



Creating A Single Global Electronic Market

1

2

## 3 **Technical Note**

# 4 **Registering Web Services in an ebXML Registry, Version** 5 **2.0**

6 **June 2005**

## 7 **Authors**

8

9 Joseph M. Chiusano, Booz Allen Hamilton

10 Farrukh Najmi, Sun Microsystems

11

## 12 **Abstract**

13

14 This document describes the current best practice for registering Web services in an  
15 ebXML Registry. It conforms to the following specifications:

16

17 OASIS/ebXML Registry Information Model (ebRIM) v3.0

18

18 OASIS/ebXML Registry Services Specification (ebRS), v3.0

19

20 This version supercedes the March 2003 version, which was based on earlier v3.0  
21 specification versions that were not yet ratified.

22

23 These specifications can be found at <http://www.oasis-open.org/committees/regrep/>.

24

## 25 **Status of this Document**

26

27 This document is an OASIS Registry Technical Committee Technical Note. Distribution  
28 of this document is unlimited.

29

30

31 **TABLE OF CONTENTS**

32 **ABSTRACT ..... 1**

33 **STATUS OF THIS DOCUMENT ..... 1**

34 **1 INTRODUCTION ..... 4**

35 **1 INTRODUCTION ..... 4**

36 **2 WEB SERVICES ..... 4**

37 **3 RELEVANT EBXML REGISTRY CLASSES.....ERROR! BOOKMARK NOT**  
 38 **DEFINED.**

39 **3.1 Class Service ..... 5**

40 **3.2 Class ServiceBinding ..... 6**

41 **3.3 Class SpecificationLink ..... 7**

42 **4 FULL SUBMITOBJECTSREQUEST EXAMPLE ..... 8**

43 **5 EXTENDED SCENARIOS ..... 9**

44 **5.1 Versioning of Web Services..... 9**

45 **5.2 Associating a Web Service with an Organization ..... 11**

46 **5.3 Associating a Web Service with an Access Control Policy..... 12**

47 **5.4 Registering a Service Description that is External to the Registry ..... 12**

48 **5.5 Web Service Redirection ..... 13**

49 **5.6 Customizing Metadata Using Slots..... 14**

50 **APPENDIX A WSDL INTRODUCTION ..... 15**

51 **APPENDIX B OASIS/EBXML COLLABORATION-PROTOCOL PROFILE**  
 52 **AND AGREEMENT (CPP/A) INTRODUCTION..... 15**

53 **APPENDIX C DAML-S INTRODUCTION ..... 16**

54

55 **Figures**

56 **FIGURE 1: RELATIONSHIP BETWEEN RIM CLASSES SERVICE,**  
57 **SERVICEBINDING, AND SPECIFICATIONLINK ..... 5**

58 **FIGURE 2: ASSOCIATING A WEB SERVICE WITH AN ORGANIZATION ..... 11**

59 **FIGURE 3: REGISTERING AN EXTERNAL SERVICE DESCRIPTION ..... 13**

**1**

## 60 **Introduction**

61 An ebXML Registry is an information system that securely manages any content type  
62 and the standardized metadata that describes it. The ebXML Registry also provides a set  
63 of services that enable sharing of content and metadata between organizational entities in  
64 a federated environment. Submitted content may be XML schema and documents,  
65 process descriptions, Web services, ebXML Core Components, context descriptions,  
66 UML models, information about parties and even software components.

67  
68 The purpose of this document is to provide a Best Practice for registering Web services  
69 and their associated entities in an ebXML Registry.

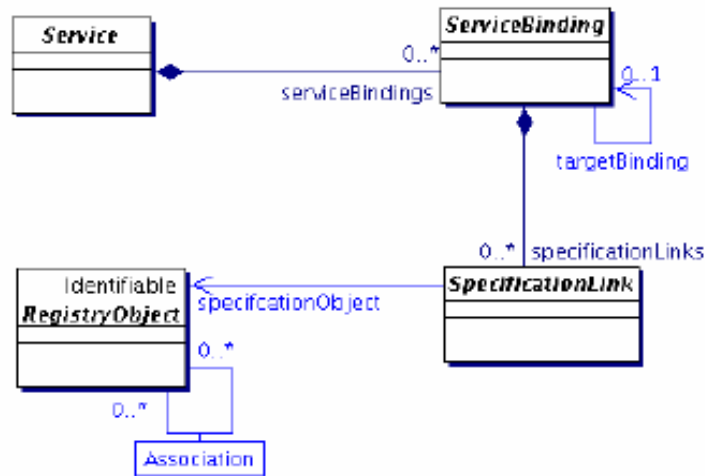
## 70 **2 Describing Web Services**

71 The most common mechanism for describing Web services today is the Web Services  
72 Description Language, or WSDL [WSDL]; however, the Service description that is  
73 registered can be in any format such as OASIS/ebXML Collaboration-Protocol Profile  
74 and Agreement (CPP/A [ebCPP]) or the emerging OWL-S [OWL-S].

75  
76 More information on WSDL, CPP/A, and OWL-S are given in the appendices of this  
77 document.

## 78 **3 Service Information Model**

79 The ebXML Registry Service Information Model defines classes in the information  
80 model support registration of service descriptions. A Web service can be represented in  
81 an ebXML Registry through several Registry Information Model [ebRIM] classes:  
82 Service, ServiceBinding, and SpecificationLink. The relationship between these RIM  
83 classes is illustrated in the figure below.



**Figure 1: Relationship between RIM classes Service, ServiceBinding, and SpecificationLink**

84

85

86

87

88 The following sections provide more information on each of the above RIM classes,  
89 specifically:

90

- A definition of the class
- The XML schema representation for the class within a *SubmitObjectsRequest*
- A sample XML instance that conforms to the schema representation

91

92

93

94

95 The reader is referred to the Registry Information Model Specification v3.0 for attributes  
96 and methods associated with each of these classes.

97

98 It should be noted that all namespace declarations are omitted from this document, for  
99 purposes of brevity.

100

## 101 **2.1 Class Service**

102 Service instances describe services, such as Web services.

103

### 104 **2.1.1 Submission XML Schema Representation**

105 The following is the XML schema representation of the Service class within the RIM.xsd  
106 schema [ebRIM Schema].

107

```

108 <element name = "Service" type = "tns:ServiceType"/>
109
110 <complexType name = "ServiceType">
111   <complexContent>
112     <extension base = "tns:RegistryObjectType">
113       <sequence>
114         <element ref = "tns:ServiceBinding" minOccurs = "0"
115           maxOccurs = "unbounded"/>
116       </sequence>
117     </extension>
118   </complexContent>
119 </complexType>
120
121

```

## 122 2.1.2 Sample XML Instance

123 The following sample XML instance illustrates the definition of a Service called  
 124 “AcmePurchaseOrderService” that accepts purchase orders for Acme Corporation. Note  
 125 that the ServiceBinding element is discussed later.

```

126
127 <Service id="urn:acme:services:purchaseorder">
128   <Name>
129     <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>
130   </Name>
131   <Description>
132     <LocalizedString lang="en_US" value = "This Web service will accept purchase orders
133       for Acme Corporation. It will validate the contents of each purchase order, and, if valid,
134       will process the purchase order and automatically generate an Invoice."/>
135   </Description>
136   ...ServiceBinding element is placed here...
137 </Service>

```

## 138 2.2 Class ServiceBinding

139 ServiceBinding instances are RegistryObject instances that represent technical  
 140 information on a specific way to access a Service instance. An example is where a  
 141 ServiceBinding is defined for each protocol that may be used to access the service. A  
 142 Service has a collection of ServiceBindings.

### 143 144 2.2.1 Submission XML Schema Representation

145 The following is the XML schema representation of the ServiceBinding class within the  
 146 RIM.xsd schema.

```

147
148 <element name = "ServiceBinding" type = "tns:ServiceBindingType"/>
149
150 <complexType name = "ServiceBindingType">
151   <complexContent>
152     <extension base = "tns:RegistryObjectType">
153       <sequence>
154         <element ref = "tns:SpecificationLink" minOccurs = "0"
155           maxOccurs = "unbounded"/>
156       </sequence>
157       <attribute name = "service" use="required" type = "tns:referenceURI"/>
158       <attribute name = "accessURI" use="optional" type = "anyURI"/>
159       <attribute name = "targetBinding" use="optional" type = "tns:referenceURI"/>
160     </extension>
161   </complexContent>

```

162 </complexType>

163

## 164 2.2.2 Sample XML Instance

165 The following sample XML instance extends the earlier example by adding a  
 166 ServiceBinding for AcmePurchaseOrderService. The “accessURI” attribute contains the  
 167 address (access point) of the Web service that is being described<sup>1</sup>. Note that the “service”  
 168 attribute refers back to the service that was represented earlier. Note also that the  
 169 SpecificationLink element is discussed later.

170

171 <Service id="urn:acme:services:purchaseorder">

172

<Name>

173

<LocalizedString lang="en\_US" value = "Acme Purchase Order Web Service"/>

174

</Name>

175

<Description>

176

<LocalizedString lang="en\_US" value = "This Web service will accept purchase orders  
 177 for Acme Corporation. It will validate the contents of each purchase order, and, if valid,  
 178 will process the purchase order and automatically generate an Invoice."/>

179

</Description>

180

<ServiceBinding id="urn:acme:services:bindings:purchaseorder"

181

service="AcmePurchaseOrderService"

182

accessURI="http://www.acme.com/purchaseorderservice">

183

....SpecificationLink element is placed here...

184

</ServiceBinding>

185

</Service>

## 186 2.3 Class SpecificationLink

187 A SpecificationLink provides the linkage between a ServiceBinding and one of its  
 188 technical specifications that describes how to use the service with that ServiceBinding.  
 189 For example, a ServiceBinding may have SpecificationLink instances that describe how  
 190 to access the service using a technical specification such as a WSDL document or a  
 191 CORBA IDL document.

192

### 193 2.3.1 Submission XML Schema Representation

194 The following is the XML schema representation of the SpecificationLink class within  
 195 the RIM.xsd schema.

196

197

<element name = "SpecificationLink" type = "tns:SpecificationLinkType"/>

198

<complexType name = "SpecificationLinkType">

199

<complexContent>

200

<extension base = "tns:RegistryObjectType">

201

<sequence minOccurs = "0" maxOccurs = "1">

202

<element ref = "tns:UsageDescription" minOccurs = "0"

203

maxOccurs="1" />

204

<element ref = "tns:UsageParameter" minOccurs = "0"

205

maxOccurs="unbounded" />

206

</sequence>

207

<attribute name = "serviceBinding" use="required" type = "tns:referenceURI"/>

208

<sup>1</sup> It should be noted that with a WSDL SOAP binding, the “location” attribute of the “soap:address” element performs the same function as the “accessURI attribute”. The OASIS/ebXML Registry v3 specifications do not specify the behavior in cases where the two addresses are different (*i.e. which address takes precedence*). This is considered an implementation issue.

```

209         <attribute name = "specificationObject" use="required" type = "tns:referenceURI"/>
210     </extension>
211 </complexContent>
212 </complexType>
213
214 <element name = "UsageDescription" type = "tns:InternationalStringType" />
215 <element name = "UsageParameter" type = "tns:FreeFormText" />
216

```

### 2.3.2 Sample XML Instance

218 The following sample XML instance extends the earlier example by adding a  
 219 SpecificationLink for the ServiceBinding. This SpecificationLink provides a linkage  
 220 between the ServiceBinding and a WSDL document that describes the  
 221 AcmePurchaseOrderService. Note that the “serviceBinding” attribute refers back to the  
 222 ServiceBinding that was represented earlier.

```

223
224 <Service id= "urn:acme:services:purchaseorder">
225     <Name>
226         <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>
227     </Name>
228     <Description>
229         <LocalizedString lang="en_US" value = "This Web service will accept purchase orders
230             for Acme Corporation. It will validate the contents of each purchase order, and, if valid,
231             will process the purchase order and automatically generate an Invoice."/>
232     </Description>
233     <ServiceBinding accessURI="http://www.acme.com/purchaseorderservice">
234         <SpecificationLink serviceBinding="urn:acme:services:bindings:purchaseorder"
235             specificationObject="wsdlForPurchaseOrder">
236             <UsageDescription>
237                 <LocalizedString lang="en_US" value = "This is the WSDL
238                     document that describes the Acme Purchase Order Web Service"/>
239             </UsageDescription>
240         </SpecificationLink>
241     </ServiceBinding>
242 </Service>
243

```

244 The RegistryObject referenced in the “specificationObject” attribute above (the WSDL  
 245 document) would first need to be registered as an ExtrinsicObject. The following is an  
 246 example of how this would be represented within a SubmitObjectsRequest:

```

247
248 <ExtrinsicObject id="urn:acme:services:descriptions:purchaseorder" mimeType="text/xml">
249     <Name>
250         <LocalizedString lang="en_US" value = "The WSDL document for the Acme Purchase Order Web
251             Service"/>
252     </Name>
253 </ExtrinsicObject>

```

## 3 Full SubmitObjectsRequest Example

255 The following is a full SubmitObjectsRequest XML instance example that combines all  
 256 XML instance examples shown above:

```

257 <SubmitObjectsRequest comment="This is the initial submission of the Acme Purchase Order Web
258     Service">
259     <rim: RegistryObjectList>
260
261     <!--Service objects-->
262

```



```

263 <Service id="urn:acme:services:purchaseorder">
264   <Name>
265     <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>
266   </Name>
267   <Description>
268     <LocalizedString lang="en_US" value = "This Web service will accept purchase orders
269       for Acme Corporation. It will validate the contents of each purchase order, and, if valid,
270       will process the purchase order and automatically generate an Invoice."/>
271   </Description>
272   <ServiceBinding id="urn:acme:services:bindings:purchaseorder"
273     service="AcmePurchaseOrderService"
274     accessURI="http://www.acme.com/purchaseorderservice">
275     <SpecificationLink serviceBinding="urn:acme:services:bindings:purchaseorder"
276       specificationObject="wsdlForPurchaseOrder">
277       <UsageDescription>
278         <LocalizedString lang="en_US" value = "This is the WSDL
279           document that describes the Acme Purchase Order Web Service"/>
280       </UsageDescription>
281     </SpecificationLink>
282   </ServiceBinding>
283 </Service>
284
285 <!--WSDL document – ExtrinsicObject -->
286
287 <ExtrinsicObject id="urn:acme:services:descriptions:purchaseorder" mimeType="text/xml">
288   <Name>
289     <LocalizedString lang="en_US" value = "The WSDL document for the Acme Purchase Order Web
290       Service"/>
291   </Name>
292 </ExtrinsicObject>
293
294 </rim: RegistryObjectList>
295 </SubmitObjectsRequest>

```

## 296 4 Extended Scenarios

297 This section includes scenarios that apply various registry features that were not  
 298 described in the earlier examples. Since most of these examples are based on XML  
 299 Schema representations that were already described in previous examples, XML Schema  
 300 representations will not be included in the scenarios below.

### 301 4.1 Versioning of Web Services

302 ebXML Registry contains registry-managed version control features that support  
 303 independent versioning of both RegistryObject metadata as well as repository item  
 304 content. The Registry Information Model defines version attributes for both the  
 305 RegistryObject and ExtrinsicObject classes.

306  
 307 Each RegistryObject instance may have a *versionInfo* attribute, whose value is of type  
 308 VersionInfo. The versionInfo class encapsulates information about the specific version of  
 309 a RegistryObject. It has the following attributes:

- 310 • **versionName:** Defines the version name identifying the VersionInfo for a  
 311 specific RegistryObject version. Automatically generated by the Registry  
 312 implementation.

- 313 • **comment:** Defines the comment associated with the VersionInfo for a  
 314 specific RegistryObject version. Value is indirectly provided by the client  
 315 as a value of the comment attribute of the <rim:Request> object, and is  
 316 automatically set by the Registry implementation if such a value exists.  
 317

318 Each ExtrinsicObject instance may have a *contentVersionInfo* attribute, whose value is  
 319 also of type VersionInfo. The contentversionInfo class provides information about the  
 320 specific version of the RepositoryItem associated with an ExtrinsicObject. It is set by the  
 321 registry.  
 322

### 323 4.1.1 Sample XML Instance

324 The following sample XML instance illustrates a change in a version to an existing  
 325 Service instance, through the submission of a new version of the Service instance and a  
 326 “Supersedes” association reflecting the relationship between the previous version and this  
 327 new version. The registry will automatically assign versioning attributes as described  
 328 above, including copying the comment provided for the SubmitObjectsRequest to the  
 329 RegistryObject.version attribute for the submitted Service:  
 330

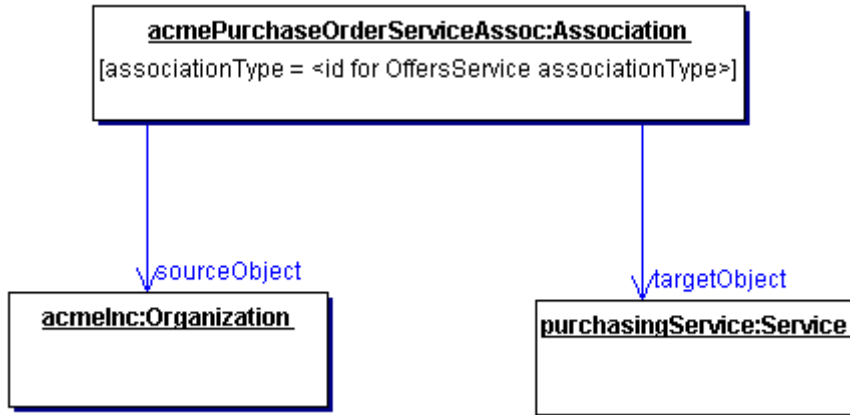
```

331 <SubmitObjectsRequest comment="This is an updated version of the Acme Purchase Order Web
332 Service based on new requirements">
333   <rim:RegistryObjectList>
334     <Service id="urn:acme:services:purchaseorder:v2.0">
335       <Name>
336         <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service – Version
337           2.0"/>
338       </Name>
339       <Description>
340         <LocalizedString lang="en_US" value = "This Web service will accept purchase orders
341           for Acme Corporation. It will validate the contents of each purchase order, and, if valid,
342           will process the purchase order and automatically generate an Invoice."/>
343       </Description>
344     </rim:ObjectRef id = "urn:acme:services:purchaseorder"/>
345     <!--
346     The following association supersedes the current version of the Service instance with the new
347     version that is being submitted.
348     -->
349     <rim:Association id = "New-AcmePurchaseOrderService-Assoc" associationType =
350       "urn:oasis:names:tc:ebxml-regrep:AssociationType:Supersedes" sourceObject =
351       "urn:acme:services:purchaseorder:v2.0" targetObject = "urn:acme:services:purchaseorder"/>
352   </rim:RegistryObjectList>
353 </SubmitObjectsRequest>
    
```

359 In the association above, the “sourceObject” attribute contains the URN of the new  
 360 Service instance, while the “targetObject” attribute contains the URN of the old (version  
 361 1.0) Service instance.  
 362

363 **4.2 Associating a Web Service with an Organization**

364 It is possible to associate a Web service with the Organization that implements the Web  
 365 service. This allows for hierarchical discovery in an ebXML Registry of Organizations  
 366 and their corresponding Web service offerings (or vice-versa).  
 367  
 368



369  
 370 **Figure 2: Associating a Web Service with an Organization**

371  
 372  
 373 **4.2.1 Sample XML Instance**

374 The following sample XML instance associates Acme Corporation with its Purchase  
 375 Order Service through an “OffersService” association. It is assumed that an Organization  
 376 instance already exists for Acme Corporation, and the Purchase Order Service and any  
 377 associated instances, such as ServiceBinding and SpecificationLink, have been registered  
 378 as well.  
 379

```

    380 <SubmitObjectsRequest>
    381   <rim:RegistryObjectList>
    382     <!--
    383       The following association denotes that Acme Corporation offers a Purchase Order Service. The
    384       sourceObject is the URN of Acme Corporation’s Organization instance, while the targetObject is
    385       the URN of the Purchase Order Service’s Service instance.
    386     -->
    387     <rim:Association id = "AcmePurchaseOrderService-Assoc" associationType =
    388       "urn:uuid_for_OffersService_association" sourceObject = "urn:uuid:a2345678-1234-1234-
    389       3345678901292" targetObject = "urn:uuid:a2345678-1234-1234-93456789012"/>
    390     <rim:Association id = "AcmePurchaseOrderService-Assoc" associationType =
    391       "urn:oasis:names:tc:ebxml-regrep:AssociationType:OffersService" sourceObject =
    392       "urn:acme:organization" targetObject = "urn:acme:services:purchaseorder"/>
    393   </rim:RegistryObjectList>
    394 </SubmitObjectsRequest>
    395
    396
    397
    398
    399
  
```

400 In the association above, the “sourceObject” attribute contains the URN of Acme  
 401 Corporation’s Organization instance, while the “targetObject” attribute contains the URN  
 402 of the Purchase Order Service’s Service instance.

403

404 [UPDATES END HERE]

405 **4.3 Associating a Web Service with an Access Control Policy**

406 It is possible to associate a Web service with an Access Control Policy in order to  
 407 authorize access to methods associated with the Service instance. This can help ensure  
 408 that only authorized users can (for example) perform life cycle operations on the Service  
 409 instance.

410

411 **4.3.1 Sample XML Instance**

412 The following sample XML instance associates Acme Corporation's Purchase Order  
 413 Service with an Access Control Policy through an "AccessControlPolicyFor" association.  
 414 It is assumed that an AccessControlPolicy instance already exists for the Access Control  
 415 Policy, and the Purchase Order Service and any associated instances, such as  
 416 ServiceBinding and SpecificationLink, have been registered as well.

417

418 &lt;SubmitObjectsRequest&gt;

419 &lt;rim:LeafRegistryObjectList&gt;

420

421 &lt;!--

422 The following association relates an existing Access Control Policy to Acme Corporation's  
 423 Purchase Order Service. The sourceObject is the UUID of Acme Corporation's Purchase Order  
 424 Service instance, while the targetObject is the UUID of the Access Control Policy instance.

425

426 --&gt;

427 <rim:Association id = "AcmePurchaseOrderService-AccessPolicyAssoc" associationType =  
 428 "urn:uuid\_for\_AccessControlPolicyFor\_association" sourceObject = "urn:uuid:a2345678-1234-  
 429 1234-8345678901262" targetObject = "urn:uuid:a2345678-1234-1234-03456789015"/>

430

431 &lt;/rim:LeafRegistryObjectList&gt;

432 &lt;/SubmitObjectsRequest&gt;

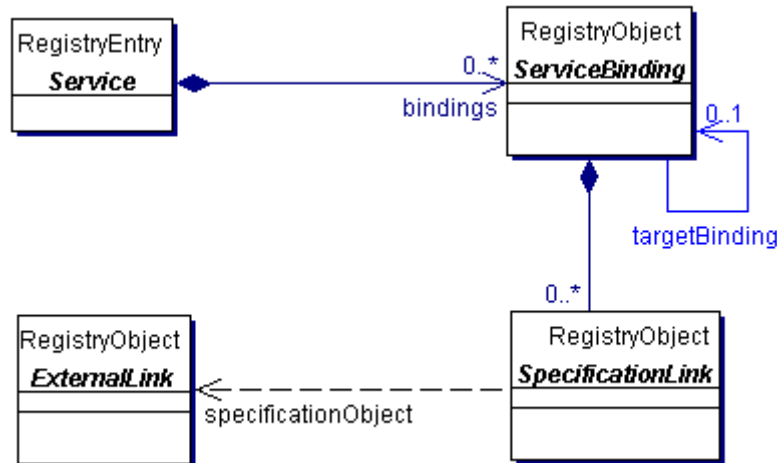
433

434 In the association above, the "sourceObject" attribute contains the UUID of Acme  
 435 Corporation's Purchase Order Service instance, while the "targetObject" attribute  
 436 contains the UUID of the Access Control Policy instance.

437 **4.4 Registering a Service Description that is External to the**  
438 **Registry**

439 It is possible to associate a Web service with a Service description that is external to the  
 440 registry by using the SpecificationLink class as shown below.

441



442  
443 **Figure 3: Registering an External Service Description**

444  
445 **4.4.1 Sample XML Instance**

446 The following sample XML instance is similar to that of Section 3.3.2 above, with the  
447 only difference being that the “specificationObject” attribute contains the URL of the  
448 external Service description.

```

449
450 <Service id="AcmePurchaseOrderService">
451   <Name>
452     <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>
453   </Name>
454   <Description>
455     <LocalizedString lang="en_US" value = "This Web service will accept purchase orders
456       for Acme Corporation. It will validate the contents of each purchase order, and, if valid,
457       will process the purchase order and automatically generate an Invoice."/>
458   </Description>
459   <ServiceBinding accessURI="http://www.acme.com/purchaseorderservice">
460     <SpecificationLink specificationObject="urn:uuid_for_ExternalLink_instance">
461       <UsageDescription>
462         <LocalizedString lang="en_US" value = "This is the WSDL
463           document that describes the Acme Purchase Order Web Service"/>
464       </UsageDescription>
465     </SpecificationLink>
466   </ServiceBinding>
467 </Service>

```

468  
469 The “specificationObject” attribute above references an ExternalLink instance that  
470 contains the URI for the WSDL document.

471 **4.5 Web Service Redirection**

472 The “targetBinding” attribute of the ServiceBinding class is used to redirect a Web  
473 service to another access point. This may be done, for example, if the service is rehoused  
474 by another service provider. If the “targetBinding” attribute is specified in a  
475 ServiceBinding instance, the “accessURI” attribute is ignored.

#### 477 **4.5.1 Sample XML Instance**

478 The following sample XML instance is similar to the XML instance in Section 3.2.2  
479 above, with the exception that the “targetBinding” attribute has been added:

```
480
481 <Service id="AcmePurchaseOrderService">
482   <Name>
483     <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>
484   </Name>
485   <Description>
486     <LocalizedString lang="en_US" value = "This Web service will accept purchase orders
487       for Acme Corporation. It will validate the contents of each purchase order, and, if valid,
488       will process the purchase order and automatically generate an Invoice."/>
489   </Description>
490   <ServiceBinding accessURI="http://www.acme.com/purchaseorderservice"
491     targetBinding=" urn:uuid_for_ExternalLink_instance">
492     ....SpecificationLink element goes here...
493   </ServiceBinding>
494 </Service>
```

495

496 In the above example, Acme Corporation’s Purchase Order Service has been rehosted to  
497 a URI that is specified in the ExternalLink instance referenced by the “targetBinding”  
498 attribute above.

#### 499 **4.6 Customizing Metadata Using Slots**

500 The Slot class provides a dynamic way to add arbitrary attributes to RegistryObject  
501 instances through the specification of name/value pairs. This ability to add attributes  
502 dynamically to RegistryObject instances enables extensibility within the Registry  
503 Information Model. Slots can be used with Web Service definitions to define information  
504 that is unique to an organization’s needs.

505

##### 506 **4.6.1 Sample XML Instance**

507 The following sample XML instance extends the example in Section 3.2.2 by adding  
508 slots for the internal Web Service Administrator Name and whether the Web service is  
509 HTTP(REST)-based or SOAP-based<sup>2</sup>:

510

```
511 <Service id="AcmePurchaseOrderService">
512   <Name>
513     <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>
514   </Name>
515   <Description>
516     <LocalizedString lang="en_US" value = "This Web service will accept purchase orders
517       for Acme Corporation. It will validate the contents of each purchase order, and, if valid,
518       will process the purchase order and automatically generate an Invoice."/>
519   </Description>
520   <Slot name = 'Web Service Administrator Name'>
521     <ValueList>
522       <Value>John Smith</Value>
523     </ValueList>
```

---

<sup>2</sup> Although this information can be obtained by inspecting a WSDL document, it can be more efficient to specify it at this metadata level so as to avoid the need to automatically open and inspect a WSDL document.

```
524     </Slot >
525     <Slot name = 'HTTP or SOAP'>
526         <ValueList>
527             <Value>SOAP</Value>
528         </ValueList>
529     </Slot >
530     <ServiceBinding accessURI="http://www.acme.com/purchaseorderservice">
531         ....SpecificationLink element goes here...
532     </ServiceBinding>
533 </Service>
```

## 534 **Appendix A WSDL Introduction**

535 The Web Service Description Language (WSDL) provides the ability to describe a Web  
536 service in abstract as well as with concrete bindings to specific protocols. In WSDL, an  
537 abstract service consists of one or more *port types* or end-points. Each port type consists  
538 of a collection of *operations*. Each operation is defined in terms of *messages* that define  
539 what data is exchanged as part of that operation. Each message is typically defined in  
540 terms of elements within an XML Schema definition. An abstract service is not bound to  
541 any specific protocol (e.g. SOAP). In WSDL, an abstract service may be used to define a  
542 concrete service by binding it to a specific protocol. This binding is done by providing a  
543 *binding* definition for each abstract port type that defines additional protocols specific  
544 details. Finally, a concrete *service* definition is defined as a collection of *ports*, where  
545 each port simply adds address information such as a URL for each concrete port.

546  
547 One of the most distinctive features of WSDL is that the abstract information can be  
548 separated from the concrete information, to form an abstract *service interface definition*  
549 and one or more concrete *service implementation definitions*. This separation allows for  
550 the creation of clearer service definitions by separating the definitions according to their  
551 level of abstraction. It also maximizes the ability to reuse service definitions of all kinds.  
552 As a result, WSDL documents structured in this way are easier to use and maintain  
553 [UDDI].  
554

## 555 **Appendix B OASIS/ebXML Collaboration-Protocol 556 Profile and Agreement (CPP/A) Introduction**

557 The OASIS/ebXML Collaboration-Protocol Profile and Agreement (CPP/A)  
558 specification defines the structure and contents of ebXML Collaboration Protocol Profiles  
559 (CPPs) and Collaboration Protocol Agreements (CPAs), both of which are used for  
560 business integration and trading partner discovery purposes. A CPP describes the  
561 message-exchange capabilities of a Party, while a CPA defines the capabilities that two  
562 Parties need to agree upon to enable them to engage in electronic business for the  
563 purposes of the particular CPA. A CPA may be created by computing the intersection of  
564 the two Partners' CPPs.

565  
566 Included in the CPP and CPA are details of transport, messaging, security constraints,  
567 and bindings to a Business Process Specification document (which may conform to the  
Registering Web Services in an ebXML Registry

568 ebXML Business Process Specification Schema, or BPSS) that contains the definition of  
569 the interactions between the two Parties while engaging in a specified electronic Business  
570 Collaboration. A Business Process Specification document, CPP, and CPA may all be  
571 stored in an ebXML Registry.

## 572 **Appendix C DAML-S Introduction**

573 DAML-S is an emerging DAML+OIL ontology for Semantic Web Services. It is a  
574 collaborative effort between BBN Technologies, Carnegie Mellon University, Nokia  
575 Research Center, SRI International, Stanford University, and Yale University. The  
576 Semantic Web is rapidly becoming a reality through the development of Semantic Web  
577 markup languages such as DAML+OIL, and these markup languages enable the creation  
578 of arbitrary domain ontologies (such as DAML-S) that support the unambiguous  
579 description of Web content.

580  
581 While WSDL provides a low-level description of Web services, DAML-S complements  
582 WSDL by providing Web service descriptions at the application layer – that is, describing  
583 *what* a service can do, not just *how* it does it. A DAML-S/WSDL binding (known as a  
584 “grounding”) has been defined that involves a complementary use of the two languages.  
585

## 586 **References**

- 587 [DAML-S] DAML-S: Web Service Descriptions for the Semantic Web  
588 <http://xml.coverpages.org/ISWC2002-DAMLS>  
589
- 590 [ebCPP] ebXML Collaboration-Protocol Profile and Agreement Specification  
591 <http://www.oasis-open.org/committees/ebxml-cppa/documents/ebcpp-2.0.pdf>  
592
- 593 [ebRIM] ebXML Registry Information Model Specification v3.0 (release pending)
- 594 [ebRIM Schema] ebXML Registry Information Model Schema v3.0  
595 <http://www.oasis-open.org/committees/regrep/documents/3.0/schema/rim.xsd>
- 596 [ebRS] ebXML Registry Services Specification v3.0 (release pending)
- 597 UDDI] Using WSDL in a UDDI Registry, Version 1.8  
598 <http://www.oasis-open.org/committees/uddi-spec/doc/bp/uddi-spec-tc-bp-using-wsdl-v108-20021110.htm>  
599
- 600 [WSA] W3C Web Services Activity  
601 <http://www.w3.org/2002/ws/>
- 602 [WSDL] Web Services Description Language (WSDL)  
603 <http://www.w3.org/TR/2002/WD-wsdl12-20020709/>



604