1

2

Technical Note

Registering Web Services in an ebXML Registry, Version 1.0

12 March 2003

Authors

Joseph M. Chiusano, Booz Allen Hamilton
Farrukh Najmi, Sun Microsystems

Abstract

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

OASIS/ebXML Registry Information Model (ebRIM) v3.0, release pending
OASIS/ebXML Registry Services Specification (ebRS), v3.0, release pending

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

Status of this Document

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

28 # TABLE OF CONTENTS

51 **Figures**

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.

---

[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.

Registering Web Services in an ebXML Registry   Page 4

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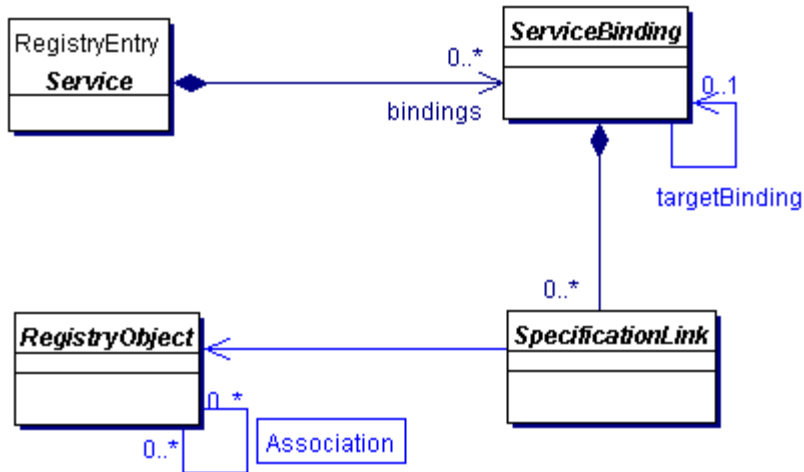
Figure 1: Relationship between RIM classes Service, ServiceBinding, and
SpecificationLink

90

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

- 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

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

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

3.1 Class Service

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

3.1.1 Submission XML Schema Representation

The following is the XML schema representation of the Service class within the RIM.xsd
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
```

3.1.2 Sample XML Instance

126

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

127
128
129
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>
```

3.2 Class ServiceBinding

142

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

143
144
145
146

3.2.1 Submission XML Schema Representation

147

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

148
149
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 described[2]. 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" />
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
```
221  <Service id="AcmePurchaseOrderService">
222      <Name>
223          <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>
224      </Name>
225      <Description>
226          <LocalizedString lang="en_US" value = "This Web service will accept purchase orders
227          for Acme Corporation.  It will validate the contents of each purchase order, and, if valid,
228           will process the purchase order and automatically generate an Invoice."/>
229      </Description>
230       <ServiceBinding accessURI="http://www.acme.com/purchaseorderservice">
231          <SpecificationLink specificationObject="wsdlForPurchaseOrder">
232              <UsageDescription>
233                  <LocalizedString lang="en_US" value = "This is the WSDL
234                    document that describes the Acme Purchase Order Web Service"/>
235              </UsageDescription>
236          </SpecificationLink>
237      </ServiceBinding>
238  </Service>
```
239

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
```
244  <ExtrinsicObject id="wsdlForPurchaseOrder" mimeType="text/xml">
245      <Name>
246          <LocalizedString lang="en_US" value = "The WSDL document for the Acme Purchase Order Web
247          Service"/>
248      </Name>
249  </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
256      <!--Service objects-->
257
258      <Service id="AcmePurchaseOrderService">
259          <Name>
260              <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>
261          </Name>
262          <Description>
263              <LocalizedString lang="en_US" value = "This Web service will accept purchase orders
264               for Acme Corporation.  It will validate the contents of each purchase order, and, if valid,
265               will process the purchase order and automatically generate an Invoice."/>
266          </Description>
267          <ServiceBinding accessURI="http://www.acme.com/purchaseorderservice">
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
278  <!—WSDL document – ExtrinsicObject -->
279  <ExtrinsicObject id="wsdlForPurchaseOrder" mimeType="text/xml">
280      <Name>
281          <LocalizedString lang="en_US" value = "The WSDL document for the Acme Purchase Order
282            Web Service"/>
283      </Name>
284  </ExtrinsicObject>
285
286    </rim:LeafRegistryObjectList>
287  </SubmitObjectsRequest>
```

5 Extended Scenarios

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

5.1 Versioning of Web Services

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

- **majorVersion:** A registry-assigned major revision number; may be updated by the registry when an object is updated

- **minorVersion:** A registry-assigned minor revision number; may be updated by the registry when an object is updated

- **userVersion:** A user-assigned revision number, similar to the majorVersion-minorVersion tuple

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

5.1.1 Sample XML Instance

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

```
313  <SubmitObjectsRequest>
314      <rim:LeafRegistryObjectList>
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>
340
```

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.

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).

348
349
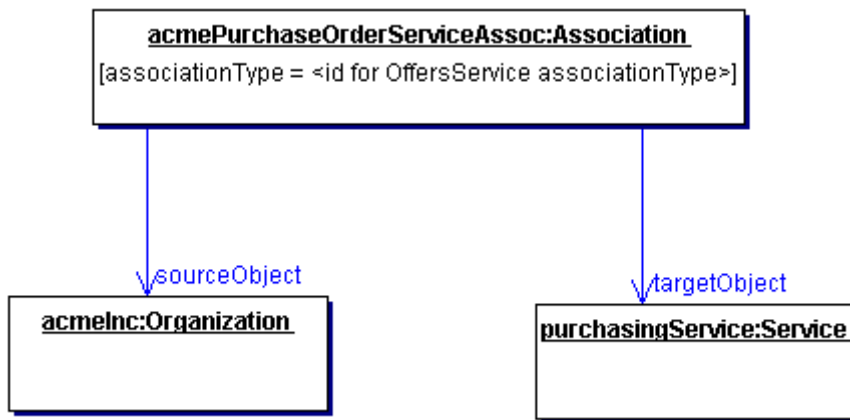
350
351 **Figure 2: Associating a Web Service with an Organization**

352
353
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
361 ```
<SubmitObjectsRequest>
    <rim:LeafRegistryObjectList>

        <!--
        The following association denotes that Acme Corporation offers a Purchase Order Service. The
        sourceObject is the UUID of Acme Corporation's Organization instance, while the targetObject is
        the UUID of the Purchase Order Service's Service instance.
        -->

        <rim:Association id = "AcmePurchaseOrderService-Assoc" associationType =
        "urn:uuid_for_OffersService_association" sourceObject = "urn:uuid:a2345678-1234-1234-
        3345678901292" targetObject = "urn:uuid:a2345678-1234-1234-93456789012"/>

    </rim:LeafRegistryObjectList>
</SubmitObjectsRequest>
```

376
377  In the association above, the "sourceObject" attribute contains the UUID of Acme
378  Corporation's Organization instance, while the "targetObject" attribute contains the
379  UUID of the Purchase Order Service's Service instance.

## 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
```
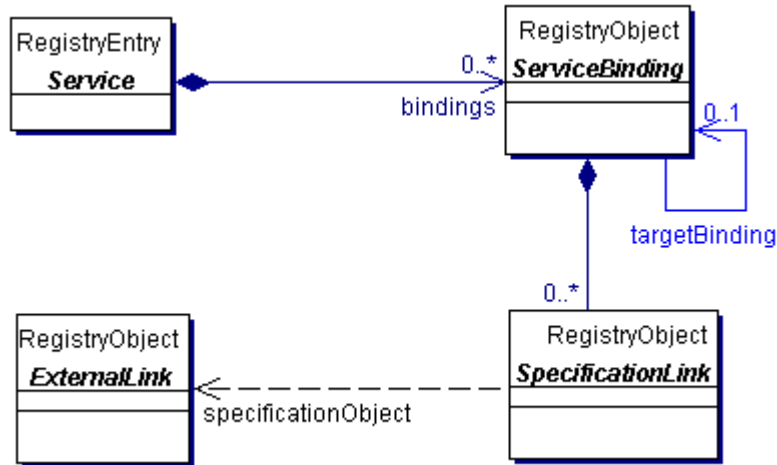<SubmitObjectsRequest>
    <rim:LeafRegistryObjectList>

        <!--
        The following association relates an existing Access Control Policy to Acme Corporation's
        Purchase Order Service. The sourceObject is the UUID of Acme Corporation's Purchase Order
        Service instance, while the targetObject is the UUID of the Access Control Policy instance.
        -->

        <rim:Association id = "AcmePurchaseOrderService-AccessPolicyAssoc" associationType =
        "urn:uuid_for_AccessControlPolicyFor_association" sourceObject = "urn:uuid:a2345678-1234-
        1234-8345678901262" targetObject = "urn:uuid:a2345678-1234-1234-03456789015"/>

    </rim:LeafRegistryObjectList>
</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.

Registering Web Services in an ebXML Registry                                  Page 11

## 5.4  Registering a Service Description that is External to the Registry

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



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

### 5.4.1  Sample XML Instance

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

```
<Service id="AcmePurchaseOrderService">
    <Name>
        <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>
    </Name>
    <Description>
        <LocalizedString lang="en_US" value = "This Web service will accept purchase orders
         for Acme Corporation.  It will validate the contents of each purchase order, and, if valid,
          will process the purchase order and automatically generate an Invoice."/>
    </Description>
     <ServiceBinding accessURI="http://www.acme.com/purchaseorderservice">
        <SpecificationLink specificationObject="urn:uuid_for_ExternalLink_instance">
            <UsageDescription>
                <LocalizedString lang="en_US" value = "This is the WSDL
                 document that describes the Acme Purchase Order Web Service"/>
            </UsageDescription>
        </SpecificationLink>
    </ServiceBinding>
</Service>
```

The "specificationObject" attribute above references an ExternalLink instance that contains the URI for the WSDL document.

5.5 Web Service Redirection

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

5.5.1 Sample XML Instance

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

```
<Service id="AcmePurchaseOrderService">
    <Name>
        <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>
    </Name>
    <Description>
        <LocalizedString lang="en_US" value = "This Web service will accept purchase orders
          for Acme Corporation.  It will validate the contents of each purchase order, and, if valid,
           will process the purchase order and automatically generate an Invoice."/>
    </Description>
    <ServiceBinding accessURI="http://www.acme.com/purchaseorderservice"
                    targetBinding=" urn:uuid_for_ExternalLink_instance">
        ….SpecificationLink element goes here…
    </ServiceBinding>
 </Service>
```

In the above example, Acme Corporation's Purchase Order Service has been rehosted to
a URI that is specified in the ExternalLink instance referenced by the "targetBinding"
attribute above.

5.6 Customizing Metadata Using Slots

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

5.6.1 Sample XML Instance

The following sample XML instance extends the example in Section 3.2.2 by adding
slots for the internal Web Service Administrator Name and whether the Web service is
HTTP(REST)-based or SOAP-based[3]:

```
<Service id="AcmePurchaseOrderService">
    <Name>
        <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service"/>
    </Name>
```

---

[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.

```
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>
```

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

Appendix B   OASIS/ebXML Collaboration-Protocol 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.

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

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
573
574
575
576
577

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-
583   wsdl-v108-20021110.htm

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