**

1419 **9.3 Subscribing to Binding changes**

1420 A client may publish a rim:Subscription instance using the WSDL Binding discovery query as the selector
1421 in order to receive notification of changes to WSDL PortBindings that they have an interest in.

1422

```
1423 <rim:Subscription id=${WSDL_SUBSCRIPTION_ID}  
1424 selector="urn:oasis:names:tc:ebxml-  
1425 regrep:profile:ws:query:BindingDiscoveryQuery">  
1426  
1427 ...  
1428 </rim:Subscription>
```

1429 **Listing 13: Example of Subscription to WSDL Bindings**

1430 **9.4 Subscribing to Port changes**

1431 A client may publish a rim:Subscription instance using the WSDL Port discovery query as the selector in
1432 order to receive notification of changes to WSDL Ports that they have an interest in.

1433

```
1434 <rim:Subscription id=${WSDL_SUBSCRIPTION_ID}  
1435 selector="urn:oasis:names:tc:ebxml-  
1436 regrep:profile:ws:query:PortDiscoveryQuery">  
1437  
1438 ...  
1439 </rim:Subscription>
```

1440 **Listing 14: Example of Subscription to WSDL Ports**

1441 **9.5 Subscribing to Service changes**

1442 A client may publish a rim:Subscription instance using the WSDL Service discovery query as the
1443 selector in order to receive notification of changes to WSDL Services that they have an interest in.

1444

```
1445 <rim:Subscription id=${WSDL_SUBSCRIPTION_ID}  
1446 selector="urn:oasis:names:tc:ebxml-  
1447 regrep:profile:ws:query:ServiceDiscoveryQuery">  
1448  
1449 ...  
1450 </rim:Subscription>
```

1451 **Listing 15: Example of Subscription to WSDL Ports**

1452

1453 **10 Security Profile**

1454 This chapter specifies security aspects governing the publish, management and discovery of web
1455 services within ebXML Registry.

1456 **10.1 SubjectRole Profile**

1457 The ebXML Registry defines a set of pre-defined roles in the SubjectRole scheme. A domain specific
1458 mapping to ebRIM MAY define additional domain specific roles by extending the SubjectRole scheme.
1459 TBD??

1460 **10.2 SubjectGroup Profile**

1461 The ebXML Registry defines a set of pre-defined roles in the *SubjectGroup* scheme. A domain specific
1462 mapping to ebRIM MAY define additional domain specific groups by extending the SubjectGroup
1463 scheme.
1464 TBD??

1465 **10.3 AccessControlPolicy Profile**

1466 The ebXML Registry provides a powerful and extensible access control feature that makes sure that a
1467 user may only perform those actions on a RegistryObject or repository item for which they are
1468 authorized.

1469 ...

1470 TBD??

1471

1472 Need to specify external link mapping in mapping chapter??

11 Canonical Metadata Definitions

1473

1474 This chapter specifies the canonical metadata defined by this profile.

11.1 ObjectType Extensions

1475

1476 The following new extensions to the canonical ObjectType ClassificationScheme are described by this
1477 profile:

1478

```
1479     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
1480 regrep:ObjectType:RegistryObject:ExtrinsicObject"  
1481 lid="urn:oasis:names:tc:ebxml-  
1482 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL" code="WSDL"  
1483 id="urn:oasis:names:tc:ebxml-  
1484 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL">  
1485     <rim:Name>  
1486         <rim:LocalizedString charset="UTF-8" value="label.WSDL"/>  
1487     </rim:Name>  
1488     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
1489 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL"  
1490 lid="urn:oasis:names:tc:ebxml-  
1491 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType"  
1492 code="PortType" id="urn:oasis:names:tc:ebxml-  
1493 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType">  
1494     <rim:Name>  
1495         <rim:LocalizedString charset="UTF-8" value="label.PortType"/>  
1496     </rim:Name>  
1497 </rim:ClassificationNode>  
1498     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
1499 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL"  
1500 lid="urn:oasis:names:tc:ebxml-  
1501 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding"  
1502 code="Binding" id="urn:oasis:names:tc:ebxml-  
1503 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding">  
1504     <rim:Name>  
1505         <rim:LocalizedString charset="UTF-8" value="label.Binding"/>  
1506     </rim:Name>  
1507 </rim:ClassificationNode>  
1508     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
1509 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL"  
1510 lid="urn:oasis:names:tc:ebxml-  
1511 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Port" code="Port"  
1512 id="urn:oasis:names:tc:ebxml-  
1513 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Port">  
1514     <rim:Name>  
1515         <rim:LocalizedString charset="UTF-8" value="label.Port"/>  
1516     </rim:Name>  
1517 </rim:ClassificationNode>  
1518     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-  
1519 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL"  
1520 lid="urn:oasis:names:tc:ebxml-  
1521 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Service"  
1522 code="Service" id="urn:oasis:names:tc:ebxml-  
1523 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Service">  
1524     <rim:Name>  
1525         <rim:LocalizedString charset="UTF-8" value="label.Service"/>  
1526     </rim:Name>  
1527 </rim:ClassificationNode>  
1528 </rim:ClassificationNode>
```

1529

Listing 16: Example of Subscription to WSDL Ports

1530

1531 11.2 Canonical ClassificationSchemes

1532 The following new canonical ClassificationSchemes are described by this profile:

```
1533     <rim:ClassificationScheme lid="urn:oasis:names:tc:ebxml-
1534     regrep:profile:ws:classificationScheme:ProtocolType"
1535     id="urn:oasis:names:tc:ebxml-
1536     regrep:profile:ws:classificationScheme:ProtocolType" isInternal="true"
1537     nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode">
1538         <rim:Name>
1539             <rim:LocalizedString charset="UTF-8"
1540             value="label.ProtocolType"/>
1541         </rim:Name>
1542         <rim:Description>
1543             <rim:LocalizedString charset="UTF-8"
1544             value="label.ProtocolType.desc"/>
1545         </rim:Description>
1546         <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1547         regrep:profile:ws:ProtocolType:SOAP" code="SOAP"
1548         id="urn:oasis:names:tc:ebxml-regrep:profile:ws:ProtocolType:SOAP"/>
1549         <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1550         regrep:profile:ws:ProtocolType:AS2" code="AS2"
1551         id="urn:oasis:names:tc:ebxml-regrep:profile:ws:ProtocolType:AS2"/>
1552         <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1553         regrep:profile:ws:ProtocolType:Atom" code="Atom"
1554         id="urn:oasis:names:tc:ebxml-regrep:profile:ws:ProtocolType:Atom"/>
1555     </rim:ClassificationScheme>
1556
1557     <rim:ClassificationScheme lid="urn:oasis:names:tc:ebxml-
1558     regrep:profile:ws:classificationScheme:TransportType"
1559     id="urn:oasis:names:tc:ebxml-
1560     regrep:profile:ws:classificationScheme:TransportType" isInternal="true"
1561     nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode">
1562         <rim:Name>
1563             <rim:LocalizedString charset="UTF-8"
1564             value="label.TransportType"/>
1565         </rim:Name>
1566         <rim:Description>
1567             <rim:LocalizedString charset="UTF-8"
1568             value="label.TransportType.desc"/>
1569         </rim:Description>
1570         <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1571         regrep:profile:ws:TransportType:HTTP" code="HTTP"
1572         id="urn:oasis:names:tc:ebxml-regrep:profile:ws:TransportType:HTTP"/>
1573         <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1574         regrep:profile:ws:TransportType:MOM" code="MOM"
1575         id="urn:oasis:names:tc:ebxml-regrep:profile:ws:TransportType:MOM"/>
1576         <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1577         regrep:profile:ws:TransportType:BEEP" code="BEEP"
1578         id="urn:oasis:names:tc:ebxml-regrep:profile:ws:TransportType:BEEP"/>
1579     </rim:ClassificationScheme>
1580
1581     <rim:ClassificationScheme lid="urn:oasis:names:tc:ebxml-
1582     regrep:profile:ws:classificationScheme:SOAPStyleType"
1583     id="urn:oasis:names:tc:ebxml-
1584     regrep:profile:ws:classificationScheme:SOAPStyleType" isInternal="true"
1585     nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode">
1586         <rim:Name>
1587             <rim:LocalizedString charset="UTF-8"
1588             value="label.SOAPStyleType"/>
1589         </rim:Name>
```

```

1590     <rim:Description>
1591         <rim:LocalizedString charset="UTF-8"
1592 value="label.SOAPType.desc"/>
1593     </rim:Description>
1594     <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1595 regrep:profile:wsPStyleType:RPC" code="RPC"
1596 id="urn:oasis:names:tc:ebxml-regrep:profile:wsPStyleType:RPC"/>
1597     <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1598 regrep:profile:wsPStyleType:Document" code="Document"
1599 id="urn:oasis:names:tc:ebxml-regrep:profile:wsPStyleType:Document"/>
1600 </rim:ClassificationScheme>

```

Listing 17: Example of Subscription to WSDL Ports

11.3 Canonical Queries

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

11.3.1 WSDL Document Discovery Query

```

1606 <!--
1607 Parameterized Adhoc Query for WSDL Discovery query.
1608 Finds by Name, Description, ComponentType, ResourceType
1609 -->
1610 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
1611 regrep:profile:ws:query:WSDLDiscoveryQuery"
1612 id="urn:oasis:names:tc:ebxml-
1613 regrep:profile:ws:query:WSDLDiscoveryQuery">
1614     <rim:Name>
1615         <rim:LocalizedString value="label.WSDLDiscoveryQuery"/>
1616     </rim:Name>
1617     <rim:Description>
1618         <rim:LocalizedString value="label.WSDLDiscoveryQuery.desc"/>
1619     </rim:Description>
1620     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1621 regrep:QueryLanguage:SQL-92">
1622 SELECT DISTINCT ro.* FROM RegistryObject ro, Name_ nm, Description d,
1623 Slot tns, Slot ins
1624 WHERE (l=1)
1625 AND (ro.objectType = 'urn:oasis:names:tc:ebxml-
1626 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL')
1627 AND (nm.parent = ro.id AND UPPER ( nm.value ) LIKE UPPER ( '$name'
1628 ) )
1629 AND (d.parent = ro.id AND UPPER ( d.value ) LIKE UPPER
1630 ( '$description' ) )
1631 AND (ro.id = tns.parent
1632 AND tns.name_ = 'urn:oasis:names:tc:ebxml-
1633 regrep:profile:ws:wSDL:targetNamespace')
1634 AND tns.value LIKE '$targetNamespace')
1635 AND (ro.id = ins.parent
1636 AND ins.name_ = 'urn:oasis:names:tc:ebxml-
1637 regrep:profile:ws:wSDL:importedNamespaces')
1638 AND ins.value LIKE '$importedNamespaces')
1639
1640     </rim:QueryExpression>
1641 </rim:AdhocQuery>

```

Listing 18: WSDL Document Discovery Query

1643 11.3.2 WSDL PortType Discovery Query

```
1644 <!--
1645 Parameterized Adhoc Query for WSDL PortType Discovery query.
1646 -->
1647 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
1648 regrep:profile:ws:query:PortTypeDiscoveryQuery"
1649 id="urn:oasis:names:tc:ebxml-
1650 regrep:profile:ws:query:PortTypeDiscoveryQuery">
1651 <rim:Name>
1652 <rim:LocalizedString value="label.PortTypeDiscoveryQuery"/>
1653 </rim:Name>
1654 <rim:Description>
1655 <rim:LocalizedString value="label.PortTypeDiscoveryQuery.desc"/>
1656 </rim:Description>
1657 <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1658 regrep:QueryLanguage:SQL-92">
1659 SELECT DISTINCT portType.* FROM ExtrinsicObject portType, Name_
1660 portTypeName, Description portTypeDesc, Slot portTypeTNS
1661 WHERE
1662 portType.objectType = 'urn:oasis:names:tc:ebxml-
1663 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType'
1664 AND (portTypeName.parent = portType.id AND UPPER ( portTypeName.value )
1665 LIKE UPPER ( '$portType.name' ) )
1666 AND (portTypeDesc.parent = portType.id AND UPPER ( portTypeDesc.value )
1667 LIKE UPPER ( '$portType.description' ) )
1668 AND (portType.id = s.parent
1669 AND portTypeTNS.name_ = 'urn:oasis:names:tc:ebxml-
1670 regrep:profile:ws:wSDL:targetNamespace'
1671 AND portTypeTNS.value LIKE '$portType.targetNamespace')
1672 </rim:QueryExpression>
1673 </rim:AdhocQuery>
```

1674 **Listing 19: WSDL PortType Discovery Query**

1675 11.3.3 WSDL Binding Discovery Query

```
1676 <!--
1677 Parameterized Adhoc Query for WSDL Binding Discovery query.
1678 -->
1679 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
1680 regrep:profile:ws:query:BindingDiscoveryQuery"
1681 id="urn:oasis:names:tc:ebxml-
1682 regrep:profile:ws:query:BindingDiscoveryQuery">
1683 <rim:Name>
1684 <rim:LocalizedString value="label.BindingDiscoveryQuery"/>
1685 </rim:Name>
1686 <rim:Description>
1687 <rim:LocalizedString value="label.BindingDiscoveryQuery.desc"/>
1688 </rim:Description>
1689 <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1690 regrep:QueryLanguage:SQL-92">
1691 SELECT DISTINCT binding.* FROM
1692 ExtrinsicObject binding, Name_ bindingName, Description bindingDesc,
1693 Slot bindingTNS,
1694 Association implements
1695 WHERE
1696 binding.objectType = 'urn:oasis:names:tc:ebxml-
1697 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding'
1698 AND (bindingName.parent = binding.id AND UPPER ( bindingName.value )
1699 LIKE UPPER ( '$binding.name' ) )
1700 AND (bindingDesc.parent = binding.id AND UPPER ( bindingDesc.value )
1701 LIKE UPPER ( '$binding.description' ) )
```

```

1702 AND (binding.id = s.parent
1703 AND bindingTNS.name_ = 'urn:oasis:names:tc:ebxml-
1704 regrep:profile:ws:wSDL:targetNamespace')
1705 AND bindingTNS.value LIKE '$binding.targetNamespace')
1706 AND (binding.id IN ( SELECT classifiedObject FROM Classification WHERE
1707 classificationNode IN ( SELECT id
1708 FROM ClassificationNode WHERE path LIKE '$binding.protocolType' ) ))
1709 AND (binding.id IN ( SELECT classifiedObject FROM Classification WHERE
1710 classificationNode IN ( SELECT id
1711 FROM ClassificationNode WHERE path LIKE
1712 '$binding.transportType%' ) ))
1713 AND (binding.id IN ( SELECT classifiedObject FROM Classification WHERE
1714 classificationNode IN ( SELECT id
1715 FROM ClassificationNode WHERE path LIKE
1716 '$binding.soapStyleType%' ) ))
1717
1718 AND ($considerPortType = 0 OR (
1719 implements.sourceObject=binding.id AND
1720 implements.associationType='urn:oasis:names:tc:ebxml-
1721 regrep:AssociationType:Implements' AND implements.targetObject IN
1722 (
1723 SELECT DISTINCT portType.id from ExtrinsicObject portType, Name_
1724 portTypeName, Description portTypeDesc, Slot portTypeTNS
1725 WHERE
1726 portType.objectType = 'urn:oasis:names:tc:ebxml-
1727 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType'
1728 AND (portTypeName.parent = portType.id AND UPPER
1729 ( portTypeName.value ) LIKE UPPER ( '$portType.name' ) )
1730 AND (portTypeDesc.parent = portType.id AND UPPER
1731 ( portTypeDesc.value ) LIKE UPPER ( '$portType.description' ) )
1732 AND (portType.id = s.parent
1733 AND portTypeTNS.name_ = 'urn:oasis:names:tc:ebxml-
1734 regrep:profile:ws:wSDL:targetNamespace')
1735 AND portTypeTNS.value LIKE '$portType.targetNamespace')
1736 ))
1737 )
1738 </rim:QueryExpression>
1739 </rim:AdhocQuery>

```

Listing 20: WSDL Binding Discovery Query

11.3.4 WSDL Port Discovery Query

```

1742 <!--
1743 Parameterized Adhoc Query for WSDL Port Discovery query.
1744 -->
1745 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
1746 regrep:profile:ws:query:PortDiscoveryQuery"
1747 id="urn:oasis:names:tc:ebxml-
1748 regrep:profile:ws:query:PortDiscoveryQuery">
1749 <rim:Name>
1750 <rim:LocalizedString value="label.PortDiscoveryQuery"/>
1751 </rim:Name>
1752 <rim:Description>
1753 <rim:LocalizedString value="label.PortDiscoveryQuery.desc"/>
1754 </rim:Description>
1755 <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1756 regrep:QueryLanguage:SQL-92">
1757 SELECT DISTINCT port.* FROM
1758 ServiceBinding port, Name_ portName, Description portDesc, Slot
1759 portTNS,
1760 Association implements
1761 WHERE

```

```

1762 (port.id IN ( SELECT classifiedObject FROM Classification WHERE
1763 classificationNode = 'urn:oasis:names:tc:ebxml-
1764 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Port' ) )
1765
1766 AND (portName.parent = port.id AND UPPER ( portName.value ) LIKE UPPER
1767 ( '$port.name' ) )
1768 AND (portDesc.parent = port.id AND UPPER ( portDesc.value ) LIKE UPPER
1769 ( '$port.description' ) )
1770 AND (port.id = s.parent
1771 AND portTNS.name_ = 'urn:oasis:names:tc:ebxml-
1772 regrep:profile:ws:wSDL:targetNamespace'
1773 AND portTNS.value LIKE '$port.targetNamespace')
1774 AND (port.accessURI LIKE '$port.accessURI')
1775
1776 AND ($considerBinding = 0 OR (
1777 implements.sourceObject=port.id AND
1778 implements.associationType='urn:oasis:names:tc:ebxml-
1779 regrep:AssociationType:Implements' AND implements.targetObject IN
1780 (
1781 SELECT DISTINCT binding.id FROM
1782 ExtrinsicObject binding, Name_ bindingName, Description
1783 bindingDesc, Slot bindingTNS,
1784 Association implements
1785 WHERE
1786 binding.objectType = 'urn:oasis:names:tc:ebxml-
1787 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding'
1788 AND (bindingName.parent = binding.id AND UPPER
1789 ( bindingName.value ) LIKE UPPER ( '$binding.name' ) )
1790 AND (bindingDesc.parent = binding.id AND UPPER
1791 ( bindingDesc.value ) LIKE UPPER ( '$binding.description' ) )
1792 AND (binding.id = s.parent
1793 AND bindingTNS.name_ = 'urn:oasis:names:tc:ebxml-
1794 regrep:profile:ws:wSDL:targetNamespace'
1795 AND bindingTNS.value LIKE '$binding.targetNamespace')
1796 AND (binding.id IN ( SELECT classifiedObject FROM Classification
1797 WHERE classificationNode IN ( SELECT id
1798 FROM ClassificationNode WHERE path LIKE '$binding.protocolType' )
1799 ) )
1800 AND (binding.id IN ( SELECT classifiedObject FROM Classification
1801 WHERE classificationNode IN ( SELECT id
1802 FROM ClassificationNode WHERE path LIKE '$binding.transportType%'
1803 ) ) )
1804 AND (binding.id IN ( SELECT classifiedObject FROM Classification
1805 WHERE classificationNode IN ( SELECT id
1806 FROM ClassificationNode WHERE path LIKE '$binding.soapStyleType%'
1807 ) ) )
1808
1809 AND ($considerPortType = 0 OR (
1810 implements.sourceObject=binding.id AND
1811 implements.associationType='urn:oasis:names:tc:ebxml-
1812 regrep:AssociationType:Implements' AND implements.targetObject IN
1813 (
1814 SELECT DISTINCT portType.id from ExtrinsicObject portType, Name_
1815 portTypeName, Description portTypeDesc, Slot portTypeTNS
1816 WHERE
1817 portType.objectType = 'urn:oasis:names:tc:ebxml-
1818 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType'
1819 AND (portTypeName.parent = portType.id AND UPPER
1820 ( portTypeName.value ) LIKE UPPER ( '$portType.name' ) )
1821 AND (portTypeDesc.parent = portType.id AND UPPER
1822 ( portTypeDesc.value ) LIKE UPPER ( '$portType.description' ) )
1823 AND (portType.id = s.parent

```

```

1824         AND portTypeTNS.name_ = 'urn:oasis:names:tc:ebxml-
1825 regrep:profile:ws:wSDL:targetNamespace'
1826         AND portTypeTNS.value LIKE '$portType.targetNamespace')
1827     )
1828 ))
1829 ))
1830 )
1831     </rim:QueryExpression>
1832 </rim:AdhocQuery>

```

Listing 21: WSDL Port Discovery Query

11.3.5 WSDL Service Discovery Query

```

1835 <!--
1836 Parameterized Adhoc Query for WSDL Service Discovery query.
1837 -->
1838 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
1839 regrep:profile:ws:query:ServiceDiscoveryQuery"
1840 id="urn:oasis:names:tc:ebxml-
1841 regrep:profile:ws:query:ServiceDiscoveryQuery">
1842     <rim:Name>
1843         <rim:LocalizedString value="label.ServiceDiscoveryQuery"/>
1844     </rim:Name>
1845     <rim:Description>
1846         <rim:LocalizedString value="label.ServiceDiscoveryQuery.desc"/>
1847     </rim:Description>
1848     <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1849 regrep:QueryLanguage:SQL-92">
1850 SELECT DISTINCT service.* FROM
1851     Service service, Name_ serviceName, Description serviceDesc, Slot
1852     serviceTNS,
1853     ServiceBinding port
1854 WHERE
1855     (service.id IN ( SELECT classifiedObject FROM Classification WHERE
1856     classificationNode = 'urn:oasis:names:tc:ebxml-
1857 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Port' ) )
1858
1859     AND (serviceName.parent = service.id AND UPPER ( serviceName.value )
1860     LIKE UPPER ( '$serviceName' ) )
1861     AND (serviceDesc.parent = service.id AND UPPER ( serviceDesc.value )
1862     LIKE UPPER ( '$service.description' ) )
1863     AND (service.id = s.parent
1864     AND serviceTNS.name_ = 'urn:oasis:names:tc:ebxml-
1865 regrep:profile:ws:wSDL:targetNamespace'
1866     AND serviceTNS.value LIKE '$service.targetNamespace')
1867
1868     AND ($considerPort = 0 OR (
1869     service.id = port.service AND port.id IN
1870     (
1871
1872         SELECT DISTINCT port.id FROM
1873         ServiceBinding port, Name_ portName, Description portDesc, Slot
1874     portTNS,
1875     Association implements
1876     WHERE
1877         (port.id IN ( SELECT classifiedObject FROM Classification WHERE
1878     classificationNode = 'urn:oasis:names:tc:ebxml-
1879 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Port' ) )
1880
1881         AND (portName.parent = port.id AND UPPER ( portName.value ) LIKE
1882     UPPER ( '$port.name' ) ) )

```

```

1883         AND (portDesc.parent = port.id AND UPPER ( portDesc.value ) LIKE
1884 UPPER ( '$port.description' ) )
1885         AND (port.id = s.parent
1886         AND portTNS.name_ = 'urn:oasis:names:tc:ebxml-
1887 regrep:profile:ws:wSDL:targetNamespace'
1888         AND portTNS.value LIKE '$port.targetNamespace')
1889         AND (port.accessURI LIKE '$port.accessURI')
1890
1891         AND ($considerBinding = 0 OR (
1892         implements.sourceObject=port.id AND
1893 implements.associationType='urn:oasis:names:tc:ebxml-
1894 regrep:AssociationType:Implements' AND implements.targetObject IN
1895 (
1896         SELECT DISTINCT binding.id FROM
1897         ExtrinsicObject binding, Name_ bindingName, Description
1898 bindingDesc, Slot bindingTNS,
1899         Association implements
1900         WHERE
1901         binding.objectType = 'urn:oasis:names:tc:ebxml-
1902 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding'
1903         AND (bindingName.parent = binding.id AND UPPER
1904 ( bindingName.value ) LIKE UPPER ( '$binding.name' ) )
1905         AND (bindingDesc.parent = binding.id AND UPPER
1906 ( bindingDesc.value ) LIKE UPPER ( '$binding.description' ) )
1907         AND (binding.id = s.parent
1908         AND bindingTNS.name_ = 'urn:oasis:names:tc:ebxml-
1909 regrep:profile:ws:wSDL:targetNamespace'
1910         AND bindingTNS.value LIKE '$binding.targetNamespace')
1911         AND (binding.id IN ( SELECT classifiedObject FROM
1912 Classification WHERE classificationNode IN ( SELECT id
1913 FROM ClassificationNode WHERE path LIKE
1914 '$binding.protocolType' ) ) )
1915         AND (binding.id IN ( SELECT classifiedObject FROM
1916 Classification WHERE classificationNode IN ( SELECT id
1917 FROM ClassificationNode WHERE path LIKE
1918 '$binding.transportType%' ) ) )
1919         AND (binding.id IN ( SELECT classifiedObject FROM
1920 Classification WHERE classificationNode IN ( SELECT id
1921 FROM ClassificationNode WHERE path LIKE
1922 '$binding.soapStyleType%' ) ) )
1923
1924         AND ($considerBinding = 0 OR (
1925         implements.sourceObject=binding.id AND
1926 implements.associationType='urn:oasis:names:tc:ebxml-
1927 regrep:AssociationType:Implements' AND implements.targetObject IN
1928 (
1929         SELECT DISTINCT portType.id from ExtrinsicObject portType,
1930 Name_ portTypeName, Description portTypeDesc, Slot portTypeTNS
1931         WHERE
1932         portType.objectType = 'urn:oasis:names:tc:ebxml-
1933 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType'
1934         AND (portTypeName.parent = portType.id AND UPPER
1935 ( portTypeName.value ) LIKE UPPER ( '$portType.name' ) )
1936         AND (portTypeDesc.parent = portType.id AND UPPER
1937 ( portTypeDesc.value ) LIKE UPPER ( '$portType.description' ) )
1938         AND (portType.id = s.parent
1939         AND portTypeTNS.name_ = 'urn:oasis:names:tc:ebxml-
1940 regrep:profile:ws:wSDL:targetNamespace'
1941         AND portTypeTNS.value LIKE '$portType.targetNamespace')
1942         )
1943         ))
1944     )
1945 ))

```

1946
1947
1948
1949
1950

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

Listing 22: WSDL Service Discovery Query

1951 **12 References**

1952 **12.1 Normative References**

1953 [ebRIM] ebXML Registry Information Model version 3.0

1954 <http://docs.oasis-open.org/regrep/regrep-rim/v3.0/regrep-rim-3.0-os.pdf>

1955

1956 [ebRS] ebXML Registry Services Specification version 3.0

1957 <http://docs.oasis-open.org/regrep/regrep-rs/v3.0/regrep-rs-3.0-os.pdf>

1958 [UML] Unified Modeling Language version 1.5

1959 <http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.pdf>

1960 [ebRR-DPT] Deployment Profile Template For ebXML Registry 3.0 OASIS Specifications V_0.1.2

1961 [ebMS-DPT] Deployment Profile Template For OASIS Specification ebXML Message Service 2.0

1962 [WSDL] WSDL Specification

1963 <http://www.w3.org/TR/wSDL>

1964 **12.2 Informative References**

1965 **Appendix Anformative**

1966 [WSDL-OVW] WSDL Essentials

1967 <http://www.developer.com/services/article.php/1602051>

1968 [IMPL] ebXML Registry 3.0 Implementations

1969 freebXML Registry: A royalty free, open source ebXML Registry Implementation

1970 <http://ebxmlrr.sourceforge.net>

1971