WSRP Relationship to UDDI

Jana Polgar
Monash University, Australia

Tony Polgar
Sensis Pty Ltd, Australia

ABSTRACT

In most cases, portlets are built to be deployed by local portals. This is not practical if the organisation wishes to publish their Web services and expects other business partners to use these services in their portals. UDDI extension for WSRP enables the discovery and access to user facing Web services provided by business partners while eliminating the need to design local user facing portlets. Most importantly, the remote portlets can be updated by the Web service providers from their own servers. Remote portlet consumers are not required to make any changes in their portals to accommodate updated remote portlets. This results in easier team development, upgrades, administration, low cost development, and usage of shared resources.

In this chapter, we deal with the technical underpinning of the UDDI extensions for WSRP (user facing remote Web services) and their role in service sharing among business partners. We outline the WSDL extensions relevant to the remote portlets and WSRP (WSRP specification version 1, 2003). publishing and binding process in UDDI.

WEB SERVICES IN UDDI

Portlets (JSR 168, 2005) provide user interface to data delivered from Web services. Before we explain the remote portlet publishing and discovery process in UDDI, we need to refresh the concept of publishing and discovering the Web services in UDDI (Hugo Haas, Moreau, Orchard, Schlimmer, & Weerawarana, 2004)). Web services expose their interfaces by registering in UDDI (UDDI Specifications, 2005). The Web service consumer must find the service, bind to it, and invoke the service. The basic mechanism for publishing and discovering Web services is in Figure 1.

Regardless of whether the Web service will be accessible to a single enterprise or to other companies (public access), the details about the service (its interface, parameters, location, etc.) must be made available to *consumers*. This is accomplished with a WSDL description of the Web service and a Web service directory, where the details of the Web service are published (refer to Web Services Description Language (WSDL)). There are three steps which have to be performed in order to discover and use a Web service published in the UDDI:

Publishing Web service (step 1): In order to be accessible to interested parties, the Web service is published in a Registry or Web service directory. There are several choices regarding where to publish a Web service:

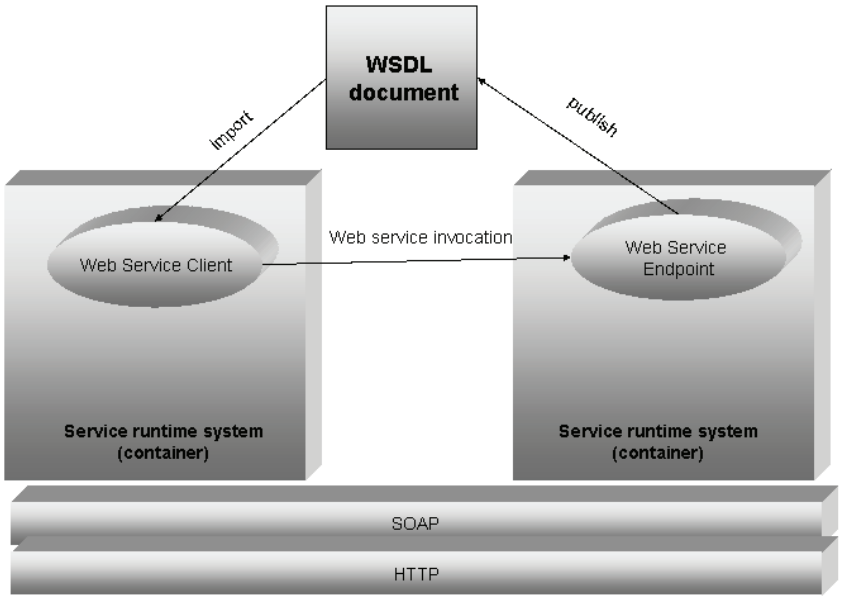
1. If the Web service is intended for the general public, then a well-known registry is recommended. Consequently, the WSDL description, together with any XML schemas referenced by this description, is made public.
2. The Web service intended for enterprise use over an intranet should be published in a corporate registry only. No public access from outside of the firewall is required.
3. Finally, providing all clients are dedicated partners in business, and there is an existing agreement on usage of this service, the Web service can be published on a well-known location on the company server, with proper security access protection. Such a server would be placed on the public side of the company firewall, but it would allow limited access, similar to a B2B Web server.
4. Web services directories are made up of a repository and the taxonomies (classification of registered entities for easier search) associated with them. There are no restrictions on publishing the Web service in multiple registries, or in multiple categories.

Discovery of Web service (step 2): Registry implementations can differ, but there are some common steps, outlined below, that the client must perform before it can discover and bind (step 3) to the service:

1. The client must determine how to access the Web service's methods, such as determining the service method parameters, return values, and so forth. This is referred to as *discovering the service definition interface*.
2. The client must locate the actual Web service (find its address). This is referred to as *discovering the service implementation*.

Copyright © 2007, Idea Group Inc., distributing in print or electronic forms without written permission of IGI is prohibited.

Figure 1. Publish-find-bind mechanism in UDDI



Bind to the Web service and invoke it (step 3): The client must be able to bind to the service's specific location. The following types of binding may occur:

1. Static binding during client development or at the deployment time.
2. Dynamic binding (at runtime).

From the client point of view, the binding type and time play important roles in possible scenarios relevant to the client's usage of the Web service. The following situations are typical:

1. A Web service (WSDL and XML schemas) is published in well-known locations. The developers of the application that use the service know the service, its location, and the interface. The client (which is a process running on a host) can bypass the registry and use the service interfaces directly. Alternatively, the client knows the location and can statically bind to the service at the deployment time.
2. The Web service expects its clients to be able to easily find the interface at build time. These clients are often generic clients. Such clients can dynamically find the specific implementation at runtime using the registry. Dynamic runtime binding is required.

Development of Web service clients requires some rules to be applied and design decisions to be made regarding which binding type is more appropriate for the given situation (static or dynamic binding). Three possible cases are discussed:

1. **Discovering the service interface definition:** If we are dealing with a known service interface, and the service implementation is known (no registry is required), the actual binding should be static.
2. **Discovering the service implementation:** In this case, static binding is also appropriate because we know the interface. We need to discover the service implementation only at build time.
3. The client does not know the service interface and needs to discover the service interface dynamically at build time. The service implementation is *discovered dynamically at runtime*. This type of invocation is called Dynamic Invocation Interface (DII). In this case, the binding must be dynamic.

Each WSDL description of the service published in UDDI must contain the following six elements: definitions, types, message, portType, binding, and service. The main elements of the UDDI data model are listed as follows (Figure 2):

5 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage: www.igi-global.com/chapter/wsrp-relationship-uddi/18032

Related Content

State Portals as a Framework to Standardize E-Government Services
Paul Chalekian (2007). *Encyclopedia of Portal Technologies and Applications (pp. 968-973).*
www.irma-international.org/chapter/state-portals-framework-standardize-government/17994

A Comprehensive Methodology for Campus Portal Development
Tharitpong Fuangvutand Helen Hasan (2007). *Encyclopedia of Portal Technologies and Applications (pp. 166-171).*
www.irma-international.org/chapter/comprehensive-methodology-campus-portal-development/17864

An Open Streaming Content Distribution Network
Giancarlo Fortinoand Carlos E. Palau (2007). *Encyclopedia of Portal Technologies and Applications (pp. 677-683).*
www.irma-international.org/chapter/open-streaming-content-distribution-network/17947

Adaptive Web Services Monitoring in Cloud Environments
Yi Weiand M. Brian Blake (2013). *International Journal of Web Portals (pp. 15-27).*
www.irma-international.org/article/adaptive-web-services-monitoring-cloud/78350

Script Familiarity and Its Effect on CAPTCHA Usability: An Experiment with Arab Participants
Ashraf Khalil, Salam Abdallah, Soha Ahmedand Hassan Hajjdiab (2012). *International Journal of Web Portals (pp. 74-87).*
www.irma-international.org/article/script-familiarity-its-effect-captcha/73916