Technical Note

Registering Web Services in an ebXML Registry, Version 2.0

June 2005

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

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

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. Distribution of this document is unlimited.

TABLE OF CONTENTS

Figures

1

Introduction

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

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

2 Describing Web Services

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

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

3 Service Information Model

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

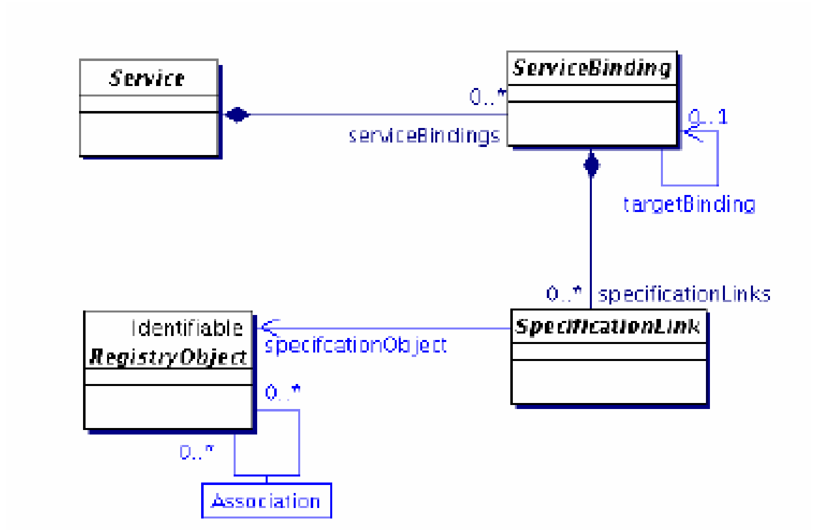

Figure 1: Relationship between RIM classes Service, ServiceBinding, and SpecificationLink

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.

2.1 Class Service

Service instances describe services, such as Web services.

2.1.1 Submission XML Schema Representation

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

```
<element name = "Service" type = "tns:ServiceType"/>

<complexType name = "ServiceType">
	<complexContent>
		<extension base = "tns:RegistryObjectType">
			<sequence>
				<element ref = "tns:ServiceBinding" minOccurs = "0"
							maxOccurs = "unbounded"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

2.1.2 Sample XML Instance

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.

```
<Service id="urn:acme:services:purchaseorder">
	<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 element is placed here…
</Service>
```

2.2 Class ServiceBinding

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

2.2.1 Submission XML Schema Representation

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

```
<element name = "ServiceBinding" type = "tns:ServiceBindingType"/>

<complexType name = "ServiceBindingType">
	<complexContent>
		<extension base = "tns:RegistryObjectType">
			<sequence>
				<element ref = "tns:SpecificationLink" minOccurs = "0"
							maxOccurs = "unbounded"/>
			</sequence>
			<attribute name = "service" use="required" type = "tns:referenceURI"/>
			<attribute name = "accessURI" use="optional" type = "anyURI"/>
			<attribute name = "targetBinding" use="optional" type = "tns:referenceURI"/>
		</extension>
	</complexContent>
```

```
</complexType>
```

2.2.2 Sample XML Instance

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

```
<Service id="urn:acme:services:purchaseorder">
       <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  id="urn:acme:services:bindings:purchaseorder"
                                  service="AcmePurchaseOrderService"
                                  accessURI="http://www.acme.com/purchaseorderservice">
               ….SpecificationLink element is placed here…
        </ServiceBinding>
 </Service>
```

2.3 Class SpecificationLink

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

2.3.1 Submission XML Schema Representation

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

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

<complexType name = "SpecificationLinkType">
       <complexContent>
              <extension base = "tns:RegistryObjectType">
                     <sequence minOccurs = "0" maxOccurs = "1">
                            <element ref = "tns:UsageDescription" minOccurs = "0"
                                                                                 maxOccurs="1" />
                            <element ref = "tns:UsageParameter" minOccurs = "0"
                                                                               maxOccurs="unbounded" />
                     </sequence>
                     <attribute name = "serviceBinding" use="required" type = "tns:referenceURI"/>
```

[1] 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.

```
                <attribute name = "specificationObject" use="required" type = "tns:referenceURI"/>
            </extension>
        </complexContent>
</complexType>

<element name = "UsageDescription" type = "tns:InternationalStringType"  />
<element name = "UsageParameter" type = "tns:FreeFormText" />
```

2.3.2 Sample XML Instance

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

```
<Service id= “urn:acme:services:purchaseorder”>
        <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 serviceBinding="urn:acme:services:bindings:purchaseorder"
                                          specificationObject="wsdlForPurchaseOrder">
                            <UsageDescription>
                                     <LocalizedString lang="en_US" value = "This is the WSDL
                                      document that describes the Acme Purchase Order Web Service"/>
                            </UsageDescription>
                  </SpecificationLink>
         </ServiceBinding>
</Service>
```

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

```
<ExtrinsicObject id="urn:acme:services:descriptions:purchaseorder" mimeType="text/xml">
        <Name>
                <LocalizedString lang="en_US" value = "The WSDL document for the Acme Purchase Order Web
                  Service"/>
        </Name>
</ExtrinsicObject>
```

3 Full SubmitObjectsRequest Example

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

```
<SubmitObjectsRequest comment=”This is the initial submission of the Acme Purchase Order Web
                                                         Service”>
     <rim: RegistryObjectList>

     <!--Service objects-->
```

```
    <Service id="urn:acme:services:purchaseorder">
        <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  id="urn:acme:services:bindings:purchaseorder"
                         service="AcmePurchaseOrderService"
                         accessURI="http://www.acme.com/purchaseorderservice">
            <SpecificationLink serviceBinding="urn:acme:services:bindings:purchaseorder"
                               specificationObject="wsdlForPurchaseOrder">
                <UsageDescription>
                    <LocalizedString lang="en_US" value = "This is the WSDL
                     document that describes the Acme Purchase Order Web Service"/>
                </UsageDescription>
            </SpecificationLink>
        </ServiceBinding>
    </Service>

    <!—WSDL document – ExtrinsicObject -->

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

    </rim: RegistryObjectList>
</SubmitObjectsRequest>
```

4 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.

4.1 Versioning of Web Services

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

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

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

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

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

4.1.1 Sample XML Instance

The following sample XML instance illustrates a change in a 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. The registry will automatically assign versioning attributes as described above, including copying the comment provided for the SubmitObjectsRequest to the RegistryObject.version attribute for the submitted Service:

```
<SubmitObjectsRequest comment="This is an updated version of the Acme Purchase Order Web
                                          Service based on new requirements">
    <rim:RegistryObjectList>

         <Service id="urn:acme:services:purchaseorder:v2.0">
              <Name>
                   <LocalizedString lang="en_US" value = "Acme Purchase Order Web Service – Version
                                                                  2.0"/>
              </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>

          <rim:ObjectRef id = "urn:acme:services:purchaseorder"/>

          <!--
            The following association supersedes the current version of the Service instance with the new
             version that is being submitted.
          -->

         <rim:Association id = "New-AcmePurchaseOrderService-Assoc" associationType =
         "urn:oasis:names:tc:ebxml-regrep:AssociationType:Supersedes" sourceObject =
          " urn:acme:services:purchaseorder:v2.0" targetObject = "urn:acme:services:purchaseorder"/>

    </rim:RegistryObjectList>
</SubmitObjectsRequest>
```

In the association above, the "sourceObject" attribute contains the URN of the new Service instance, while the "targetObject" attribute contains the URN of the old (version 1.0) Service instance.

4.2 Associating a Web Service with an Organization

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

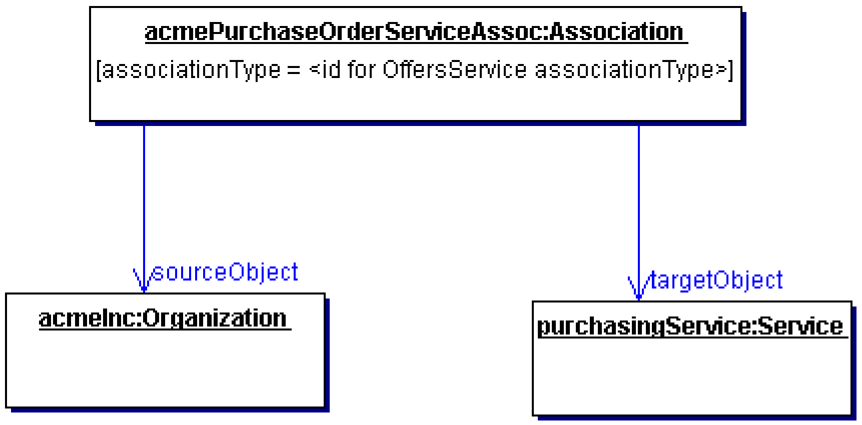
acmePurchaseOrderServiceAssoc:Association
[associationType = <id for OffersService associationType>]
sourceObject
targetObject
acmeInc:Organization
purchasingService:Service

Figure 2: Associating a Web Service with an Organization

4.2.1 Sample XML Instance

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

```
<SubmitObjectsRequest>
    <rim:RegistryObjectList>
         <!--
          The following association denotes that Acme Corporation offers a Purchase Order Service. The
          sourceObject is the URN of Acme Corporation’s Organization instance, while the targetObject is
          the URN 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:Association id = "AcmePurchaseOrderService-Assoc" associationType =
        "urn:oasis:names:tc:ebxml-regrep:AssociationType:OffersService" sourceObject =
         " urn:acme:organization" targetObject = "urn:acme:services:purchaseorder"/>
    </rim:RegistryObjectList>
</SubmitObjectsRequest>
```

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

[UPDATES END HERE]

4.3 Associating a Web Service with an Access Control Policy

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

4.3.1 Sample XML Instance

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

```
<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>
```

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

4.4 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.

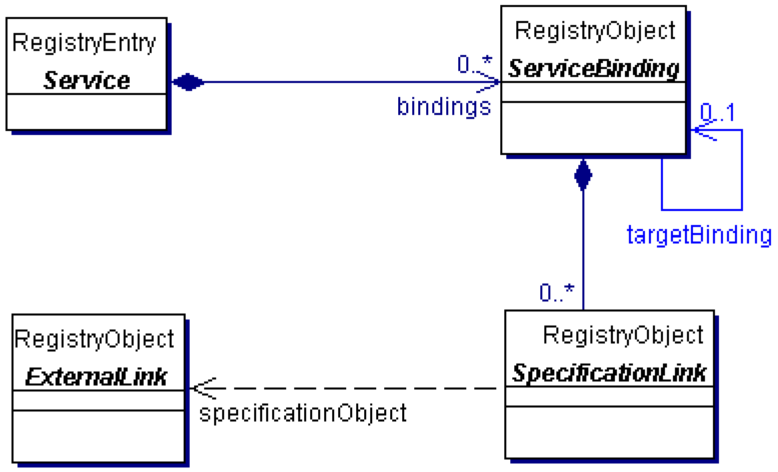

Figure 3: Registering an External Service Description

4.4.1 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.

4.5 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.

4.5.1 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.

4.6 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.

4.6.1 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[2]:

```
<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>
        <Slot name = 'Web Service Administrator Name'>
                <ValueList>
                         <Value>John Smith</Value>
               </ValueList>
```

[2] 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.

```
        </Slot >
        <Slot name = 'HTTP or SOAP'>
              <ValueList>
                    <Value>SOAP</Value>
              </ValueList>
        </Slot >
         <ServiceBinding accessURI="http://www.acme.com/purchaseorderservice">
              ….SpecificationLink element goes here…
        </ServiceBinding>
 </Service>
```

Appendix A WSDL Introduction

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

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

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

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

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

Appendix C DAML-S Introduction

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

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

References

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

http:// xml.coverpages.org/ISWC2002-DAMLS

[ebCPP] ebXML Collaboration-Protocol Profile and Agreement Specification

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

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

[ebRIM Schema] ebXML Registry Information Model Schema v3.0

http://www.oasis-open.org/committees/regrep/documents/3.0/schema/rim.xsd

[ebRS] ebXML Registry Services Specification v3.0 (release pending)

UDDI] Using WSDL in a UDDI Registry, Version 1.8

http://www.oasis-open.org/committees/uddi-spec/doc/bp/uddi-spec-tc-bp-using-wsdl-v108-20021110.htm

[WSA] W3C Web Services Activity

http://www.w3.org/2002/ws/

[WSDL] Web Services Description Language (WSDL)

http://www.w3.org/TR/2002/WD-wsdl12-20020709/