# Experiences with the Development and Deployment of WS-* Specifications

Shrideep Pallickara, Geoffrey Fox, Mehmet Aktas, Harshawardhan Gadgil, Sangyoon Oh, Sima Patel, Damodar Yemme and Beytullah Yildiz
(spallick, gcf, makatas,hgadgil, ohsangy,skpatel, dyemme,byildiz)@indiana.edu
Community Grids Lab, Indiana University

In this chapter we report on our experiences in implementing several Web Service specifications and their deployments in containers that host the web service endpoints involved in interactions with each other. We plan to begin this chapter with some observations regarding Web Service specifications. This includes the Simple Object Access Protocol (SOAP) [1] centric nature of the specifications, their reliance on one-way asynchronous message exchanges, how a specification can itself leverage other specifications, and, finally, how these specifications are intended to be stackable to facilitate use in tandem with each other.

Several specifications leverage the WS-Addressing [2] specification, and we will present an overview of the WS-Addressing specification and the various rules associated with this specification. WS-Addressing, has in recent years, become the de facto standard to target Web Service instances, and is also leveraged quite heavily in the Web Service Resource Framework (WSRF) [3] which is the dominant framework for building Grid applications.

Next we present an overview of the specifications that we have implemented. These include WS-ReliableMessaging [4], WS-Reliability [5], WS-Eventing [6], WS-Context [7], the Universal Description, Discovery and Integration (UDDI) [8], WS-Management [9] and WS-Transfer [10]. Depending on the interactions and exchanges that are part of these specifications the complexity of the implementation and corresponding deployments varies. The WS-ReliableMessaging and WS-Reliability specifications pertain to providing support for reliable messaging between Web service endpoints. These aforementioned specifications guarantee delivery of messages in the presence of failures and disconnects; endpoints can retrieve lost messages after a failure. The WS-Eventing specification provides a mechanism for routing notifications from the producers to the registered consumers. Consumers can register their interest in specific messages using XPath queries; only messages that satisfy the previously specified constraint are routed to a consumer. WS-Context models session metadata as an external entity where more than two services can access/store highly dynamic shared metadata. The UDDI specification defines a searchable repository of Web Service Description Language (WSDL) [11] specifications of Grid/Web Services. We have designed and implemented a hybrid information service that provides semantics for both types of information that are defined by the WS-Context and UDDI Specifications.

In distributed applications there is a need for effective management of components. Here, management typically involves common operations such as the ability to control the resource (e.g. start and stop), configure the resource for a specific task and to monitor the status (e.g. heartbeat) of the resource. We have been designing a WS-Management based framework that performs these functions in the context of managing resources related to the NaradaBrokering (http://www.naradabrokering.org) substrate. We also provide a WS-Transfer based interface to CREATE and DELETE these resources.

We then proceed to describe our implementation strategy for several of these Web Service specifications in detail. Here there are four crucial phases. First, we need to make a decision on the choice of the tool to process schemas associated with a given specification and the rationale for its use. In the second phase we need to cope with specifications that are leveraged by the specification in question. For example, the WS-ReliableMessaging specification leverages both WS-Addressing

and WS-Policy. The third phase pertains to the actual implementation, and enforcement, of the rules and processing associated with the specification. This includes incorporating support for all exchanges, and reporting any faults, as specified in the specification. The final phase corresponds to the deployment of these implementations within various Web Service containers.

Before, we proceed with the deployment phase of Web Service specifications we include a discussion on our approach towards high performance Web Services. In our scheme we separate message content from the XML syntax, choose a preferred representation and exchange messages in a streaming fashion. This increases the efficiency of message exchange, since applications can avoid the textual conversion and conventional parsing. Message size can be further reduced by storing static parts of the message and having an optimal representation, for subsequent messages, that can conserve network bandwidth. The Handheld Flexible representation architecture that we have developed incorporates these strategies.

We then proceed to the deployment phase of the specifications. Here we also describe the prototype lightweight container that we developed to test the various exchanges that entities have in some of the Web Service specifications that we implemented. In our deployment experiments we leveraged the open-source Apache Axis Web Service container. Here, we encountered several interesting issues related to support for exchanges within some of the WS-* specifications. We will report on the problems that we encountered and our solutions to circumvent these problems.

Finally, we also report on our experiences with deployments of Web Service specifications in the United Kingdom's Open Middleware Infrastructure Institute's (OMII) container. The OMII container's security framework leverages WS-Security. Here, we describe how web service endpoints can leverage this security framework.

## References


[1] . M. Gudgin, et al, "SOAP Version 1.2 Part 1: Messaging Framework," June 2003. http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

[2] . Web Services Addressing (WSAddressing) ftp://www6.software.ibm. com/software/developer/library/ wsadd200403.pdf

[3] . The Web Services Resource Framework. http://www.globus.org/wsrf/

[4] . Web Services Reliable Messaging Protocol (WS-ReliableMessaging) http://www-106.ibm.com/ developerworks/webservices/library/ws-rm/

[5] . Web Services Reliability TC WS-Reliability. http://www.oasis-open.org/committees/download.php/ 5155/WS-Reliability-2004-01-26.pdf

[6] . Web Services Eventing. Microsoft, IBM & BEA. http://ftpna2.bea.com/pub/downloads/WS-Eventing.pdf

[7] . Bunting, B., Chapman, M., Hurley, O., Little M,, Mischinkinky, J., Newcomer, E., Webber, J., and Swenson, K., "Web Services Context (WS-Context), http://www.arjuna.com/library/specs/ws_caf_1-0/WS-CTX.pdf

[8] . T. Bellwood, L. Clement, and C. Riegen, "UDDI Version 3.0.1: UDDI Specification," Technical Committee Specification. http://uddi.org/pubs/uddi-v3.0.1-20031014.htm.

[9] . Web Service Management (WS - Management), Sun, Microsoft, Intel, et al., June 2005. Available from https://wiseman.dev.java.net/specs/2005/06/management.pdf

[10] . Web Service Transfer (WS - Transfer), Microsoft et al., September 2004. Available from http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-transfer.pdf

[11] . Web Services Description Language (WSDL) 1.1 http://www.w3.org/TR/wsdl