



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

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 |

| | | |
|-----|---|----|
| 71 | 4.3.3 Element Name..... | 21 |
| 72 | 4.3.4 Element Description..... | 22 |
| 73 | 4.3.5 Element Classification..... | 22 |
| 74 | 4.4 wsdl:portType → rim:ExtrinsicObject Mapping..... | 23 |
| 75 | 4.4.1 Attribute objectType..... | 23 |
| 76 | 4.4.2 Attribute id..... | 23 |
| 77 | 4.4.3 Element Name..... | 23 |
| 78 | 4.4.4 Element Description..... | 24 |
| 79 | 4.5 wsdl:port «wsdl:binding to rim:Association Mapping..... | 24 |
| 80 | 4.5.1 Attribute id..... | 24 |
| 81 | 4.5.2 Attribute sourceObject..... | 24 |
| 82 | 4.5.3 Attribute targetObject..... | 24 |
| 83 | 4.5.4 Attribute associationType..... | 24 |
| 84 | 4.6 wsdl:binding «wsdl:portType Association Mapping..... | 25 |
| 85 | 4.6.1 Attribute id..... | 25 |
| 86 | 4.6.2 Attribute sourceObject..... | 25 |
| 87 | 4.6.3 Attribute targetObject..... | 25 |
| 88 | 4.6.4 Attribute associationType..... | 25 |
| 89 | 5 Publishing Profile..... | 26 |
| 90 | 5.1 Structure of WSDL Documents..... | 26 |
| 91 | 5.1.1 XxInterfaces.wsdl..... | 26 |
| 92 | 5.1.2 XxBindings.wsdl..... | 26 |
| 93 | 5.1.3 XxServices.wsdl..... | 26 |
| 94 | 6 Validation Service Profile..... | 27 |
| 95 | 6.1 Invocation Control File..... | 27 |
| 96 | 6.2 Business Rules..... | 27 |
| 97 | 7 Cataloging Service Profile..... | 28 |
| 98 | 7.1 Invocation Control File..... | 28 |
| 99 | 7.2 Input Metadata..... | 28 |
| 100 | 7.3 Input Content..... | 28 |
| 101 | 7.4 Output Metadata..... | 29 |
| 102 | 7.4.1 Changes to Input Metadata..... | 29 |
| 103 | 7.4.2 wsdl:service → rim:Service..... | 29 |
| 104 | 7.4.3 wsdl:port → rim:ServiceBinding..... | 29 |
| 105 | 7.4.4 wsdl:binding → rim:ExtrinsicObject..... | 29 |
| 106 | 7.4.5 wsdl:portType → rim:ExtrinsicObject..... | 29 |
| 107 | 7.4.6 wsdl:port « wsdl:binding Association..... | 29 |
| 108 | 7.4.7 wsdl:binding « wsdl:portType Association..... | 29 |
| 109 | 8 Discovery Profile..... | 31 |
| 110 | 8.1 Overview..... | 31 |
| 111 | 8.1.1 Discovery Query Patterns..... | 32 |
| 112 | 8.1.2 Parameter \$name..... | 32 |
| 113 | 8.1.3 Parameter \$description..... | 32 |
| 114 | 8.1.4 Parameter \$targetNamespace..... | 32 |
| 115 | 8.1.5 Parameter \$importedNamespace..... | 32 |

| | | |
|-----|---|----|
| 116 | 8.1.6 Example of WSDL Document Discovery Query..... | 33 |
| 117 | 8.2 PortType Discovery Query..... | 33 |
| 118 | 8.2.1 Parameter \$portType.name..... | 33 |
| 119 | 8.2.2 Parameter \$portType.description..... | 33 |
| 120 | 8.2.3 Parameter \$portType.targetNamespace..... | 33 |
| 121 | 8.2.4 Parameter \$portType.schemaNamespaces..... | 33 |
| 122 | 8.2.5 Example of WSDL PortType Discovery Query..... | 33 |
| 123 | 8.3 WSDL Binding Discovery Query..... | 34 |
| 124 | 8.3.1 Parameter \$binding.name..... | 34 |
| 125 | 8.3.2 Parameter \$binding.description..... | 34 |
| 126 | 8.3.3 Parameter \$binding.targetNamespace..... | 34 |
| 127 | 8.3.4 Parameter \$binding.protocolType..... | 34 |
| 128 | 8.3.5 Parameter \$binding.transportType..... | 34 |
| 129 | 8.3.6 Parameter \$binding.soapStyle..... | 34 |
| 130 | 8.3.7 Parameter \$considerPortType..... | 34 |
| 131 | 8.3.8 Parameter \$portType.name..... | 34 |
| 132 | 8.3.9 Parameter \$portType.description..... | 35 |
| 133 | 8.3.10 Parameter \$portType.targetNamespace..... | 35 |
| 134 | 8.3.11 Parameter \$portType.schemaNamespace..... | 35 |
| 135 | 8.3.12 Example of WSDL Binding Discovery Query..... | 35 |
| 136 | 8.4 WSDL Port Discovery Query..... | 35 |
| 137 | 8.4.1 Parameter \$port.name..... | 35 |
| 138 | 8.4.2 Parameter \$port.description..... | 35 |
| 139 | 8.4.3 Parameter \$port.targetNamespace..... | 36 |
| 140 | 8.4.4 Parameter \$port.accessURI..... | 36 |
| 141 | 8.4.5 Parameter \$considerBinding..... | 36 |
| 142 | 8.4.6 Parameter \$binding.name..... | 36 |
| 143 | 8.4.7 Parameter \$binding.description..... | 36 |
| 144 | 8.4.8 Parameter \$binding.targetNamespace..... | 36 |
| 145 | 8.4.9 Parameter \$binding.protocolType..... | 36 |
| 146 | 8.4.10 Parameter \$binding.transportType..... | 36 |
| 147 | 8.4.11 Parameter \$binding.soapStyle..... | 36 |
| 148 | 8.4.12 Parameter \$considerPortType..... | 36 |
| 149 | 8.4.13 Parameter \$portType.name..... | 37 |
| 150 | 8.4.14 Parameter \$portType.description..... | 37 |
| 151 | 8.4.15 Parameter \$portType.targetNamespace..... | 37 |
| 152 | 8.4.16 Parameter \$portType.schemaNamespace..... | 37 |
| 153 | 8.4.17 Example of WSDL Port Discovery Query..... | 37 |
| 154 | 8.5 WSDL Service Discovery Query..... | 37 |
| 155 | 8.5.1 Parameter \$service.name..... | 37 |
| 156 | 8.5.2 Parameter \$service.description..... | 37 |
| 157 | 8.5.3 Parameter \$service.targetNamespace..... | 38 |
| 158 | 8.5.4 Parameter \$considerPort..... | 38 |
| 159 | 8.5.5 Parameter \$port.name..... | 38 |
| 160 | 8.5.6 Parameter \$port.description..... | 38 |

| | | |
|-----|---|----|
| 161 | 8.5.7 Parameter \$port.targetNamespace..... | 38 |
| 162 | 8.5.8 Parameter \$port.accessURI..... | 38 |
| 163 | 8.5.9 Parameter \$considerBinding..... | 38 |
| 164 | 8.5.10 Parameter \$binding.name..... | 38 |
| 165 | 8.5.11 Parameter \$binding.description..... | 38 |
| 166 | 8.5.12 Parameter \$binding.targetNamespace..... | 38 |
| 167 | 8.5.13 Parameter \$binding.protocolType..... | 38 |
| 168 | 8.5.14 Parameter \$binding.transportType..... | 39 |
| 169 | 8.5.15 Parameter \$binding.soapStyle..... | 39 |
| 170 | 8.5.16 Parameter \$considerPortType..... | 39 |
| 171 | 8.5.17 Parameter \$portType.name..... | 39 |
| 172 | 8.5.18 Parameter \$portType.description..... | 39 |
| 173 | 8.5.19 Parameter \$portType.targetNamespace..... | 39 |
| 174 | 8.5.20 Parameter \$portType.schemaNamespace..... | 39 |
| 175 | 8.5.21 Example of WSDL Service Discovery Query..... | 39 |
| 176 | 9 Event Notification Profile..... | 41 |
| 177 | 9.1 Subscribing to a WSDL Document..... | 41 |
| 178 | 9.2 Subscribing to PortType changes..... | 41 |
| 179 | 9.3 Subscribing to Binding changes..... | 42 |
| 180 | 9.4 Subscribing to Port changes..... | 42 |
| 181 | 9.5 Subscribing to Service changes..... | 42 |
| 182 | 10 Security Profile..... | 43 |
| 183 | 10.1 SubjectRole Profile..... | 43 |
| 184 | 10.2 SubjectGroup Profile..... | 43 |
| 185 | 10.3 AccessControlPolicy Profile..... | 43 |
| 186 | 11 Canonical Metadata Definitions..... | 44 |
| 187 | 11.1 ObjectType Extensions..... | 44 |
| 188 | 11.2 Canonical ClassificationSchemes..... | 44 |
| 189 | 11.3 Canonical Queries..... | 46 |
| 190 | 11.3.1 WSDL Document Discovery Query..... | 46 |
| 191 | 11.3.2 WSDL PortType Discovery Query..... | 46 |
| 192 | 11.3.3 WSDL Binding Discovery Query..... | 47 |
| 193 | 11.3.4 WSDL Port Discovery Query..... | 48 |
| 194 | 11.3.5 WSDL Service Discovery Query..... | 49 |
| 195 | 12 References..... | 52 |
| 196 | 12.1 Normative References..... | 52 |
| 197 | 12.2 Informative References..... | 52 |
| 198 | | |

Illustration Index

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

Index of Tables

200

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

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

245 • **Constants**

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

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

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:

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

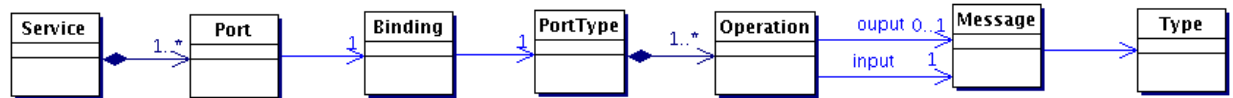


Illustration 1: WSDL Information Model

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 more Ports to provide an implementation of a web service complete with all its concrete protocol specific interfaces.

2.1 Element <service>

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

2.2 Element <port>

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

2.3 Element <binding>

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

2.4 Element <portType>

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

2.5 Element <operation>

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

2.6 Element <message>

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

294 **2.7 Element <types>**

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

3 ebXML Registry Overview

This chapter provides an overview of ebXML Registry Information Model [ebRIM] and an overview of the specific domain and/or application. The [ebRIM] is the target for the mapping patterns defined by this document and the specific domain is the source information model. The information presented is informative and is not intended to replace the normative information defined by ebXML Registry.

3.1 Overview of [ebRIM]

This section is provided in the « Deployment Profile Template for ebXML V3 specs » and can be removed in a specific profile. Normally only specifics topics needs to be developed here (but the profile editor can prefer to leave it) This section summarizes the ebXML Registry Information Model [ebRIM]. This model is the target of the mapping defined in this document. The reader SHOULD read [CMRR] for a more detailed overview of ebXML Registry as a whole

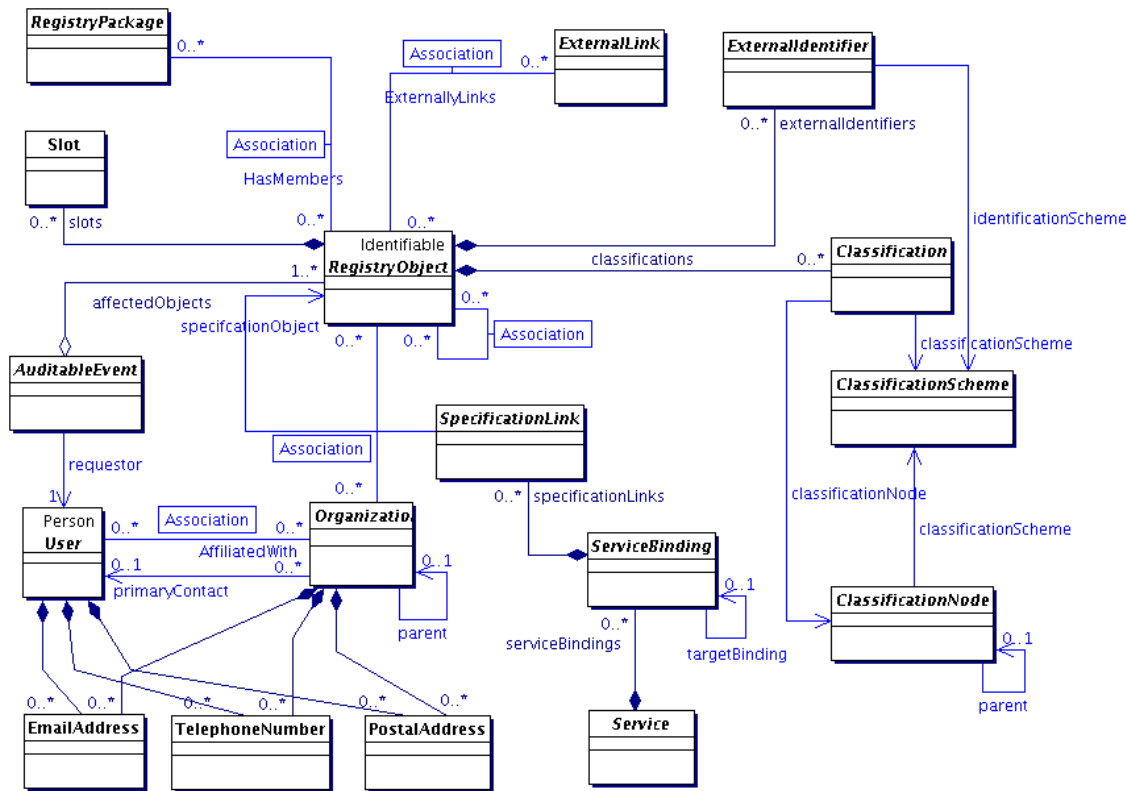
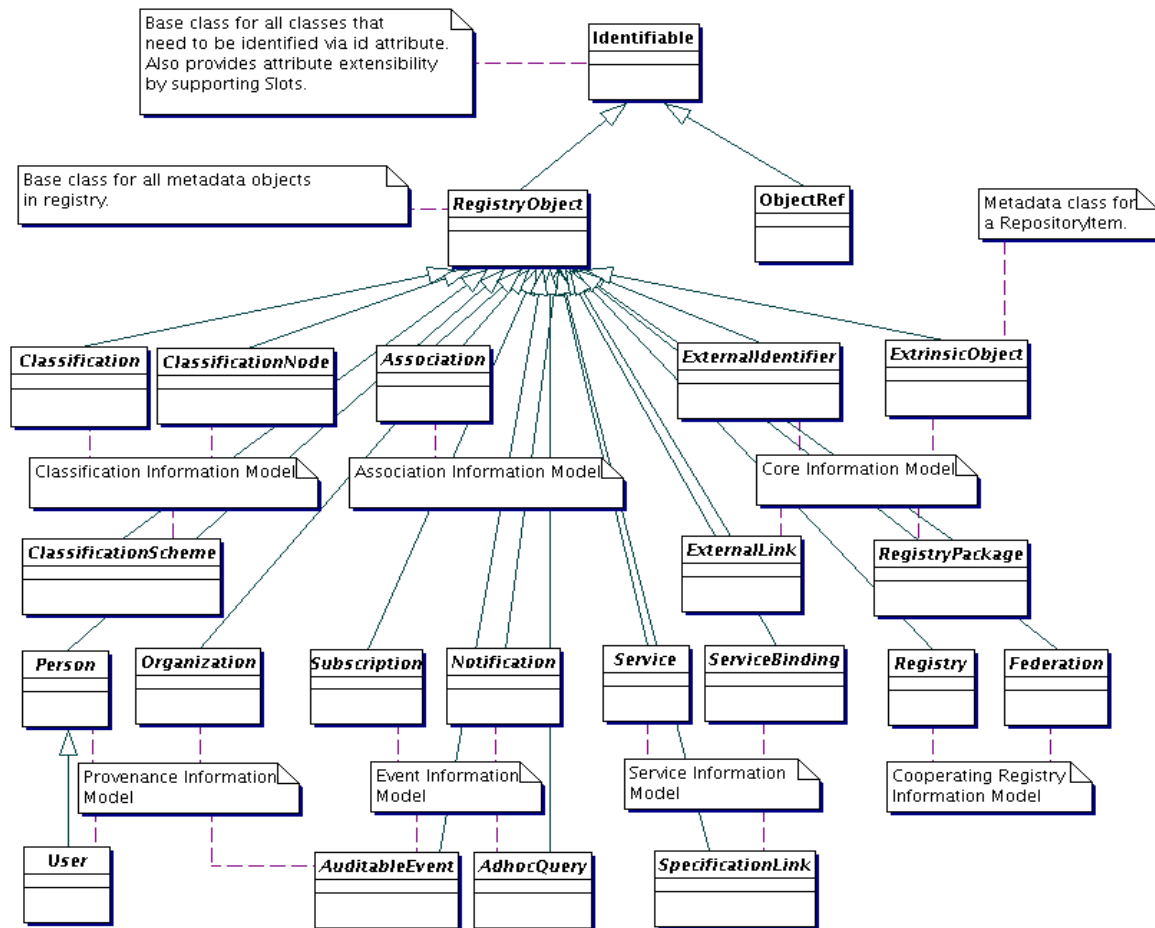


Figure 1: ebXML Registry Information Model, High Level Public View

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



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

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

324 **3.1.1 RegistryObject**

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

329 **3.1.2 Object Identification**

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

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

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

334 The id attribute value MAY potentially be human friendly.

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

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

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

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

342

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

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

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

348

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

356
357
358 **Listing 4: Example of ExternalIdentifier**

359 3.1.3 Object Naming and Description

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

362

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

377
378
379 **Listing 5: Example of Name and Description**

380

381 3.1.4 Object Attributes

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

384 3.1.4.1 Slot Attributes

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

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

391

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

```

394 <rim:Slot name="urn:acme:slot:NASDAQSymbol">
395 <rim:ValueList>
396 <rim:Value>ACME</rim:Value>
397 </rim:ValueList>
398 </rim:Slot>
399
400 <rim:Name>
401 <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
402 </rim:Name>
403 <rim:Description>
404 <rim:LocalizedString value="ACME makes Java. Provider of free Java
405 software."
406 xml:lang="en-US"/>
407 </rim:Description>
408 <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
409 identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
410 value="ACME"/>
411 </rim:ExternalIdentifier>
412 </rim:Organization>
413

```

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.

```

421 <rim:Organization id=${ACME_ORG_ID}
422 lid="urn:acme:ACMEOrganization">
423 <rim:Slot name="urn:acme:slot:NASDAQSymbol">
424 <rim:ValueList>
425 <rim:Value>ACME</rim:Value>
426 </rim:ValueList>
427 </rim:Slot>
428 <rim:Name>
429 <rim:LocalizedString value="ACME Inc." xml:lang="en-US"/>
430 </rim:Name>
431 <rim:Description>
432 <rim:LocalizedString value="ACME makes Java. Provider of free Java
433 software." xml:lang="en-US"/>
434 </rim:Description>
435 <rim:ExternalIdentifier id=${EXTERNAL_IDENTIFIER_ID}
436 identificationScheme=${DUNS_CLASSIFICATIONSCHEME_ID}
437 value="ACME"/>
438 </rim:ExternalIdentifier>
439
440 <!--Classify Organization as a Software Publisher using NAICS Taxonomy-->
441 <rim:Classification id=${CLASSIFICATION_ID}
442 classificationNode=${NAICS_SOFTWARE_PUBLISHER_NODE_ID}
443 classifiedObject=${ACME_ORG_ID}>
444
445 </rim:Organization>

```

Listing 7: Example of Object Classification

3.1.6 Object Association

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

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

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

```

457 <rim:Association
458 id=${ASSOCIATION_ID}

```

```

459     associationType=${CANONICAL_ASSOCIATION_TYPE_OFFERS_SERVICE_ID}
460     sourceObject=${ACME_ORG_ID}
461     targetObject=${ACME_SERVICE1_ID}/>

```

Listing 8: Example of Object Association

3.1.7 Object References To Web Content

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

```

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

```

Listing 9: Example of Reference to Web Content Using ExternalLink

3.1.8 Object Packaging

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

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

```

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

```

Listing 10: Example of Object Packaging Using RegistryPackages

3.1.9 Service Description

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

3.2 Overview of [ebRS]

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

4 Mapping WSDL Information Model to [ebRIM]

511

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

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

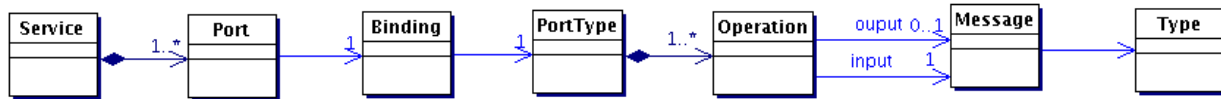


Illustration 2: WSDL Information Model

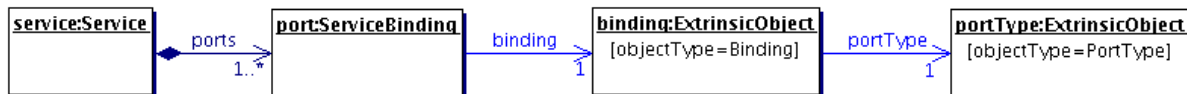


Illustration 3: Mapping of WSDL Information Model to ebRIM

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

4.1 wsd:service → rim:Service Mapping

521

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

523

```
524 <service name="ebXMLRegistrySOAPService">  
525 <port binding="bindings:QueryManagerSOAPBinding"  
526 name="QueryManagerPort">  
527 <soap:address location="http://your.server.com/soap"/>  
528 </port>  
529 <port binding="bindings:LifeCycleManagerSOAPBinding"  
530 name="LifeCycleManagerPort">  
531 <soap:address location="http://your.server.com/soap"/>  
532 </port>  
533 </service>
```

534

Example wsd:service

4.1.1 Attribute id

535

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

539

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

548

Example of rim:Service id Attribute Mapping

549 4.1.2 Element Name

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

553

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

564

Example of rim:Service name Attribute Mapping

565 4.1.3 Element Description

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

570

```
571 WSDL fragment:  
572 <wsdl:service name="ebXMLRegistrySOAPService">  
573 <wsdl:documentation>  
574 An implementation of ebXML Registry 3.0  
575 </wsdl:documentation>  
576 </wsdl:service>  
577  
578 <rim:Service  
579 id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:Service:  
580 ebXMLRegistrySOAPService">  
581 <rim:Description>  
582 <rim:LocalizedString value="An implementation of ebXML Registry  
583 3.0"/>  
584 </rim:Description>  
585 </rim:Service>
```

586

Example of rim:Service description Attribute Mapping

587 4.1.4 Elements Classification

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

590

| 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 |

591

```
592 <rim:Service  
593 id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:Service:  
594 ebXMLRegistrySOAPService">  
595
```

```

596     <Classification classificationScheme="urn:oasis:names:tc:ebxml-
597     regrep:ObjectType" classificationNode="urn:oasis:names:tc:ebxml-
598     regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Service"/>
599
600 </rim:Service>

```

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

602 4.1.5 Elements ServiceBinding

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

```

606 <rim:Service
607   id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:Service:
608   ebXMLRegistrySOAPService">
609   <rim:ServiceBinding .../>
610   <rim:ServiceBinding .../>
611 </rim:Service>

```

612 **Example of wsdl:port → rim:ServiceBinding Mapping**

613 4.2 wsdl:port → rim:ServiceBinding Mapping

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

615 4.2.1 Attribute id

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

```

620 <wsdl:port binding="bindings:QueryManagerSOAPBinding
621   name="QueryManagerPort">
622
623   <rim:ServiceBinding
624     id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:port:
625     QueryManagerPort">

```

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

627 4.2.2 Element Name

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

```

632 <wsdl:port binding="bindings:QueryManagerSOAPBinding
633   name="QueryManagerPort">
634
635   <rim:ServiceBinding
636     id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:port:
637     QueryManagerPort">
638     <rim:Name>
639       <rim:LocalizedString value="QueryManagerPort"/>
640     </rim:Name>
641   </rim:ServiceBinding>

```

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

643 **4.2.3 Element Description**

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

648

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

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

665 **4.2.4 Attribute accessURI**

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

669

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

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

680 **4.2.5 Attribute service**

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

683 **4.2.6 Element Classification**

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

686

| 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 |

687

688

```

689 <wsdl:port binding="bindings:QueryManagerSOAPBinding
690   name="QueryManagerPort">
691   <soap:address location="http://your.server.com/soap"/>
692 </wsdl:port>
693
694 <rim:ServiceBinding
695   id="urn:acmeinc:ebxml:registry:3.0:services:wsdl:port:
696   QueryManagerPort" accessURI="http://your.server.com/soap">
697
698   <Classification classificationScheme="urn:oasis:names:tc:ebxml-
699   regrep:ObjectType" classificationNode="urn:oasis:names:tc:ebxml-
700   regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Port"/>
701
702 </rim:ServiceBinding>

```

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

704 4.3 wsdl:binding → rim:ExtrinsicObject Mapping

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

706 4.3.1 Attribute objectType

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

```

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

```

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

715 4.3.2 Attribute id

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

719

```

720 <binding name="QueryManagerSOAPBinding"
721   type="interfaces:QueryManagerPortType">
722
723 <rim:ExtrinsicObject
724   id="urn:acmeinc:ebxml:registry:3.0:ExtrinsicObject:wsdl:binding:
725   QueryManagerSOAPBinding">

```

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

727 4.3.3 Element Name

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

731

```

732 <binding name="QueryManagerSOAPBinding"
733   type="interfaces:QueryManagerPortType">
734
735 <rim:ExtrinsicObject

```

```

736     id="urn:acmeinc:ebxml:registry:3.0:services:wSDL:binding:
737     QueryManagerSOAPBinding">
738     <rim:Name>
739     <rim:LocalizedString value="QueryManagerSOAPBinding"/>
740     </rim:Name>
741 </rim:ExtrinsicObject>

```

742 **Example of rim:ExtrinsicObject name element mapping for wSDL:binding**

743 4.3.4 Element Description

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

748

```

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

```

764 **Example of rim:ExtrinsicObject description element Mapping for wSDL:binding**

765 4.3.5 Element Classification

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

768

| 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. |

769

770

```

771 <binding name="QueryManagerSOAPBinding"
772 type="interfaces:QueryManagerPortType">
773 <soap:binding style="document "
774 transport="http://schemas.xmlsoap.org/soap/http"/>
775 ...
776 </binding>
777
778 <rim:ExtrinsicObject

```

```

780     <Classification classificationScheme="urn:oasis:names:tc:ebxml-
781     regrep:profile:ws:classificationScheme:ProtocolType"
782     classificationNode="urn:oasis:names:tc:ebxml-
783     regrep:profile:ws:ProtocolType:SOAP"/>
784
785     <Classification classificationScheme="urn:oasis:names:tc:ebxml-
786     regrep:profile:ws:classificationScheme:TransportType"
787     classificationNode="urn:oasis:names:tc:ebxml-
788     regrep:profile:ws:TransportType:HTTP"/>
789
790     <Classification classificationScheme="urn:oasis:names:tc:ebxml-
791     regrep:profile:ws:classificationScheme:SOAPStyle"
792     classificationNode="urn:oasis:names:tc:ebxml-
793     regrep:profile:ws:SOAPStyle:Document"/>
794
795 </rim:ExtrinsicObject>

```

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

797 4.4 wsdl:portType → rim:ExtrinsicObject Mapping

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

800 4.4.1 Attribute objectType

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

```

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

```

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

809 4.4.2 Attribute id

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

813

```

814 TargetNamespace= urn:oasis:names:tc:ebxml-
815 regrep:wsdl:registry:interfaces:3.0
816 <portType name="QueryManagerPortType"/>
817
818 <rim:ExtrinsicObject
819   id="urn:oasis:names:tc:ebxml-
820   regrep:wsdl:registry:interfaces:3.0:portType:QueryManagerPortType" ...>

```

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

822 4.4.3 Element Name

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

826

```

827 <wsdl:portType name="QueryManagerPortType">
828
829 <rim:ExtrinsicObject

```

```

830     id="urn:oasis:names:tc:ebxml-
831     regrep:wsl:registry:interfaces:3.0:portType:QueryManagerPortType">
832     <rim:Name>
833       <rim:LocalizedString value="QueryManagerPortType"/>
834     </rim:Name>
835   </rim:ExtrinsicObject>

```

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

837 4.4.4 Element Description

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

842

```

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

```

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

858 4.5 wsdl:port ↔ wsdl:binding to rim:Association Mapping

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

860 4.5.1 Attribute id

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

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

863 4.5.2 Attribute sourceObject

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

866 4.5.3 Attribute targetObject

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

869 4.5.4 Attribute associationType

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

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

872 which is the id of the canonical "Implements" ClassificationNode within the canonical AssociationType
873 ClassificationScheme.

874 **4.6 wsdl:binding ↔wsdl:portType Association Mapping**

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

876 **4.6.1 Attribute id**

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

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

879 **4.6.2 Attribute sourceObject**

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

882 **4.6.3 Attribute targetObject**

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

885 **4.6.4 Attribute associationType**

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

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

888 which is the id of the canonical "Implements" ClassificationNode within the canonical AssociationType
889 ClassificationScheme.

890 **5 Publishing Profile**

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

893 **5.1 Structure of WSDL Documents**

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

898 **5.1.1 XxInterfaces.wsdl**

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

900 **5.1.2 XxBindings.wsdl**

901 This file **SHOULD** contain binding elements.

902 **5.1.3 XxServices.wsdl**

903 This file **SHOULD** contain the service elements.

904 6 Validation Service Profile

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

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

911 6.1 Invocation Control File

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

915 6.2 Business Rules

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

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

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

931 7 Cataloging Service Profile

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

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

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

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

943 7.1 Invocation Control File

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

946 7.2 Input Metadata

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

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

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

```
958 <rim:ExtrinsicObject id="urn:acmeinc:ebxml:registry:3.0:services:wSDL">  
959 ...  
960 </rim:ExtrinsicObject>
```

961 Example of ExtrinsicObject Input Metadata

```
963 <rim:ExternalLink  
964 id="urn:acmeinc:ebxml:registry:3.0:services:wSDL"  
965 externalURI="http://www.acme.com/wSDL/ebXMLRegistryService.wSDL"  
966 >  
967 ...  
968 </rim:ExternalLink>
```

969 Example of ExternalLink Input Metadata

970 7.3 Input Content

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

976 defined in ??.

977 **7.4 Output Metadata**

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

979 **7.4.1 Changes to Input Metadata**

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

982 **Slot importedNameSpaces**

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

987 **Slot targetNamespace**

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

992 **7.4.2 wsdL:service → rim:Service**

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

996 **7.4.3 wsdL:port → rim:ServiceBinding**

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

1000 **7.4.4 wsdL:binding → rim:ExtrinsicObject**

1001 The WSDL Cataloging service MUST automatically produce an rim:ExtrinsicObject instance for each
1002 wsdL:binding element within the input WSDL or its imports, as specified in the wsdL:binding →
1003 rim:ExtrinsicObject mapping earlier in this document.

1004 **7.4.5 wsdL:portType → rim:ExtrinsicObject**

1005 The WSDL Cataloging service MUST automatically produce an rim:ExtrinsicObject instance for each
1006 wsdL:portType element within the input WSDL or its imports, as specified in the wsdL:portType →
1007 rim:ExtrinsicObject mapping earlier in this document.

1008 **7.4.6 wsdL:port ↔ wsdL:binding Association**

1009 The WSDL Cataloging service MUST automatically produce rim:Association instances for each wsdL:port
1010 element within the input WSDL or its imports, as specified in the wsdL:port → wsdL:binding Association
1011 mapping earlier in this document.

1012 **7.4.7 wsdL:binding ↔ wsdL:portType Association**

1013 The WSDL Cataloging service MUST automatically produce rim:Association instances for each
1014 wsdL:binding element within the input WSDL or its imports, as specified in the wsdL:binding →

1015 wsdl:portType Association mapping earlier in this document.

1016

8 Discovery Profile

1017 The ebXML Registry provides the ability for a user defined parameterized queries to be configured for
1018 each type of content. The queries may be as complex or simple as the discovery use case requires. The
1019 complexity of the parameterized queries may hidden from the registry client by storing them within the
1020 ebXML Registry as instances of the AdhocQuery class, and being invoked by simply providing their
1021 parameters. Query parameters are often pattern strings that may contain wildcard characters '%'
1022 (matches any number of characters) and '_' (matches exactly one character) as described by [ebRS].

1023 An ebXML Registry SHOULD provide a graphical user interface that displays any configured
1024 parameterized query as a form which contains an appropriate field for entering each query parameter.

1025

The screenshot shows a web form titled 'Discovery'. At the top, there are 'Search' and 'Clear Form' buttons. Below them are radio buttons for 'Business Query' (unselected) and 'Ad hoc Query' (selected). A 'Search Criteria' section contains a 'Select Query' dropdown menu with 'WSDL Service Discovery Query' selected. The main section is 'Ad Hoc Query Parameters', which includes numerous input fields and dropdown menus for configuring search criteria. Fields include Service Name, Service Description, Service Status (StatusType), Service Target Namespace, Consider Port (checkbox), Port Name, Port Description, Port Status (StatusType), Port Target Namespace, Port Endpoint, Consider Binding (checkbox), Binding Name, Binding Description, Binding Status (StatusType), Binding Target Namespace, Binding Protocol Type (ProtocolType), Binding Transport Type (TransportType), SOAP Binding Style (SOAPStyleType), Consider PortType (checkbox), PortType Name, PortType Description, PortType StatusPortType (StatusType), and PortType Target Namespace.

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

1027

1028 This chapter defines the queries that MUST be support by an ebXML Registry implementing the WS
1029 Profile for discovering WSDL content. An implementation MAY also support additional discovery queries
1030 for WSDL content.

1031 8.1 Overview

1032 Refer to the layered architecture of WSDL information model described in chapter 2.

1033 Discovery is the process that enables discovering higher level objects based on search criteria that
1034 predicates upon attributes of the type of object being discovered as well. as the attributes of lower level
1035 objects in the model.

1036 In contrast, drill-down is the process that enables explore a higher level object and determining the lower
1037 level objects that were used in building the higher level object.

1038 [ebRIM] provides support for drill-down use cases in a generic manner out-of-box. For example, [ebRIM]
1039 provides:

- 1040 • A generic ability to explore all rim:ServiceBindings used within a rim:Service
- 1041 • A generic ability to explore all Association involving a ServiceBinding or any RegistryObject

1042 The queries defined in this chapter are therefor focused on discovery rather than drill-down. Because
1043 ebXML Registry supports a flexible and extensible query capability, this chapter defines a single
1044 discovery query for each discoverable type within the WSDL information model. Each query may be as
1045 complex as necessary. However, the complexity is hidden from the client using parameterized queries
1046 stored in the Registry as instances of the AdhocQuery type, in the same manner as any other
1047 RegistryObject.

1048 In the subsequent section each query is described simply in terms of its supported parameters that serve
1049 as its search criteria. The actual AdhocQuery instances are much more complex in comparison but they
1050 are not exposed to the client making the query. Details on these queries are specified canonically in
1051 chapter 11.

1052 **8.1.1 Discovery Query Patterns**

1053 The discovery queries are specified from the bottom of the layered WSDL information model to the top
1054 (portType, binding, port and service). The query for each layer specifies parameters specific to it as well
1055 as parameters specific to each of the lower layers that it builds upon. Thus the number of parameters
1056 increase as queries are defined for higher level types in the model. This is key to being able to discover
1057 higher level objects based on attributes of the lower level objects that they build upon.

1058 There are many parameters supported for the discovery query for each higher level type in the model.
1059 However, it is often the case that discovery may not require parameters specific to all lower level types.
1060 To facilitate pruning of the discovery query for unwanted predicates related to lower level types there is a
1061 special parameter name \$considerXXX where XXX represents a lower level type within the model. If the
1062 value of this parameter is set to "0" then all parameter values specific to lower level type MUST are
1063 ignored by the discovery query.

1064 **WSDL Document Discovery Query**

1065 The WSDL Document discovery query MUST be implemented by an ebXML Registry implementing this
1066 profile. It allows the discovery of WSDL documents using zero or more of the parameters described next.

1067 **8.1.2 Parameter \$name**

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

1070 **8.1.3 Parameter \$description**

1071 This parameter's value MAY specify a string containing a pattern to match against the description
1072 attribute value of RegistryObjects that have objectType of WSDL.

1073 **8.1.4 Parameter \$targetNamespace**

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

1076 **8.1.5 Parameter \$importedNamespace**

1077 This parameter's value MAY specify a string containing a pattern to match against the namespaces
1078 imported by a WSDL document.

1079 8.1.6 Example of WSDL Document Discovery Query

1080 The following example illustrates how to find all WSDL documents that have a targetNamespace
1081 containing the string "oasis" and that import a namespace with string "org.w3". Note that additional
1082 supported parameters MAY also be specified if needed.

1083

```
1084 <AdhocQueryRequest>  
1085   <rim:AdhocQuery id="urn:oasis:names:tc:ebxml-  
1086   regrep:profile:ws:query:WSDLDiscoveryQuery">  
1087     <rim:Slot name="$targetNamespace">  
1088       <rim:ValueList>  
1089         <rim:Value>%oasis%</rim:Value>  
1090       </rim:ValueList>  
1091     </rim:Slot>  
1092     <rim:Slot name="$importedNamespace">  
1093       <rim:ValueList>  
1094         <rim:Value>%org.w3%</rim:Value>  
1095       </rim:ValueList>  
1096     </rim:Slot>  
1097   </rim:AdhocQuery>  
1098 </AdhocQueryRequest>
```

1099

Example of WSDL Document Discovery Query

1100 8.2 PortType Discovery Query

1101 The WSDL PortType discovery query allows the discovery of wsdl:portType instances using zero or
1102 more of the parameters described next.

1103 8.2.1 Parameter \$portType.name

1104 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1105 value of wsdl:portType instances.

1106 8.2.2 Parameter \$portType.description

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

1109 8.2.3 Parameter \$portType.targetNamespace

1110 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1111 of wsdl:portType instances.

1112 8.2.4 Parameter \$portType.schemaNamespaces

1113 This parameter's value MAY specify a string containing a pattern to match against the XML schema
1114 namespaces used within the wsdl:message instances used within the wsdl:operation instances used
1115 within the wsdl:portType instances.

1116 8.2.5 Example of WSDL PortType Discovery Query

1117 The following example illustrates how to find all wsdl:portType instances that have a name containing the
1118 string "QueryManager" and have import an XML Schema with namespace containing the string "ebxml-
1119 regrep". Note that additional supported parameters MAY also be specified if needed.

1120

```
1121 <AdhocQueryRequest>  
1122   <rim:AdhocQuery id="urn:oasis:names:tc:ebxml-  
1123   regrep:profile:ws:query:WSDLPortTypeDiscoveryQuery">  
1124     <rim:Slot name="$portType.name">
```

```

1125     <rim:ValueList>
1126         <rim:Value>%QueryManager%</rim:Value>
1127     </rim:ValueList>
1128 </rim:Slot>
1129 <rim:Slot name="$portType.schemaNamespaces">
1130     <rim:ValueList>
1131         <rim:Value>%ebxml-regrep%</rim:Value>
1132     </rim:ValueList>
1133 </rim:Slot>
1134 </rim:AdhocQuery>
1135 </AdhocQueryRequest>

```

1136 **Example of WSDL PortType Discovery Query**

1137 **8.3 WSDL Binding Discovery Query**

1138 The WSDL Binding discovery query allows the discovery of wsdl:binding instances using zero or more of
1139 the parameters described next.

1140 **8.3.1 Parameter \$binding.name**

1141 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1142 value of wsdl:binding instances.

1143 **8.3.2 Parameter \$binding.description**

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

1146 **8.3.3 Parameter \$binding.targetNamespace**

1147 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1148 of wsdl:binding instances.

1149 **8.3.4 Parameter \$binding.protocolType**

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 protocolType supported by wsdl:binding instances.

1152 **8.3.5 Parameter \$binding.transportType**

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 transportType supported by wsdl:binding instances.

1155 **8.3.6 Parameter \$binding.soapStyle**

1156 This parameter's value MAY specify a string containing a pattern to match against the id attribute value
1157 of the ClassificationNode representing the soap style of wsdl:binding instances.

1158 **8.3.7 Parameter \$considerPortType**

1159 This parameter's value MAY specify a string a "1" or "0" to indicate whether or not to consider the
1160 portType specific parameters that follow when processing the query. If unspecified the value defaults to
1161 "0".

1162 **8.3.8 Parameter \$portType.name**

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

1165 **8.3.9 Parameter \$portType.description**

1166 This parameter's value MAY specify a string containing a pattern to match against the content of the
1167 wsdl:documentation element within wsdl:portType instances that are used by the objects being
1168 discovered.

1169 **8.3.10 Parameter \$portType.targetNamespace**

1170 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1171 of wsdl:portType instances that are used by the objects being discovered.

1172 **8.3.11 Parameter \$portType.schemaNamespace**

1173 This parameter's value MAY specify a string containing a pattern to match against the XML schema
1174 namespaces used within the wsdl:message instances used within the wsdl:operation instances used
1175 within the wsdl:portType instances that are used by the objects being discovered.

1176 **8.3.12 Example of WSDL Binding Discovery Query**

1177 The following example illustrates how to find all wsdl:binding instances that have a name containing the
1178 string "QueryManager" and have a binding that supports SOAP protocol using HTTP transport. Note that
1179 additional supported parameters MAY also be specified if needed.

1180

```
1181 <AdhocQueryRequest>  
1182   <rim:AdhocQuery id="urn:oasis:names:tc:ebxml-  
1183   regrep:profile:ws:query:WSDLBindingDiscoveryQuery">  
1184     <rim:Slot name="$binding.name">  
1185       <rim:ValueList>  
1186         <rim:Value>%QueryManager%</rim:Value>  
1187       </rim:ValueList>  
1188     </rim:Slot>  
1189     <rim:Slot name="$binding.protocolType">  
1190       <rim:ValueList>  
1191         <rim:Value>urn:oasis:names:tc:ebxml-  
1192   regrep:profile:ws:ProtocolType:SOAP</rim:Value>  
1193       </rim:ValueList>  
1194     </rim:Slot>  
1195   </rim:AdhocQuery>  
1196 </AdhocQueryRequest>
```

1197

Example of WSDL Port Discovery Query

1198 **8.4 WSDL Port Discovery Query**

1199 The WSDL Port discovery query allows the discovery of wsdl:port instances using zero or more of the
1200 parameters described next.

1201 **8.4.1 Parameter \$port.name**

1202 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1203 value of wsdl:port instances.

1204 **8.4.2 Parameter \$port.description**

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

1207 **8.4.3 Parameter \$port.targetNamespace**

1208 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1209 of wsdl:port instances.

1210 **8.4.4 Parameter \$port.accessURI**

1211 This parameter's value MAY specify a string containing a pattern to match against the accessURI for the
1212 endpoint defined for the wsdl:port instances.

1213 **8.4.5 Parameter \$considerBinding**

1214 This parameter's value MAY specify a string a "1" or "0" to indicate whether or not to consider the binding
1215 specific parameters that follow when processing the query. If unspecified the value defaults to "0".

1216 **8.4.6 Parameter \$binding.name**

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

1219 **8.4.7 Parameter \$binding.description**

1220 This parameter's value MAY specify a string containing a pattern to match against the content of the
1221 wsdl:documentation element within wsdl:binding instances that are used by the objects being
1222 discovered.

1223 **8.4.8 Parameter \$binding.targetNamespace**

1224 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1225 of wsdl:binding instances that are used by the objects being discovered.

1226 **8.4.9 Parameter \$binding.protocolType**

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 protocolType supported by wsdl:binding instances that are
1229 used by the objects being discovered.

1230 **8.4.10 Parameter \$binding.transportType**

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 transportType supported by wsdl:binding instances that are
1233 used by the objects being discovered.

1234 **8.4.11 Parameter \$binding.soapStyle**

1235 This parameter's value MAY specify a string containing a pattern to match against the id attribute value
1236 of the ClassificationNode representing the soap style of wsdl:binding instances that are used by the
1237 objects being discovered.

1238 **8.4.12 Parameter \$considerPortType**

1239 This parameter's value MAY specify a string a "1" or "0" to indicate whether or not to consider the
1240 portType specific parameters that follow when processing the query. If unspecified the value defaults to
1241 "0".

1242 **8.4.13 Parameter \$portType.name**

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

1245 **8.4.14 Parameter \$portType.description**

1246 This parameter's value MAY specify a string containing a pattern to match against the content of the
1247 wsdl:documentation element within wsdl:portType instances that are used by the objects being
1248 discovered.

1249 **8.4.15 Parameter \$portType.targetNamespace**

1250 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1251 of wsdl:portType instances that are used by the objects being discovered.

1252 **8.4.16 Parameter \$portType.schemaNamespace**

1253 This parameter's value MAY specify a string containing a pattern to match against the XML schema
1254 namespaces used within the wsdl:message instances used within the wsdl:operation instances used
1255 within the wsdl:portType instances that are used by the objects being discovered.

1256 **8.4.17 Example of WSDL Port Discovery Query**

1257 The following example illustrates how to find all wsdl:port instances that have a name containing the
1258 string "QueryManager" and have a binding that supports SOAP protocol using HTTP transport. Note that
1259 additional supported parameters MAY also be specified if needed.

1260

```
1261 <AdhocQueryRequest>  
1262   <rim:AdhocQuery id="urn:oasis:names:tc:ebxml-  
1263   regrep:profile:ws:query:WSDLPortDiscoveryQuery">  
1264     <rim:Slot name="$port.name">  
1265       <rim:ValueList>  
1266         <rim:Value>%QueryManager%</rim:Value>  
1267       </rim:ValueList>  
1268     </rim:Slot>  
1269     <rim:Slot name="$port.accessURI">  
1270       <rim:ValueList>  
1271         <rim:Value>http://acme%</rim:Value>  
1272       </rim:ValueList>  
1273     </rim:Slot>  
1274   </rim:AdhocQuery>  
1275 </AdhocQueryRequest>
```

1276

Example of WSDL Port Discovery Query

1277 **8.5 WSDL Service Discovery Query**

1278 The WSDL Service discovery query allows the discovery of wsdl:service instances using zero or more of
1279 the parameters described next.

1280 **8.5.1 Parameter \$service.name**

1281 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1282 value of wsdl:service instances.

1283 **8.5.2 Parameter \$service.description**

1284 This parameter's value MAY specify a string containing a pattern to match against the content of the
1285 wsdl:documentation element within wsdl:service instances.

1286 **8.5.3 Parameter \$service.targetNamespace**

1287 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1288 of wsdl:service instances.

1289 **8.5.4 Parameter \$considerPort**

1290 This parameter's value MAY specify a string a "1" or "0" to indicate whether or not to consider the port
1291 specific parameters that follow when processing the query. If unspecified the value defaults to "0".

1292 **8.5.5 Parameter \$port.name**

1293 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1294 value of wsdl:port instances that are used by the objects being discovered.

1295 **8.5.6 Parameter \$port.description**

1296 This parameter's value MAY specify a string containing a pattern to match against the content of the
1297 wsdl:documentation element within wsdl:port instances that are used by the objects being discovered.

1298 **8.5.7 Parameter \$port.targetNamespace**

1299 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1300 of wsdl:port instances that are used by the objects being discovered.

1301 **8.5.8 Parameter \$port.accessURI**

1302 This parameter's value MAY specify a string containing a pattern to match against the accessURI for the
1303 endpoint defined for the wsdl:port instances that are used by the objects being discovered.

1304 **8.5.9 Parameter \$considerBinding**

1305 This parameter's value MAY specify a string a "1" or "0" to indicate whether or not to consider the
1306 binding specific parameters that follow when processing the query. If unspecified the value defaults to
1307 "0".

1308 **8.5.10 Parameter \$binding.name**

1309 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1310 value of wsdl:binding instances that are used by the objects being discovered.

1311 **8.5.11 Parameter \$binding.description**

1312 This parameter's value MAY specify a string containing a pattern to match against the content of the
1313 wsdl:documentation element within wsdl:binding instances that are used by the objects being
1314 discovered.

1315 **8.5.12 Parameter \$binding.targetNamespace**

1316 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1317 of wsdl:binding instances that are used by the objects being discovered.

1318 **8.5.13 Parameter \$binding.protocolType**

1319 This parameter's value MAY specify a string containing a pattern to match against the id attribute value
1320 of the ClassificationNode representing the protocolType supported by wsdl:binding instances that are
1321 used by the objects being discovered.

1322 **8.5.14 Parameter \$binding.transportType**

1323 This parameter's value MAY specify a string containing a pattern to match against the id attribute value
1324 of the ClassificationNode representing the transportType supported by wsdl:binding instances that are
1325 used by the objects being discovered.

1326 **8.5.15 Parameter \$binding.soapStyle**

1327 This parameter's value MAY specify a string containing a pattern to match against the id attribute value
1328 of the ClassificationNode representing the soap style of wsdl:binding instances that are used by the
1329 objects being discovered.

1330 **8.5.16 Parameter \$considerPortType**

1331 This parameter's value MAY specify a string a "1" or "0" to indicate whether or not to consider the
1332 portType specific parameters that follow when processing the query. If unspecified the value defaults to
1333 "0".

1334 **8.5.17 Parameter \$portType.name**

1335 This parameter's value MAY specify a string containing a pattern to match against the name attribute
1336 value of wsdl:portType instances that are used by the objects being discovered.

1337 **8.5.18 Parameter \$portType.description**

1338 This parameter's value MAY specify a string containing a pattern to match against the content of the
1339 wsdl:documentation element within wsdl:portType instances that are used by the objects being
1340 discovered.

1341 **8.5.19 Parameter \$portType.targetNamespace**

1342 This parameter's value MAY specify a string containing a pattern to match against the targetNamespace
1343 of wsdl:portType instances that are used by the objects being discovered.

1344 **8.5.20 Parameter \$portType.schemaNamespace**

1345 This parameter's value MAY specify a string containing a pattern to match against the XML schema
1346 namespaces used within the wsdl:message instances used within the wsdl:operation instances used
1347 within the wsdl:portType instances that are used by the objects being discovered.

1348 **8.5.21 Example of WSDL Service Discovery Query**

1349 The following example illustrates how to find all wsdl:service instances that have a name containing the
1350 string "QueryManager" and have a targetNamespace containing the string "ebXML". Note that additional
1351 supported parameters MAY also be specified if needed.

1352

```
1353 <AdhocQueryRequest>  
1354   <rim:AdhocQuery id="urn:oasis:names:tc:ebxml-  
1355   regrep:profile:ws:query:WSDLServiceDiscoveryQuery">  
1356     <rim:Slot name="$name">  
1357       <rim:ValueList>  
1358         <rim:Value>%QueryManager%</rim:Value>  
1359       </rim:ValueList>  
1360     </rim:Slot>  
1361     <rim:Slot name="$targetNamespace">  
1362       <rim:ValueList>  
1363         <rim:Value>%ebXML%</rim:Value>  
1364       </rim:ValueList>  
1365     </rim:Slot>
```

1366
1367
1368

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

Example of WSDL Service Discovery Query

9 Event Notification Profile

1369

1370 The ebXML Registry provides the ability for a user or an automated service to create a subscription to
1371 events that match a specified criteria. Whenever an event matching the specified criteria occurs, the
1372 registry notifies the subscriber that the event transpired. The event matching criteria is specified using a
1373 stored parameterized query similar to the discovery queries described in previous chapter.

1374 This chapter specifies the template Subscriptions that MUST be implemented by an ebXML Registry
1375 implementing the Web Services profile. To subscribe to instances of types defined within the WSDL
1376 information model a subscription can simply reuse the discovery queries defined in the previous chapter
1377 or define more specific queries.

9.1 Subscribing to a WSDL Document

1378

1379 A client may publish a rim:Subscription instance using the WSDL Document discovery query as the
1380 selector in order to receive notification of changes to WSDL documents that they have an interest in.

1381

```
1382 <rim:Subscription id=${WSDL_SUBSCRIPTION_ID}  
1383   selector="urn:oasis:names:tc:ebxml-  
1384   regrep:profile:ws:query:WSDLDiscoveryQuery">  
1385   <rim:Slot name="$name">  
1386     <rim:ValueList>  
1387       <rim:Value>%ebXML%</rim:Value>  
1388     </rim:ValueList>  
1389   </rim:Slot>  
1390   <rim:Slot name="$targetNamespace">  
1391     <rim:ValueList>  
1392       <rim:Value>%oasis%</rim:Value>  
1393     </rim:ValueList>  
1394   </rim:Slot>  
1395  
1396   <!-- email address endPoint for receiving notification via email -->  
1397   <rim:NotifyAction notificationOption="urn:oasis:names:tc:ebxml-  
1398   regrep:NotificationOptionType:ObjectRefs"  
1399     endPoint="mailto:farrukh.najmi@sun.com"/>  
1400  
1401   <!-- Web Service endPoint for receiving notification via SOAP -->  
1402   <rim:NotifyAction notificationOption="urn:oasis:names:tc:ebxml-  
1403   regrep:NotificationOptionType:Objects"  
1404     endPoint="urn:acme:wSDLChangeListenerService"/>  
1405 </rim:Subscription>
```

1406

Listing 11: Example of Subscription to WSDL Documents

1407

1408 The above example show how to create a subscription for WSDL Documents where name contains the
1409 string “ebXML” and targetNamespace contains the string “oasis”. Light-weight notifications containing
1410 references to WSDL documents are configured to be sent to an email address while heavy-weight
1411 notifications containing actual WSDL documents are configured to be sent to a listener service using the
1412 SOAP protocol.

9.2 Subscribing to PortType changes

1413

1414 A client may publish a rim:Subscription instance using the WSDL PortType discovery query as the
1415 selector in order to receive notification of changes to WSDL PortTypes that they have an interest in.

1416

```
1417 <rim:Subscription id=${WSDL_SUBSCRIPTION_ID}  
1418   selector="urn:oasis:names:tc:ebxml-  
1419   regrep:profile:ws:query:PortTypeDiscoveryQuery">  
1420   ...  
1421 </rim:Subscription>
```

1422

1423

Listing 12: Example of Subscription to WSDL PortTypes

1424 9.3 Subscribing to Binding changes

1425 A client may publish a rim:Subscription instance using the WSDL Binding discovery query as the selector in
1426 order to receive notification of changes to WSDL PortBindings that they have an interest in.

1427

```
1428 <rim:Subscription id=${WSDL_SUBSCRIPTION_ID}  
1429   selector="urn:oasis:names:tc:ebxml-  
1430   regrep:profile:ws:query:BindingDiscoveryQuery">  
1431  
1432   ...  
1433 </rim:Subscription>
```

1434

Listing 13: Example of Subscription to WSDL Bindings

1435 9.4 Subscribing to Port changes

1436 A client may publish a rim:Subscription instance using the WSDL Port discovery query as the selector in
1437 order to receive notification of changes to WSDL Ports that they have an interest in.

1438

```
1439 <rim:Subscription id=${WSDL_SUBSCRIPTION_ID}  
1440   selector="urn:oasis:names:tc:ebxml-  
1441   regrep:profile:ws:query:PortDiscoveryQuery">  
1442  
1443   ...  
1444 </rim:Subscription>
```

1445

Listing 14: Example of Subscription to WSDL Ports

1446 9.5 Subscribing to Service changes

1447 A client may publish a rim:Subscription instance using the WSDL Service discovery query as the selector
1448 in order to receive notification of changes to WSDL Services that they have an interest in.

1449

```
1450 <rim:Subscription id=${WSDL_SUBSCRIPTION_ID}  
1451   selector="urn:oasis:names:tc:ebxml-  
1452   regrep:profile:ws:query:ServiceDiscoveryQuery">  
1453  
1454   ...  
1455 </rim:Subscription>
```

1456

Listing 15: Example of Subscription to WSDL Ports

1457

1458 **10 Security Profile**

1459 This chapter specifies security aspects governing the publish, management and discovery of web
1460 services within ebXML Registry.

1461 **10.1 SubjectRole Profile**

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

1465 **10.2 SubjectGroup Profile**

1466 The ebXML Registry defines a set of pre-defined roles in the *SubjectGroup* scheme. A domain specific
1467 mapping to ebRIM MAY define additional domain specific groups by extending the SubjectGroup
1468 scheme.
1469 TBD??

1470 **10.3 AccessControlPolicy Profile**

1471 The ebXML Registry provides a powerful and extensible access control feature that makes sure that a
1472 user may only perform those actions on a RegistryObject or repository item for which they are
1473 authorized.
1474 ...
1475 TBD??

1476

11 Canonical Metadata Definitions

1477 This chapter specifies the canonical metadata defined by this profile.

11.1 ObjectType Extensions

1479 The following new extensions to the canonical ObjectType ClassificationScheme are described by this
1480 profile:

1481

```

1482     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
1483 regrep:ObjectType:RegistryObject:ExtrinsicObject"
1484 lid="urn:oasis:names:tc:ebxml-
1485 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL" code="WSDL"
1486 id="urn:oasis:names:tc:ebxml-
1487 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL">
1488     <rim:Name>
1489       <rim:LocalizedString charset="UTF-8" value="label.WSDL"/>
1490     </rim:Name>
1491     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
1492 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL"
1493 lid="urn:oasis:names:tc:ebxml-
1494 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType"
1495 code="PortType" id="urn:oasis:names:tc:ebxml-
1496 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType">
1497     <rim:Name>
1498       <rim:LocalizedString charset="UTF-8" value="label.PortType"/>
1499     </rim:Name>
1500     </rim:ClassificationNode>
1501     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
1502 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL"
1503 lid="urn:oasis:names:tc:ebxml-
1504 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding"
1505 code="Binding" id="urn:oasis:names:tc:ebxml-
1506 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding">
1507     <rim:Name>
1508       <rim:LocalizedString charset="UTF-8" value="label.Binding"/>
1509     </rim:Name>
1510     </rim:ClassificationNode>
1511     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
1512 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL"
1513 lid="urn:oasis:names:tc:ebxml-
1514 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Port" code="Port"
1515 id="urn:oasis:names:tc:ebxml-
1516 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Port">
1517     <rim:Name>
1518       <rim:LocalizedString charset="UTF-8" value="label.Port"/>
1519     </rim:Name>
1520     </rim:ClassificationNode>
1521     <rim:ClassificationNode parent="urn:oasis:names:tc:ebxml-
1522 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL"
1523 lid="urn:oasis:names:tc:ebxml-
1524 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Service"
1525 code="Service" id="urn:oasis:names:tc:ebxml-
1526 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Service">
1527     <rim:Name>
1528       <rim:LocalizedString charset="UTF-8" value="label.Service"/>
1529     </rim:Name>
1530     </rim:ClassificationNode>
1531   </rim:ClassificationNode>

```

1532

Listing 16: Example of Subscription to WSDL Ports

1533

11.2 Canonical ClassificationSchemes

1535 The following new canonical ClassificationSchemes are described by this profile:

```

1536     <rim:ClassificationScheme lid="urn:oasis:names:tc:ebxml-
1537 regrep:profile:ws:classificationScheme:ProtocolType"
1538 id="urn:oasis:names:tc:ebxml-
1539 regrep:profile:ws:classificationScheme:ProtocolType" isInternal="true"
1540 nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode">
1541     <rim:Name>
1542     <rim:LocalizedString charset="UTF-8"
1543 value="label.ProtocolType"/>
1544     </rim:Name>
1545     <rim:Description>
1546     <rim:LocalizedString charset="UTF-8"
1547 value="label.ProtocolType.desc"/>
1548     </rim:Description>
1549     <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1550 regrep:profile:ws:ProtocolType:SOAP" code="SOAP"
1551 id="urn:oasis:names:tc:ebxml-regrep:profile:ws:ProtocolType:SOAP"/>
1552     <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1553 regrep:profile:ws:ProtocolType:AS2" code="AS2"
1554 id="urn:oasis:names:tc:ebxml-regrep:profile:ws:ProtocolType:AS2"/>
1555     <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1556 regrep:profile:ws:ProtocolType:Atom" code="Atom"
1557 id="urn:oasis:names:tc:ebxml-regrep:profile:ws:ProtocolType:Atom"/>
1558     </rim:ClassificationScheme>
1559
1560     <rim:ClassificationScheme lid="urn:oasis:names:tc:ebxml-
1561 regrep:profile:ws:classificationScheme:TransportType"
1562 id="urn:oasis:names:tc:ebxml-
1563 regrep:profile:ws:classificationScheme:TransportType" isInternal="true"
1564 nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode">
1565     <rim:Name>
1566     <rim:LocalizedString charset="UTF-8"
1567 value="label.TransportType"/>
1568     </rim:Name>
1569     <rim:Description>
1570     <rim:LocalizedString charset="UTF-8"
1571 value="label.TransportType.desc"/>
1572     </rim:Description>
1573     <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1574 regrep:profile:ws:TransportType:HTTP" code="HTTP"
1575 id="urn:oasis:names:tc:ebxml-regrep:profile:ws:TransportType:HTTP"/>
1576     <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1577 regrep:profile:ws:TransportType:MOM" code="MOM"
1578 id="urn:oasis:names:tc:ebxml-regrep:profile:ws:TransportType:MOM"/>
1579     <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1580 regrep:profile:ws:TransportType:BEEP" code="BEEP"
1581 id="urn:oasis:names:tc:ebxml-regrep:profile:ws:TransportType:BEEP"/>
1582     </rim:ClassificationScheme>
1583
1584     <rim:ClassificationScheme lid="urn:oasis:names:tc:ebxml-
1585 regrep:profile:ws:classificationScheme:SOAPStyleType"
1586 id="urn:oasis:names:tc:ebxml-
1587 regrep:profile:ws:classificationScheme:SOAPStyleType" isInternal="true"
1588 nodeType="urn:oasis:names:tc:ebxml-regrep:NodeType:UniqueCode">
1589     <rim:Name>
1590     <rim:LocalizedString charset="UTF-8"
1591 value="label.SOAPStyleType"/>
1592     </rim:Name>
1593     <rim:Description>
1594     <rim:LocalizedString charset="UTF-8"
1595 value="label.SOAPType.desc"/>
1596     </rim:Description>
1597     <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1598 regrep:profile:wsPStyleType:RPC" code="RPC"
1599 id="urn:oasis:names:tc:ebxml-regrep:profile:wsPStyleType:RPC"/>
1600     <rim:ClassificationNode lid="urn:oasis:names:tc:ebxml-
1601 regrep:profile:wsPStyleType:Document" code="Document"
1602 id="urn:oasis:names:tc:ebxml-regrep:profile:wsPStyleType:Document"/>
1603     </rim:ClassificationScheme>

```

1604

Listing 17: Example of Subscription to WSDL Ports

1605 11.3 Canonical Queries

1606 The following new canonical queries are described by this profile. Note that while these queries are
1607 complex, the complexity is hidden from clients by exposing only the query parameters to them.

1608 11.3.1 WSDL Document Discovery Query

```
1609 <!--  
1610 Parameterized Adhoc Query for WSDL Discovery query.  
1611 Finds by Name, Description, ComponentType, ResourceType  
1612 -->  
1613 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-  
1614 regrep:profile:ws:query:WSDLDiscoveryQuery"  
1615 id="urn:oasis:names:tc:ebxml-  
1616 regrep:profile:ws:query:WSDLDiscoveryQuery">  
1617 <rim:Name>  
1618 <rim:LocalizedString value="label.WSDLDiscoveryQuery"/>  
1619 </rim:Name>  
1620 <rim:Description>  
1621 <rim:LocalizedString value="label.WSDLDiscoveryQuery.desc"/>  
1622 </rim:Description>  
1623 <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-  
1624 regrep:QueryLanguage:SQL-92">  
1625 SELECT DISTINCT ro.* FROM RegistryObject ro, Name_ nm, Description d,  
1626 Slot tns, Slot ins  
1627 WHERE (1=1)  
1628 AND (ro.objectType = 'urn:oasis:names:tc:ebxml-  
1629 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL')  
1630 AND (nm.parent = ro.id AND UPPER ( nm.value ) LIKE UPPER ( '$name'  
1631 ) )  
1632 AND (d.parent = ro.id AND UPPER ( d.value ) LIKE UPPER  
1633 ( '$description' ) )  
1634 AND (ro.id = tns.parent  
1635 AND tns.name_ = 'urn:oasis:names:tc:ebxml-  
1636 regrep:profile:ws:wSDL:targetNamespace'  
1637 AND tns.value LIKE '$targetNamespace')  
1638 AND (ro.id = ins.parent  
1639 AND ins.name_ = 'urn:oasis:names:tc:ebxml-  
1640 regrep:profile:ws:wSDL:importedNamespaces'  
1641 AND ins.value LIKE '$importedNamespaces')  
1642 </rim:QueryExpression>  
1643 </rim:AdhocQuery>
```

1645 Listing 18: WSDL Document Discovery Query

1646 11.3.2 WSDL PortType Discovery Query

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

```

1669 AND (portTypeDesc.parent = portType.id AND UPPER ( portTypeDesc.value )
1670 LIKE UPPER ( '$portType.description' ) )
1671 AND (portType.id = s.parent
1672 AND portTypeTNS.name_ = 'urn:oasis:names:tc:ebxml-
1673 regrep:profile:ws:wSDL:targetNamespace'
1674 AND portTypeTNS.value LIKE '$portType.targetNamespace')
1675 </rim:QueryExpression>
1676 </rim:AdhocQuery>

```

1677 **Listing 19: WSDL PortType Discovery Query**

1678 11.3.3 WSDL Binding Discovery Query

```

1679 <!--
1680 Parameterized Adhoc Query for WSDL Binding Discovery query.
1681 -->
1682 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
1683 regrep:profile:ws:query:BindingDiscoveryQuery"
1684 id="urn:oasis:names:tc:ebxml-
1685 regrep:profile:ws:query:BindingDiscoveryQuery">
1686 <rim:Name>
1687 <rim:LocalizedString value="label.BindingDiscoveryQuery"/>
1688 </rim:Name>
1689 <rim:Description>
1690 <rim:LocalizedString value="label.BindingDiscoveryQuery.desc"/>
1691 </rim:Description>
1692 <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1693 regrep:QueryLanguage:SQL-92">
1694 SELECT DISTINCT binding.* FROM
1695 ExtrinsicObject binding, Name_ bindingName, Description bindingDesc,
1696 Slot bindingTNS,
1697 Association implements
1698 WHERE
1699 binding.objectType = 'urn:oasis:names:tc:ebxml-
1700 regrep:Object:RegistryObject:ExtrinsicObject:WSDL:Binding'
1701 AND (bindingName.parent = binding.id AND UPPER ( bindingName.value )
1702 LIKE UPPER ( '$binding.name' ) )
1703 AND (bindingDesc.parent = binding.id AND UPPER ( bindingDesc.value )
1704 LIKE UPPER ( '$binding.description' ) )
1705 AND (binding.id = s.parent
1706 AND bindingTNS.name_ = 'urn:oasis:names:tc:ebxml-
1707 regrep:profile:ws:wSDL:targetNamespace'
1708 AND bindingTNS.value LIKE '$binding.targetNamespace')
1709 AND (binding.id IN ( SELECT classifiedObject FROM Classification WHERE
1710 classificationNode IN ( SELECT id
1711 FROM ClassificationNode WHERE path LIKE '$binding.protocolType' ) ) )
1712 AND (binding.id IN ( SELECT classifiedObject FROM Classification WHERE
1713 classificationNode IN ( SELECT id
1714 FROM ClassificationNode WHERE path LIKE
1715 '$binding.transportType%' ) ) )
1716 AND (binding.id IN ( SELECT classifiedObject FROM Classification WHERE
1717 classificationNode IN ( SELECT id
1718 FROM ClassificationNode WHERE path LIKE
1719 '$binding.soapStyleType%' ) ) )
1720
1721 AND ($considerPortType = 0 OR (
1722 implements.sourceObject=binding.id AND
1723 implements.associationType='urn:oasis:names:tc:ebxml-
1724 regrep:AssociationType:Implements' AND implements.targetObject IN
1725 (
1726 SELECT DISTINCT portType.id from ExtrinsicObject portType, Name_
1727 portTypeName, Description portTypeDesc, Slot portTypeTNS
1728 WHERE
1729 portType.objectType = 'urn:oasis:names:tc:ebxml-
1730 regrep:Object:RegistryObject:ExtrinsicObject:WSDL:PortType'
1731 AND (portTypeName.parent = portType.id AND UPPER
1732 ( portTypeName.value ) LIKE UPPER ( '$portType.name' ) )
1733 AND (portTypeDesc.parent = portType.id AND UPPER
1734 ( portTypeDesc.value ) LIKE UPPER ( '$portType.description' ) )
1735 AND (portType.id = s.parent
1736 AND portTypeTNS.name_ = 'urn:oasis:names:tc:ebxml-
1737 regrep:profile:ws:wSDL:targetNamespace'

```

```

1738     AND portTypeTNS.value LIKE ('$portType.targetNamespace')
1739   ))
1740 )
1741   </rim:QueryExpression>
1742 </rim:AdhocQuery>

```

Listing 20: WSDL Binding Discovery Query

11.3.4 WSDL Port Discovery Query

```

1745 <!--
1746 Parameterized Adhoc Query for WSDL Port Discovery query.
1747 -->
1748 <rim:AdhocQuery lid="urn:oasis:names:tc:ebxml-
1749 regrep:profile:ws:query:PortDiscoveryQuery"
1750 id="urn:oasis:names:tc:ebxml-
1751 regrep:profile:ws:query:PortDiscoveryQuery">
1752   <rim:Name>
1753     <rim:LocalizedString value="label.PortDiscoveryQuery"/>
1754   </rim:Name>
1755   <rim:Description>
1756     <rim:LocalizedString value="label.PortDiscoveryQuery.desc"/>
1757   </rim:Description>
1758   <rim:QueryExpression queryLanguage="urn:oasis:names:tc:ebxml-
1759 regrep:QueryLanguage:SQL-92">
1760     SELECT DISTINCT port.* FROM
1761     ServiceBinding port, Name_ portName, Description portDesc, Slot
1762     portTNS,
1763     Association implements
1764     WHERE
1765     (port.id IN ( SELECT classifiedObject FROM Classification WHERE
1766     classificationNode = 'urn:oasis:names:tc:ebxml-
1767     regrep:Object:RegistryObject:ExtrinsicObject:WSDL:Port' ) )
1768
1769     AND (portName.parent = port.id AND UPPER ( portName.value ) LIKE UPPER
1770     ( '$port.name' ) )
1771     AND (portDesc.parent = port.id AND UPPER ( portDesc.value ) LIKE UPPER
1772     ( '$port.description' ) )
1773     AND (port.id = s.parent
1774     AND portTNS.name_ = 'urn:oasis:names:tc:ebxml-
1775     regrep:profile:ws:wSDL:targetNamespace'
1776     AND portTNS.value LIKE '$port.targetNamespace')
1777     AND (port.accessURI LIKE '$port.accessURI')
1778
1779     AND ($considerBinding = 0 OR (
1780     implements.sourceObject=port.id AND
1781     implements.associationType='urn:oasis:names:tc:ebxml-
1782     regrep:AssociationType:Implements' AND implements.targetObject IN
1783     (
1784     SELECT DISTINCT binding.id FROM
1785     ExtrinsicObject binding, Name_ bindingName, Description
1786     bindingDesc, Slot bindingTNS,
1787     Association implements
1788     WHERE
1789     binding.objectType = 'urn:oasis:names:tc:ebxml-
1790     regrep:Object:RegistryObject:ExtrinsicObject:WSDL:Binding'
1791     AND (bindingName.parent = binding.id AND UPPER
1792     ( bindingName.value ) LIKE UPPER ( '$binding.name' ) )
1793     AND (bindingDesc.parent = binding.id AND UPPER
1794     ( bindingDesc.value ) LIKE UPPER ( '$binding.description' ) )
1795     AND (binding.id = s.parent
1796     AND bindingTNS.name_ = 'urn:oasis:names:tc:ebxml-
1797     regrep:profile:ws:wSDL:targetNamespace'
1798     AND bindingTNS.value LIKE '$binding.targetNamespace')
1799     AND (binding.id IN ( SELECT classifiedObject FROM Classification
1800     WHERE classificationNode IN ( SELECT id
1801     FROM ClassificationNode WHERE path LIKE '$binding.protocolType' )
1802     ) )
1803     AND (binding.id IN ( SELECT classifiedObject FROM Classification
1804     WHERE classificationNode IN ( SELECT id
1805     FROM ClassificationNode WHERE path LIKE '$binding.transportType%'
1806     ) ) )

```



```

1807         AND (binding.id IN ( SELECT classifiedObject FROM Classification
1808 WHERE classificationNode IN ( SELECT id
1809         FROM ClassificationNode WHERE path LIKE '$binding.soapStyleType%'
1810 ) ) )
1811
1812         AND ($considerPortType = 0 OR (
1813         implements.sourceObject=binding.id AND
1814         implements.associationType='urn:oasis:names:tc:ebxml-
1815         regrep:AssociationType:Implements' AND implements.targetObject IN
1816         (
1817         SELECT DISTINCT portType.id from ExtrinsicObject portType, Name_
1818         portTypeName, Description portTypeDesc, Slot portTypeTNS
1819         WHERE
1820         portType.objectType = 'urn:oasis:names:tc:ebxml-
1821         regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType'
1822         AND (portTypeName.parent = portType.id AND UPPER
1823         ( portTypeName.value ) LIKE UPPER ( '$portType.name' ) )
1824         AND (portTypeDesc.parent = portType.id AND UPPER
1825         ( portTypeDesc.value ) LIKE UPPER ( '$portType.description' ) )
1826         AND (portType.id = s.parent
1827         AND portTypeTNS.name_ = 'urn:oasis:names:tc:ebxml-
1828         regrep:profile:ws:wSDL:targetNamespace'
1829         AND portTypeTNS.value LIKE '$portType.targetNamespace')
1830         )
1831         ) )
1832     ) )
1833 )
1834     </rim:QueryExpression>
1835 </rim:AdhocQuery>

```

1836

Listing 21: WSDL Port Discovery Query

1837 11.3.5 WSDL Service Discovery Query

```

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

```

```

1876         ServiceBinding port, Name_ portName, Description portDesc, Slot
1877 portTNS,
1878         Association implements
1879         WHERE
1880         (port.id IN ( SELECT classifiedObject FROM Classification WHERE
1881 classificationNode = 'urn:oasis:names:tc:ebxml-
1882 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Port' ) )
1883
1884         AND (portName.parent = port.id AND UPPER ( portName.value ) LIKE
1885 UPPER ( '$port.name' ) )
1886         AND (portDesc.parent = port.id AND UPPER ( portDesc.value ) LIKE
1887 UPPER ( '$port.description' ) )
1888         AND (port.id = s.parent
1889         AND portTNS.name_ = 'urn:oasis:names:tc:ebxml-
1890 regrep:profile:ws:wSDL:targetNamespace'
1891         AND portTNS.value LIKE '$port.targetNamespace')
1892         AND (port.accessURI LIKE '$port.accessURI')
1893
1894         AND ($considerBinding = 0 OR (
1895         implements.sourceObject=port.id AND
1896         implements.associationType='urn:oasis:names:tc:ebxml-
1897 regrep:AssociationType:Implements' AND implements.targetObject IN
1898         (
1899         SELECT DISTINCT binding.id FROM
1900         ExtrinsicObject binding, Name_ bindingName, Description
1901 bindingDesc, Slot bindingTNS,
1902         Association implements
1903         WHERE
1904         binding.objectType = 'urn:oasis:names:tc:ebxml-
1905 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:Binding'
1906         AND (bindingName.parent = binding.id AND UPPER
1907 ( bindingName.value ) LIKE UPPER ( '$binding.name' ) )
1908         AND (bindingDesc.parent = binding.id AND UPPER
1909 ( bindingDesc.value ) LIKE UPPER ( '$binding.description' ) )
1910         AND (binding.id = s.parent
1911         AND bindingTNS.name_ = 'urn:oasis:names:tc:ebxml-
1912 regrep:profile:ws:wSDL:targetNamespace'
1913         AND bindingTNS.value LIKE '$binding.targetNamespace')
1914         AND (binding.id IN ( SELECT classifiedObject FROM
1915 Classification WHERE classificationNode IN ( SELECT id
1916 FROM ClassificationNode WHERE path LIKE
1917 '$binding.protocolType' ) ) )
1918         AND (binding.id IN ( SELECT classifiedObject FROM
1919 Classification WHERE classificationNode IN ( SELECT id
1920 FROM ClassificationNode WHERE path LIKE
1921 '$binding.transportType%' ) ) )
1922         AND (binding.id IN ( SELECT classifiedObject FROM
1923 Classification WHERE classificationNode IN ( SELECT id
1924 FROM ClassificationNode WHERE path LIKE
1925 '$binding.soapStyleType%' ) ) )
1926
1927         AND ($considerBinding = 0 OR (
1928         implements.sourceObject=binding.id AND
1929         implements.associationType='urn:oasis:names:tc:ebxml-
1930 regrep:AssociationType:Implements' AND implements.targetObject IN
1931         (
1932         SELECT DISTINCT portType.id from ExtrinsicObject portType,
1933 Name_ portTypeName, Description portTypeDesc, Slot portTypeTNS
1934         WHERE
1935         portType.objectType = 'urn:oasis:names:tc:ebxml-
1936 regrep:ObjectType:RegistryObject:ExtrinsicObject:WSDL:PortType'
1937         AND (portTypeName.parent = portType.id AND UPPER
1938 ( portTypeName.value ) LIKE UPPER ( '$portType.name' ) )
1939         AND (portTypeDesc.parent = portType.id AND UPPER
1940 ( portTypeDesc.value ) LIKE UPPER ( '$portType.description' ) )
1941         AND (portType.id = s.parent
1942         AND portTypeTNS.name_ = 'urn:oasis:names:tc:ebxml-
1943 regrep:profile:ws:wSDL:targetNamespace'
1944         AND portTypeTNS.value LIKE '$portType.targetNamespace')
1945         )
1946         ))
1947         )

```

1948
1949
1950
1951
1952
1953

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

Listing 22: WSDL Service Discovery Query

1954 **12 References**

1955 **12.1 Normative References**

1956 [ebRIM] ebXML Registry Information Model version 3.0

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

1958

1959 [ebRS] ebXML Registry Services Specification version 3.0

1960 <http://docs.oasis-open.org/regrep/regrep-rs/v3.0/regrep-rs-3.0-os.pdf>

1961 [UML] Unified Modeling Language version 1.5

1962 <http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.pdf>

1963 [ebRR-DPT] Deployment Profile Template For ebXML Registry 3.0 OASIS Specifications V_0.1.2

1964 [ebMS-DPT] Deployment Profile Template For OASIS Specification ebXML Message Service 2.0

1965 [WSDL] WSDL Specification

1966 <http://www.w3.org/TR/wSDL>

1967 **12.2 Informative References**

1968 **Appendix Anformative**

1969 [WSDL-OVW] WSDL Essentials

1970 <http://www.developer.com/services/article.php/1602051>

1971 [IMPL] ebXML Registry 3.0 Implementations

1972 freebXML Registry: A royalty free, open source ebXML Registry Implementation

1973 <http://ebxmlrr.sourceforge.net>

1974