OASIS ebXML Registry

Proposal: ebXML Registry as a Web Service

Category: New functionality to draft specifications

Date: August 15, 2001

Author: ebXML Registry RAWS Sub-team

Status of this Document

This document is a draft proposal whose purpose is to solicit additional input.

1 Abstract

This document proposes focused enhancements to the ebXML Registry Services specification that will allow the ebXML Registry services to be accessible as a set of abstract web services with concrete normative bindings specified for ebXML Messaging Service and SOAP.

Currently the only normative access to the ebXML Registry is over the ebXML Messaging Service. What is lacking is a clean separation between an abstract service interface specification and multiple concrete technology specific bindings (e.g. ebXML Messaging Service) .

The proposal allows more flexibility and ease of access to clients by defining a second normative interface to the ebXML Registry that is based on the widely adopted SOAP protocol.

2 Motivation

The primary motivation behind this proposal is to further ebXML Registry adoption. It is our assertion that adoption is furthered by:

1. Building registry clients with limited infrastructure
2. Enabling additional technology bindings for accessing the registry service
3. Aligning with emerging and de facto standards

ebXML Registry adoption may be measured in the number of operational public ebXML Registries. Currently this number is one. We would like it to higher.

2.1 Building Clients with Limited Infrastructure

Currently, an ebXML Registry client must use an ebXML Messaging service to interact with an ebXML Registry. This requires that the client have access to ebXML Messaging Service infrastructure. This may become a barrier to ebXML Registry adoption.

Making ebXML Registry available as an abstract web service with additional technology bindings (e.g. SOAP) gives clients more options to interact with an ebXML Registry.

A normative SOAP binding (SOAP 1.1 and SOAP with Attachments with http) is proposed since SOAP has considerable mind share and adoption and has in fact been adopted by the ebXML Messaging Service itself. Numerous tools exist that make it very simple for clients to access any SOAP based web service.

2.2 Aligning With Emerging and De Facto Standards

Much has happened in the industry and standards space since ebXML Registry V 1.0 was developed:

1. XML Schema is now a W3C recommendation
2. SOAP Version 1.2 and XML Protocol Abstract Model Working Drafts have been published within W3C
3. WSDL has been submitted as a W3C note
4. Varieties of tools are available that support WSDL and SOAP

The proposal will align ebXML Registry with all of the above standards and trends in the industry and thus further adoption.

3 Proposed Deliverables

The following concrete deliverables are proposed:

1. XML Schema definition for [ebRIM] and [ebRS] with full support for XML namespaces, data types, constraints etc. This schema would replace Registry.dtd
2. Abstract service definition of Registry Services
3. WSDL description of the abstract Registry Services and related concrete SOAP binding

4 Use Cases

4.1 SOAP Based Access of ebXML Registry

An IT shop wants to write a client program to use the ebXML Registry. They do not have the knowledge or infrastructure for using an ebXML Messaging service to access the registry. However, they have the knowledge to use raw SOAP to access ebXML Registry over SOAP. They use the SOAP binding to ebXML Registry to write a custom SOAP client for the ebXML Registry. The client calls are synchronous.

4.2 Automatic Client Stub Generation

The same IT shop now has access to a WSDL compiler that can automatically generated stubs for accessing the SOAP based ebXML Registry services. The stubs provide simplified access to the ebXML Registry in C++ or Java. The client programmer does not even need to know SOAP. All SOAP specific details are hidden in the bindings generated by the WSDL compiler. The client calls are synchronous.

4.3 Support for Other Technology Bindings

The ebXML Registry team may define additional technology bindings for the abstract services defined by this proposal beyond ebXML Registry and SOAP. For example, an IIOP binding may be defined. These bindings could be layered easily on top of the abstract service definitions in WSDL.

5 Changes to [ebRS]

The following sections provide initial text changes to the [ebRS]. These section will need to be edited into the main spec upon approval. Note, that the [ebRS] will likely have systemic impact of this proposal that is not captured in this section.

Note that cross-references are not absolute and will change when merged into main specification. For example, Appendix A in this document may become Appendix E in main specification.

5.1 Removal of Dependency on CPP/CPA

Section 6.1 and other places define linkages between [ebRS] and CPP/CPA. This needs to be replaced with concepts that will work with WSDL also. CPP/CPA should be mentioned as providing an alternate protocol to bootstrap with registry.

5.2 Add Section in Chapter 6: Web Service Based Architecture

This section should be near the beginning of chapter 6. Initial text proposal follows:

The ebXML Registry will expose an abstract registry service that may be implemented using WSDL or ebXML CPP/CPA. Here is a description of the abstract interface …

5.3 Abstract Registry Service

The architecture defines an abstract registry service as shown in Figure 1. The figure shows how an abstract ebXML Registry must provide two key functional interfaces called `QueryManager`[1] (QM) and `LifeCycleManager`[2] (LM). When mapping to WSDL,these interfaces are represented as `port types` within the WSDL description in A.2.

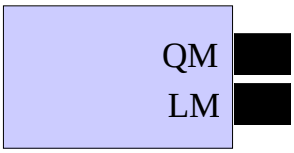

Figure 1: The Abstract ebXML Registry Service

`[Note]Remove Fig 2 in [ebRS]`

[1] Known as ObjectQueryManager in V1.0

[2] Known as ObjectManager in V1.0

5.4 Concrete Registry Services

The architecture further defines how concrete implementations of the abstract registry may be realized as a web service. This is defined in appendix A.3 using `binding` and `service` definitions within the WSDL description, where the abstract `port types` are mapped to `ports` bound to specific protocols.

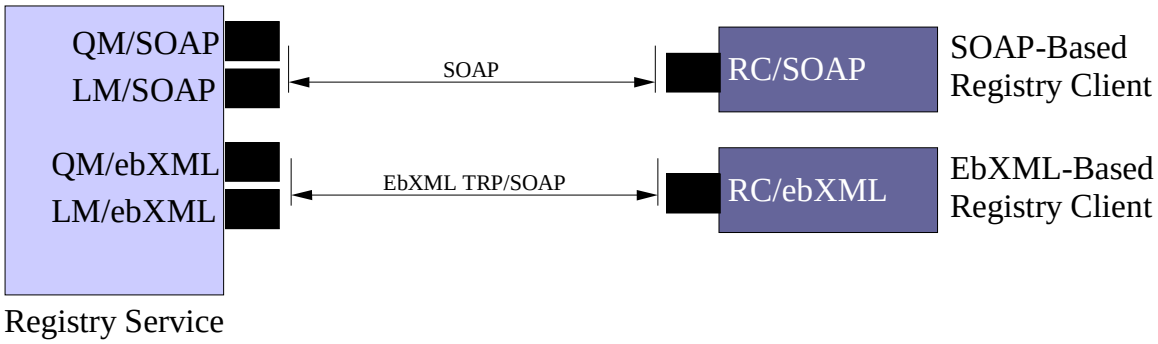

Figure 2: A Concrete ebXML Registry Service

Figure 2 shows a concrete implementation of the abstract ebXML Registry as a web service (name RegistryService) on the left side. The RegistryService provides the QueryManager and LifeCycleManager interfaces available with multiple protocol bindings (SOAP and ebXML TRP). Each interface/protocol combination is defined as a port definition in the WSDL in appendix A.3.

Figure 2 also shows two different clients of the ebXML Registry on the right side. The top client uses SOAP protocol to access the registry while the lower client uses ebXML TRP. Each client uses the appropriate `port` within the RegistryService `service` based upon their protocol preference.

5.5 Interoperability Requirements

The architecture requires that any ebXML compliant registry client can access any ebXML compliant registry service in an interoperable manner. This is done by requiring that all ebXML Registry services, at minimum, support the normative SOAP interface. An ebXML Registry may implement any number of additional protocol bindings in addition to the SOAP protocol. The support of additional protocol bindings is optional.

```
[Note]Need to remove first assumption in section 4.2
     (Caveats and Assumptions) since TRP is no
     longer required. Add requirement that at least
     one of the normative interfaces must be
     supported.
```

5.6 Section 6.2 Changes (Communication Bootstrapping)

This section needs to be re-written to state the following:

Each ebXML Registry must provide a WSDL description for its RegistryService as defined by appendix A.3. A client uses the WSDL description to determine the address information of the RegistryService in a protocol specific manner. For example the SOAP/HTTP based ports of the RegistryService may be accessed via a URL specified in the WSDL for the registry.

The use of WSDL enables the client to use automated tools such as a WSDL compiler to generate stubs that provide access to the registry in a language specific manner.

At minimum, any client may access the registry over SOAP/HTTP using the address information within the WSDL, with minimal infrastructure requirements other than the ability to make synchronous SOAP call to the SOAP based ports on the RegistryService.

5.7 Issues

1. Need to use URNs for namespace specification
2. Terminology for abstract service description of the registry
3. Should a non-normative Registry.dtd be maintained by someone outside the specification.

5.8 Add Reference To WSDL W3C Note

[WSDL] W3C Note. Web Services Description Language (WSDL) 1.1

http://www.w3.org/TR/wsdl

[SOAP11]W3C Note. Simple Object Access Protocol, May 2000, http://www.w3.org/TR/SOAP/

[SOAPAt]W3C Note: SOAP with Attachments, Dec 2000, http://www.w3.org/TR/SOAP-attachments

Appendix A Web Service Architecture

A.1 WSDL Terminology Primer

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 is bound to a specific protocol 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 is simply adds address information such as a URL for each concrete port.

A.2 Registry Service Abstract Specification

Registry.wsdl file goes here

A.3 Registry Service SOAP Binding

RegistrySOAPBinding.wsdl file goes here