



Creating A Single Global Electronic Market

1

2

3 **Technical Note**

4 **Registering Web Services in an ebXML Registry, Version** 5 **1.0**

6 **12 March 2003**

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, release pending

18

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

19

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

21

22 **Status of this Document**

23

24 This document is an OASIS Registry Technical Committee Technical Note Working
25 Draft. Distribution of this document is unlimited.

26

27

28

28 **Table of Contents**

29 **ABSTRACT 1**

30 **STATUS OF THIS DOCUMENT 1**

31 **1 INTRODUCTION 4**

32 **1 INTRODUCTION 4**

33 **2 WEB SERVICES 4**

34 **3 RELEVANT EBXML REGISTRY CLASSES 4**

35 **3.1 Class Service 5**

36 **3.2 Class ServiceBinding 6**

37 **3.3 Class SpecificationLink 7**

38 **4 FULL SUBMITOBJECTSREQUEST EXAMPLE 8**

39 **5 EXTENDED SCENARIOS 9**

40 **5.1 Versioning of Web Services 9**

41 **5.2 Associating a Web Service with an Organization 10**

42 **5.3 Associating a Web Service with an Access Control Policy 11**

43 **5.4 Registering a Service Description that is External to the Registry 12**

44 **5.5 Web Service Redirection 13**

45 **5.6 Customizing Metadata Using Slots 13**

46 **APPENDIX A WSDL INTRODUCTION 14**

47 **APPENDIX B OASIS/EBXML COLLABORATION-PROTOCOL PROFILE**

48 **AND AGREEMENT (CPP/A) INTRODUCTION 15**

49 **APPENDIX C DAML-S INTRODUCTION 15**

50

51 **Figures**

52 **FIGURE 1: RELATIONSHIP BETWEEN RIM CLASSES SERVICE,**
53 **SERVICEBINDING, AND SPECIFICATIONLINK..... 5**

54 **FIGURE 2: ASSOCIATING A WEB SERVICE WITH AN ORGANIZATION..... 10**

55 **FIGURE 3: REGISTERING AN EXTERNAL SERVICE DESCRIPTION 12**
56

56 **1 Introduction**

57 An ebXML Registry provides a stable store where information submitted by a Submitting
58 Organization is made persistent. Such information is used to facilitate ebXML-based
59 Business to Business (B2B) partnerships and transactions. Submitted content may be
60 XML schema and documents, process descriptions, Web services, ebXML Core
61 Components, context descriptions, UML models, information about parties and even
62 software components.

63
64 The purpose of this document is to provide a Best Practice for registering Web services
65 and their associated entities in an ebXML Registry.

66 **2 Web Services**

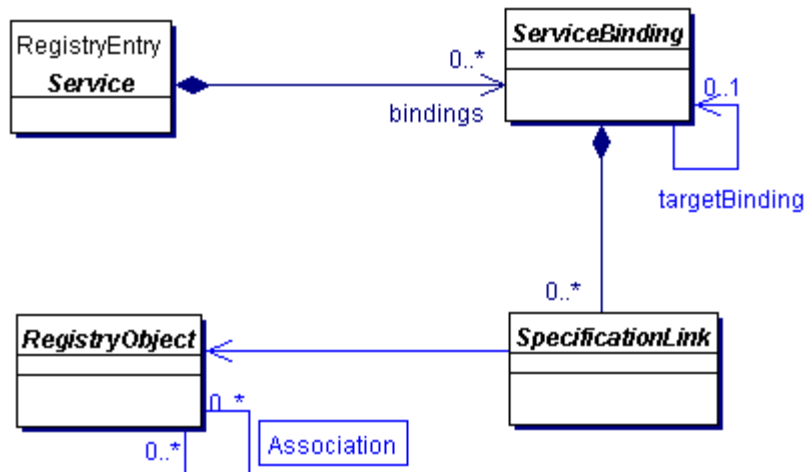
67 A Web service is “a software system identified by a URI whose public interfaces and
68 bindings are defined and described using XML. Its definition can be discovered by other
69 software systems. These systems may then interact with the Web service in a manner
70 described by its definition, using XML based messages conveyed by Internet protocols.”
71 [WSA]. The characteristics, capabilities, and requirements of a Web service can be
72 described and published, thereby enabling automatic discovery and invocation of the
73 service. One mechanism by which these descriptions can be published is in an ebXML
74 Registry.

75
76 The most common mechanism for describing Web services today is the Web Services
77 Description Language, or WSDL [WSDL]; however, the Service description that is
78 registered can be in any format such as OASIS/ebXML Collaboration-Protocol Profile
79 and Agreement (CPP/A [ebCPP]) or the emerging DAML-S [DAML-S].

80
81 More information on WSDL, CPP/A, and DAML-S are given in the appendices of this
82 document.

83 **3 Relevant ebXML Registry Classes**

84 A Web service can be represented in an ebXML Registry through several Registry
85 Information Model [ebRIM] classes: Service, ServiceBinding, and SpecificationLink.
86 The relationship between these RIM classes is illustrated in the figure below.



87
88 **Figure 1: Relationship between RIM classes Service, ServiceBinding, and**
89 **SpecificationLink**

90
91 The following sections provide more information on each of the above RIM classes,
92 specifically:

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

97
98 The reader is referred to the Registry Information Model Specification v3.0 for attributes
99 and methods associated with each of these classes.

100
101 It should be noted that all namespace declarations are omitted from this document, for
102 purposes of brevity.

104 3.1 Class Service

105 Service instances are RegistryEntry instances that provide information on services (e.g.,
106 Web services).

108 3.1.1 Submission XML Schema Representation

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

```

112 <element name = "Service" type = "tns:ServiceType"/>
113
114 <complexType name = "ServiceType">
115   <complexContent>
116     <extension base = "tns:RegistryEntryType">
117       <sequence>
118         <element ref = "tns:ServiceBinding" minOccurs = "0"
119           maxOccurs = "unbounded"/>
120       </sequence>
121     </extension>
122   </complexContent>
123 </complexType>
124
125

```

126 3.1.2 Sample XML Instance

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

```

130
131 <Service id="AcmePurchaseOrderService">
132   <Name>
133     <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>
134   </Name>
135   <Description>
136     <LocalizedString lang="en_US" value = "This Web service will accept purchase orders
137       for Acme Corporation. It will validate the contents of each purchase order, and, if valid,
138       will process the purchase order and automatically generate an Invoice."/>
139   </Description>
140   ...ServiceBinding element is placed here...
141 </Service>

```

142 3.2 Class ServiceBinding

143 ServiceBinding instances are RegistryObject instances that represent technical
 144 information on a specific way to access a specific interface offered by a Service instance.
 145 A Service has a collection of ServiceBindings.

147 3.2.1 Submission XML Schema Representation

148 The following is the XML schema representation of the ServiceBinding class within the
 149 RIM.xsd schema.

```

150
151 <element name = "ServiceBinding" type = "tns:ServiceBindingType"/>
152
153 <complexType name = "ServiceBindingType">
154   <complexContent>
155     <extension base = "tns:RegistryObjectType">
156       <sequence>
157         <element ref = "tns:SpecificationLink" minOccurs = "0" maxOccurs =
158           "unbounded"/>
159       </sequence>
160       <attribute name = "accessURI" use="optional" type = "anyURI"/>
161       <attribute name = "targetBinding" use="optional" type = "anyURI"/>
162     </extension>
163   </complexContent>
164 </complexType>
165

```

166 3.2.2 Sample XML Instance

167 The following sample XML instance extends the earlier example by adding a
 168 ServiceBinding for AcmePurchaseOrderService. The “accessURI” attribute contains the
 169 address (access point) of the Web service that is being described². Note that the
 170 SpecificationLink element is discussed later.

```

171
172 <Service id="AcmePurchaseOrderService">
173   <Name>
174     <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>
175   </Name>
176   <Description>
177     <LocalizedString lang="en_US" value = "This Web service will accept purchase orders
178     for Acme Corporation. It will validate the contents of each purchase order, and, if valid,
179     will process the purchase order and automatically generate an Invoice."/>
180   </Description>
181   <ServiceBinding accessURI="http://www.acme.com/purchaseorderservice">
182     ....SpecificationLink element is placed here...
183   </ServiceBinding>
184 </Service>

```

185 3.3 Class SpecificationLink

186 A SpecificationLink provides the linkage between a ServiceBinding and one of its
 187 technical specifications that describes how to use the service with that ServiceBinding.
 188 For example, a ServiceBinding may have a specific SpecificationLink instance that
 189 describes how to access the service using a technical specification (such as a WSDL
 190 document).

191

192 3.3.1 Submission XML Schema Representation

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

```

195
196 <element name = "SpecificationLink" type = "tns:SpecificationLinkType"/>
197
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 = "specificationObject" use="required" type = "anyURI"/>
208     </extension>
209   </complexContent>
210 </complexType>
211
212 <element name = "UsageDescription" type = "tns:InternationalStringType" />
213 <element name = "UsageParameter" type = "tns:FreeFormText" />

```

² 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.

214
215

3.3.2 Sample XML Instance

216 The following sample XML instance extends the earlier example by adding a
217 SpecificationLink for the ServiceBinding. This SpecificationLink provides a linkage
218 between the ServiceBinding and a WSDL document that describes the
219 AcmePurchaseOrderService:

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

240 The RegistryObject referenced in the “specificationObject” attribute above (the WSDL
241 document) would first need to be registered as an ExtrinsicObject in a request such as the
242 following:

```
243 <ExtrinsicObject id="wsdlForPurchaseOrder" mimeType="text/xml">
244   <Name>
245     <LocalizedString lang="en_US" value = "The WSDL document for the Acme Purchase Order Web
246     Service"/>
247   </Name>
248 </ExtrinsicObject>
```

250 4 Full SubmitObjectsRequest Example

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

```
253 <SubmitObjectsRequest>
254   <rim:LeafRegistryObjectList>
255     <!--Service objects-->
256     <Service id="AcmePurchaseOrderService">
257       <Name>
258         <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>
259       </Name>
260       <Description>
261         <LocalizedString lang="en_US" value = "This Web service will accept purchase orders
262         for Acme Corporation. It will validate the contents of each purchase order, and, if valid,
263         will process the purchase order and automatically generate an Invoice."/>
264       </Description>
265       <ServiceBinding accessURI="http://www.acme.com/purchaseorderservice">
```

Registering Web Services in an ebXML Registry


```

268         <SpecificationLink specificationObject="wsdlForPurchaseOrder">
269             <UsageDescription>
270                 <LocalizedString lang="en_US" value = "This is the WSDL
271                     document that describes the Acme Purchase Order Web Service"/>
272             </UsageDescription>
273         </SpecificationLink>
274     </ServiceBinding>
275 </Service>
276
277 <!--WSDL document – ExtrinsicObject -->
278 <ExtrinsicObject id="wsdlForPurchaseOrder" mimeType="text/xml">
279     <Name>
280         <LocalizedString lang="en_US" value = "The WSDL document for the Acme Purchase Order
281             Web Service"/>
282     </Name>
283 </ExtrinsicObject>
284 </rim:LeafRegistryObjectList>
285 </SubmitObjectsRequest>
286
287

```

288 5 Extended Scenarios

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

293 5.1 Versioning of Web Services

294 The Registry Information Model defines several version attributes for the RegistryEntry
 295 class. These are:

- 296 • **majorVersion:** A registry-assigned major revision number; may be
 297 updated by the registry when an object is updated
- 298 • **minorVersion:** A registry-assigned minor revision number; may be
 299 updated by the registry when an object is updated
- 300 • **userVersion:** A user-assigned revision number, similar to the
 301 majorVersion-minorVersion tuple
 302

303 Only the “userVersion” attribute will be referenced in the example in Section 5.1.1
 304 below, as it is the only user-assigned version attribute.

306 5.1.1 Sample XML Instance

307 The following sample XML instance illustrates a change in a user-assigned version to an
 308 existing Service instance, through the submission of a new version of the Service instance
 309 and a “Supersedes” association reflecting the relationship between the previous version
 310 and this new version. It assumes that the user-assigned version is being updated from 1.0
 311 to 2.0.

```

312 <SubmitObjectsRequest>
313     <rim:LeafRegistryObjectList>
314
315

```

```

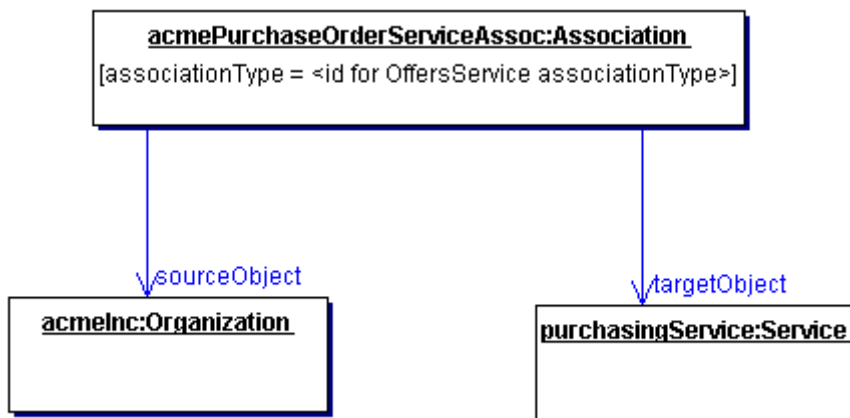
316     <Service id="AcmePurchaseOrderService2" userVersion="2.0">
317       <Name>
318         <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>
319       </Name>
320       <Description>
321         <LocalizedString lang="en_US" value = "This Web service will accept purchase orders
322         for Acme Corporation. It will validate the contents of each purchase order, and, if valid,
323         will process the purchase order and automatically generate an Invoice."/>
324       </Description>
325     </Service>
326
327     <rim:ObjectRef id = "urn:uuid:a2345678-1234-1234-234567890129"/>
328
329     <!--
330     The following association supersedes the current version of the Service instance with the new
331     version that is being submitted.
332     -->
333
334     <rim:Association id = "New-AcmePurchaseOrderService-Assoc" associationType =
335     "urn:uuid_for_Supersedes_association" sourceObject = "AcmePurchaseOrderService2"
336     targetObject = "urn:uuid:a2345678-1234-1234- 234567890129"/>
337
338   </rim:LeafRegistryObjectList>
339 </SubmitObjectsRequest>

```

341 In the association above, the “sourceObject” attribute contains the UUID of the new
 342 Service instance, while the “targetObject” attribute contains the UUID of the old (version
 343 1.0) Service instance.

344 5.2 Associating a Web Service with an Organization

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



350
 351 **Figure 2: Associating a Web Service with an Organization**

352
 353

354 **5.2.1 Sample XML Instance**

355 The following sample XML instance associates Acme Corporation with its Purchase
 356 Order Service through an “OffersService” association. It is assumed that an Organization
 357 instance already exists for Acme Corporation, and the Purchase Order Service and any
 358 associated instances, such as ServiceBinding and SpecificationLink, have been registered
 359 as well.

```

360 <SubmitObjectsRequest>
361   <rim:LeafRegistryObjectList>
362     <!--
363       The following association denotes that Acme Corporation offers a Purchase Order Service. The
364       sourceObject is the UUID of Acme Corporation's Organization instance, while the targetObject is
365       the UUID of the Purchase Order Service's Service instance.
366     -->
367     <!--
368       -->
369     <rim:Association id = "AcmePurchaseOrderService-Assoc" associationType =
370       "urn:uuid_for_OffersService_association" sourceObject = "urn:uuid:a2345678-1234-1234-
371       3345678901292" targetObject = "urn:uuid:a2345678-1234-1234-93456789012"/>
372   </rim:LeafRegistryObjectList>
373 </SubmitObjectsRequest>
    
```

374 In the association above, the “sourceObject” attribute contains the UUID of Acme
 375 Corporation’s Organization instance, while the “targetObject” attribute contains the
 376 UUID of the Purchase Order Service’s Service instance.

380 **5.3 Associating a Web Service with an Access Control Policy**

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

385 **5.3.1 Sample XML Instance**

387 The following sample XML instance associates Acme Corporation’s Purchase Order
 388 Service with an Access Control Policy through an “AccessControlPolicyFor” association.
 389 It is assumed that an AccessControlPolicy instance already exists for the Access Control
 390 Policy, and the Purchase Order Service and any associated instances, such as
 391 ServiceBinding and SpecificationLink, have been registered as well.

```

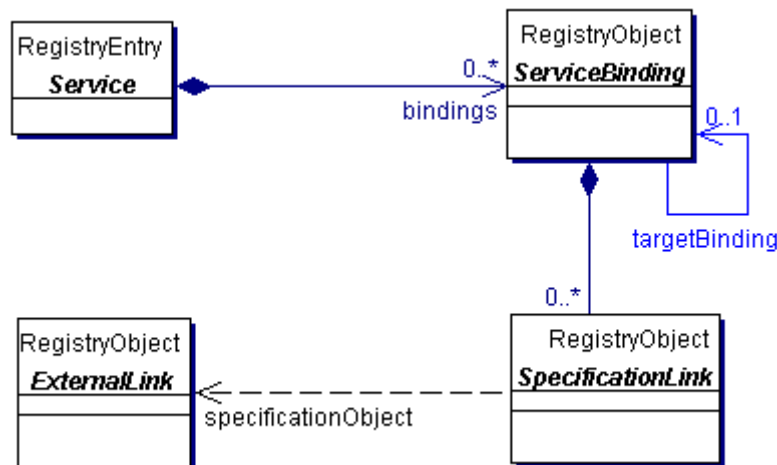
392 <SubmitObjectsRequest>
393   <rim:LeafRegistryObjectList>
394     <!--
395       The following association relates an existing Access Control Policy to Acme Corporation's
396       Purchase Order Service. The sourceObject is the UUID of Acme Corporation's Purchase Order
397       Service instance, while the targetObject is the UUID of the Access Control Policy instance.
398     -->
399     <!--
400       -->
401     <rim:Association id = "AcmePurchaseOrderService-AccessPolicyAssoc" associationType =
402       "urn:uuid_for_AccessControlPolicyFor_association" sourceObject = "urn:uuid:a2345678-1234-
403       1234-8345678901262" targetObject = "urn:uuid:a2345678-1234-1234-03456789015"/>
404   </rim:LeafRegistryObjectList>
405 </SubmitObjectsRequest>
    
```

408

409 In the association above, the “sourceObject” attribute contains the UUID of Acme
 410 Corporation’s Purchase Order Service instance, while the “targetObject” attribute
 411 contains the UUID of the Access Control Policy instance.

412 **5.4 Registering a Service Description that is External to the**
 413 **Registry**

414 It is possible to associate a Web service with a Service description that is external to the
 415 registry by using the SpecificationLink class as shown below.
 416



417

418 **Figure 3: Registering an External Service Description**

419

420 **5.4.1 Sample XML Instance**

421 The following sample XML instance is similar to that of Section 3.3.2 above, with the
 422 only difference being that the “specificationObject” attribute contains the URL of the
 423 external Service description.
 424

```

425 <Service id="AcmePurchaseOrderService">
426   <Name>
427     <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>
428   </Name>
429   <Description>
430     <LocalizedString lang="en_US" value = "This Web service will accept purchase orders
431     for Acme Corporation. It will validate the contents of each purchase order, and, if valid,
432     will process the purchase order and automatically generate an Invoice."/>
433   </Description>
434   <ServiceBinding accessURI="http://www.acme.com/purchaseorderservice">
435     <SpecificationLink specificationObject="urn:uuid_for_ExternalLink_instance">
436       <UsageDescription>
437         <LocalizedString lang="en_US" value = "This is the WSDL
438         document that describes the Acme Purchase Order Web Service"/>
439       </UsageDescription>
440     </SpecificationLink>
441   </ServiceBinding>
442 </Service>
443
    
```

444 The “specificationObject” attribute above references an ExternalLink instance that
445 contains the URI for the WSDL document.

446 **5.5 Web Service Redirection**

447 The “targetBinding” attribute of the ServiceBinding class is used to redirect a Web
448 service to another access point. This may be done, for example, if the service is rehosted
449 by another service provider. If the “targetBinding” attribute is specified in a
450 ServiceBinding instance, the “accessURI” attribute is ignored.
451

452 **5.5.1 Sample XML Instance**

453 The following sample XML instance is similar to the XML instance in Section 3.2.2
454 above, with the exception that the “targetBinding” attribute has been added:
455

```
456 <Service id="AcmePurchaseOrderService">  
457   <Name>  
458     <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>  
459   </Name>  
460   <Description>  
461     <LocalizedString lang="en_US" value = "This Web service will accept purchase orders  
462       for Acme Corporation. It will validate the contents of each purchase order, and, if valid,  
463       will process the purchase order and automatically generate an Invoice."/>  
464   </Description>  
465   <ServiceBinding accessURI="http://www.acme.com/purchaseorderservice"  
466     targetBinding=" urn:uuid_for_ExternalLink_instance">  
467     ....SpecificationLink element goes here...  
468   </ServiceBinding>  
469 </Service>
```

470
471 In the above example, Acme Corporation’s Purchase Order Service has been rehosted to
472 a URI that is specified in the ExternalLink instance referenced by the “targetBinding”
473 attribute above.

474 **5.6 Customizing Metadata Using Slots**

475 The Slot class provides a dynamic way to add arbitrary attributes to RegistryObject
476 instances through the specification of name/value pairs. This ability to add attributes
477 dynamically to RegistryObject instances enables extensibility within the Registry
478 Information Model. Slots can be used with Web Service definitions to define information
479 that is unique to an organization’s needs.
480

481 **5.6.1 Sample XML Instance**

482 The following sample XML instance extends the example in Section 3.2.2 by adding
483 slots for the internal Web Service Administrator Name and whether the Web service is
484 HTTP(REST)-based or SOAP-based³:
485

³ 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.

```
486 <Service id="AcmePurchaseOrderService">
487   <Name>
488     <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>
489   </Name>
490   <Description>
491     <LocalizedString lang="en_US" value = "This Web service will accept purchase orders
492     for Acme Corporation. It will validate the contents of each purchase order, and, if valid,
493     will process the purchase order and automatically generate an Invoice."/>
494   </Description>
495   <Slot name = 'Web Service Administrator Name'>
496     <ValueList>
497       <Value>John Smith</Value>
498     </ValueList>
499   </Slot >
500   <Slot name = 'HTTP or SOAP'>
501     <ValueList>
502       <Value>SOAP</Value>
503     </ValueList>
504   </Slot >
505   <ServiceBinding accessURI="http://www.acme.com/purchaseorderservice">
506     ....SpecificationLink element goes here...
507   </ServiceBinding>
508 </Service>
```

509 **Appendix A WSDL Introduction**

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

521
522 One of the most distinctive features of WSDL is that the abstract information can be
523 separated from the concrete information, to form an abstract *service interface definition*
524 and one or more concrete *service implementation definitions*. This separation allows for
525 the creation of clearer service definitions by separating the definitions according to their
526 level of abstraction. It also maximizes the ability to reuse service definitions of all kinds.
527 As a result, WSDL documents structured in this way are easier to use and maintain
528 [UDDI].
529

530 **Appendix B OASIS/ebXML Collaboration-Protocol** 531 **Profile and Agreement (CPP/A) Introduction**

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

540
541 Included in the CPP and CPA are details of transport, messaging, security constraints,
542 and bindings to a Business Process Specification document (which may conform to the
543 ebXML Business Process Specification Schema, or BPSS) that contains the definition of
544 the interactions between the two Parties while engaging in a specified electronic Business
545 Collaboration. A Business Process Specification document, CPP, and CPA may all be
546 stored in an ebXML Registry.

547 **Appendix C DAML-S Introduction**

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

555
556 While WSDL provides a low-level description of Web services, DAML-S complements
557 WSDL by providing Web service descriptions at the application layer – that is, describing
558 *what* a service can do, not just *how* it does it. A DAML-S/WSDL binding (known as a
559 “grounding”) has been defined that involves a complementary use of the two languages.
560

561 **References**

562 [DAML-S] DAML-S: Web Service Descriptions for the Semantic Web

563 <http://xml.coverpages.org/ISWC2002-DAMLS>

564

565 [ebCPP] ebXML Collaboration-Protocol Profile and Agreement Specification

566 <http://www.oasis-open.org/committees/ebxml-cppa/documents/ebcpp-2.0.pdf>

567

568 [ebRIM] ebXML Registry Information Model Specification v3.0 (release pending)

- 569 [ebRIM Schema] ebXML Registry Information Model Schema v3.0
570 <http://www.oasis-open.org/committees/regrep/documents/3.0/schema/rim.xsd>
- 571 [ebRS] ebXML Registry Services Specification v3.0 (release pending)
572
- 581 UDDI] Using WSDL in a UDDI Registry, Version 1.8
582 <http://www.oasis-open.org/committees/uddi-spec/doc/bp/uddi-spec-tc-bp-using-wsdl-v108-20021110.htm>
583
- 584 [WSA] W3C Web Services Activity
585 <http://www.w3.org/2002/ws/>
- 586 [WSDL] Web Services Description Language (WSDL)
587 <http://www.w3.org/TR/2002/WD-wsdl12-20020709/>
588