# Definition Languages for RESTful Web Services: WADL vs. WSDL 2.0

Toshiro Takase[1], Satoshi Makino[1], Shinya Kawanaka[1], Ken Ueno[1],
Christopher Ferris[2], and Arthur Ryman[3]
[1]*Tokyo Research Laboratory, IBM Research*
*{E30809, MAK0702, SHINYAK, KENUENO}@jp.ibm.com*
[2]*Standards Strategy, Software Group*
*chrisfer@us.ibm.com*
[3]*Rational Division, Software Group*
*ryman@ca.ibm.com*

June 21, 2008

***Abstract***

*There are two specifications for describing interfaces of HTTP based web applications, WADL and the WSDL 2.0 HTTP binding extension. These two languages are very similar, but there are some differences. This paper attempts to provide an unbiased, objective comparison of the two technologies, highlighting both the differences and similarities between WADL and the WSDL 2.0 HTTP binding.*

## 1. Introduction

The World Wide Web has, until recently, not had a formal means by which a web application can be described in a machine-processable manner. In a purely browser-based context, it has not been necessary to provide such a description, because the web application provided its interface to the user in the form of (X)HTML and web forms (such as HTML forms and XForms [7]) that were rendered in the browser, to be interacted with directly by an end user.

However, the lack of a formal description of a web application's interface has made it difficult (though certainly not impossible) to develop non-browser-based interactions with web applications. Authors of web applications that wanted to encourage non-browser-based access typically provided natural language-based descriptions that were subject to both mis-interpretation and version skew issues (e.g where the application's interface changed but the natural language description did not change accordingly).

While the natural language-based description of web applications are of value, such descriptions do not aid in automating and/or simplifying the development of the software intended to interact with the described web applications. Additionally, it is a fairly common occurrence that the description becomes inconsistent with the actual service interface, because there is no formal link between the two.

Two specifications have emerged that each aim to provide a machine-processable means of formally describing the interface to a REST style web application: WADL (Web Application Description Language) [1] and the WSDL 2.0 (Web Services Description Language) [3] HTTP binding extension.

A machine processable description provides us with an opportunity to develop tooling and runtime software that can simplify the both the design and development of software intended to interact with the service. Such descriptions, when either generated from the web application's interface, or used to generate the interface, ensure that the description is aligned with the implementation, thus mitigating the issues with natural language-based descriptions covered above.

In this paper, we offer an unbiased comparison of these two technologies as they relate to the expression of a REST-style service interaction and describe any substantive differences between the two.

## 2. WADL and WSDL 2.0

WADL is a description language for HTTP-based Web applications, such as applications which follow the REST (Representational State Transfer) [2] architectural style.

As of this writing, WADL is not being standardized by any standards development organization, such as the W3C or OASIS. The latest version of WADL was published on November 9, 2006, by Sun Microsystems. The author of the specification, Marc J. Hadley, is a senior staff engineer in the Office of the CTO, at Sun Microsystems. He represented Sun on the W3C XML Protocol and W3C WS-Addressing working groups where he is co-editor of the SOAP 1.2 and WS-Addressing 1.0 specifications. He is also co-specification lead for JSR-224 (Java API for XML-Based Web Services (JAX-WS) 2.0) [4] and JSR-311 (JAX-RS: The Java API for RESTful Web Services) [5]. Tools that can automatically generate code (both client and server) from a WADL description, such as wadl2java, are being developed at java.net [6]. There is a discussion list related to WADL and other web application description languages hosted by the W3C and archived at http://lists.w3.org/Archives/Public/public-web-http-desc/.

WSDL 2.0 is a specification for describing the interfaces of a Web Service, such as SOAP-based applications that also offers the ability to describe web applications that do not use SOAP.

WSDL 2.0 was released by W3C as a Recommendation on June 26, 2007. The WSDL 2.0 specification is a formal standardization of WSDL 1.1 [25], which has become a de facto standard for the description of Web services. While WSDL 2.0 is now a W3C Recommendation, WSDL 1.1 remains the predominant technology for describing Web Services as of this writing. However, despite the fact that WSDL1.1 is still the predominant technology, tooling and runtime support for WSDL 2.0 is being developed by the Apache Woden project [27] and certain vendor products have begun to incorporate support for WSDL 2.0, but we won't list them here. Additionally, other standards specifications have begun to normatively reference WSDL 2.0, including WS-Policy 1.5, SPARQL and WS-Addressing 1.0. Thus, it is only a matter of time before WSDL2.0 becomes more broadly adopted.

Here is what's new in WSDL 2.0 as contrasted with WSDL 1.1:

First, WSDL 2.0 has a much better HTTP binding than its WSDL1.1 predecessor. It allows you to specify the HTTP verb (such as GET, PUT, or POST) on a per-operation basis, unlike

WSDL 1.1 which was either all GET or all POST for all operations in a given WSDL binding. It also provides a flexible binding of input parameters to URIs.

Second, WSDL 2.0 let's you describe references to other Web services. This is like hyperlinking for Web services. Instead of just saying in the schema "this is an xsd:anyURI" you can add attributes to a URI that specify the interface or binding that is to be expected when interacting with the endpoint at that URI.

To read about this, we recommend the WSDL 2.0 Primer, section 5.3 "Describing Web Service Messages that Refer to Other Web Services" [8]. To see the actual WSDL and XSD files, look at the W3C Test Suite, ServiceReference-1G [9].

The attributes to describe the interface and binding of endpoints are described in the specification, Part 1, 3.3 "Describing Messages that Refer to Services and Endpoints". [10]

Third, there is a "safe" attribute for describing operations that exhibit no side-effects that would cause modification of state as a result of the operation being invoked. Such operations map naturally to an HTTP GET, but could equally apply in the context of a WS-Transfer GET [22] .

The HTTP binding is described in detail in Part 2, section 6 "WSDL HTTP Binding Extension" [11]. The specification includes a mechanism to describe the input parameters that can be mapped onto a URI template. These parameters are described abstractly using a simple XML schema that follows the IRI style rules [12]. These get bound to the request URL using flexible rules [13] that let you include some parameters in the base URL and some as query parameters.

There are several examples of the WSDL 2.0 HTTP binding in the Test Suite, including WSDL 2.0 descriptions of the Flickr [14] and Bugzilla [15] web application interfaces.

**3. REST vs. SOAP**

REST is the term coined by Roy Fielding in his Ph.D thesis. [2] REST is the architectural style that informed the development of HTTP (Hypertext Transfer Protocol) [16], though the thesis was written long after the HTTP protocol had been finalized. Some people say that the REST architectural style defines an appropriate usage of HTTP.

In the REST architectural style, resources are basic objects and accessed by unique URIs. HTTP explicitly supports CRUD (Create, Read, Update and Delete) basic operations for resources by HTTP methods (PUT, GET, POST, and DELETE). REST interactions are stateless, in other words, state should be maintained within the resource. This stateless characteristic makes the resources (which are accessed by GET method) cacheable. WADL specification does not say that WADL is only for REST explicitly however, WADL covers only HTTP.

SOAP is a messaging layer protocol that is independent of any underlying transport protocol such as HTTP. In many cases, SOAP is transferred between endpoints using HTTP. However, SOAP can be transferred between endpoints using other protocols such as SMTP, FTP, Message Queuing (like, WebSphere MQ), and so on. Some people say that HTTP POST usage by SOAP is incorrect against HTTP method usage. Actually,

the argument is that SOAP, when used as an RPC-style interaction, can result in situations in which the HTTP protocol verb used (POST) is inconsistent with the semantic of the operation, which could be GET, PUT or DELETE. The classic example of such inconsistency is the "getStockQuote" example often used in SOAP tutorials. The argument is that in order for SOAP to correctly leverage the HTTP protocol in a RESTful manner, that such an operation would actually be bound to the HTTP GET operation, not an HTTP POST as required by the SOAP/HTTP binding (note that this is only the case for SOAP 1.1, which does not have a SOAP Response MEP as does SOAP 1.2). This argument is what prompted the XML Protocol WG to define the SOAP Response MEP and corresponding HTTP binding. It allows an HTTP GET to be used to retrieve a SOAP envelope as the representation of the resource identified by the URI to which the HTTP GET was issued.

We would point out that SOAP is increasingly used for document style interactions, not just for invoking operations. The term "operations" is something that is attributed to SOAP largely because of WSDL, and because the first usages of SOAP was rpc-centric in nature. WSDL 2.0 still preserves this operation-centric aspect.

WSDL (both 1.1 and 2.0) was specifically designed such that it could describe any service interface, not just those that use SOAP. Both WSDL 1.1 and 2.0 explicitly include specification of SOAP binding extensions (for both SOAP 1.1 and SOAP 1.2) and both include an HTTP binding extension. However, it is widely accepted that the WSDL 1.1 HTTP binding extension is inadequate in many regards. For instance, the WSDL 1.1 HTTP binding extension does not support all of the HTTP verbs and it only permits use of a single verb for all of the operations in a portType. The WSDL 1.1 HTTP binding extension received little in the way of product-level support, possibly, because WS-I expressly excluded its use in the WS-I Basic Profile [27].

The WSDL 2.0 HTTP binding extension was designed to address the inadequacies of the WSDL 1.1 extension and to provide improved support for the description of REST-style applications. However, what WSDL 2.0 doesn't provide is a resource-centric model. It is still very interface-centric.

## 4. Differences between WADL and WSDL 2.0 HTTP binding

Although both WADL and the WSDL 2.0 HTTP binding are very similar in certain regards, there are some differences. In this section, we describe those differences in detail.

### 4.1. Resources vs. Interfaces

WADL is a resource-centric description language. A WADL document is composed as a set of resource descriptions. On the other hand, WSDL is an interface-centric description language.

A WSDL 2.0 description is composed as a set of interface definitions which are each comprised of operation definitions.

In WADL, even a complicated business application is described as basic operations (PUT, GET, POST, or DELETE) on the resources that comprise the application's state.

### 4.2. HTTP only vs. transport protocol independent

WADL only supports the description of web applications that use the HTTP protocol. Because it is limited to HTTP, WADL is a much simpler language than WSDL 2.0, which had as a design goal the capability of describing service interfaces that can use just about any protocol imaginable. However, there is value in simplicity. WADL was specifically designed to provide a means of describing an HTTP-based web interface. The predominant protocol of the web is HTTP. True, there are other protocols such as FTP and SMTP. However, of the protocols available, only HTTP is remotely RESTful.

On the other hand, the WSDL 2.0 description language is independent of transport protocol. Note that technically, HTTP is a "transfer" protocol, not a transport protocol. This is an important distinction that is often misunderstood. This is largely because HTTP itself is a general purpose "application" protocol. While WSDL is certainly independent of transport protocol, it treats HTTP as a transport protocol, rather than as an application protocol. If one were to align the two, one would describe the HTTP application as an interface that had the methods GET, PUT, POST, DELETE, etc. WSDL 2.0 explicitly includes specifications of both a SOAP binding extension and an HTTP binding extension. The parts other than binding in WSDL document can be reused by other bindings. See the example below.

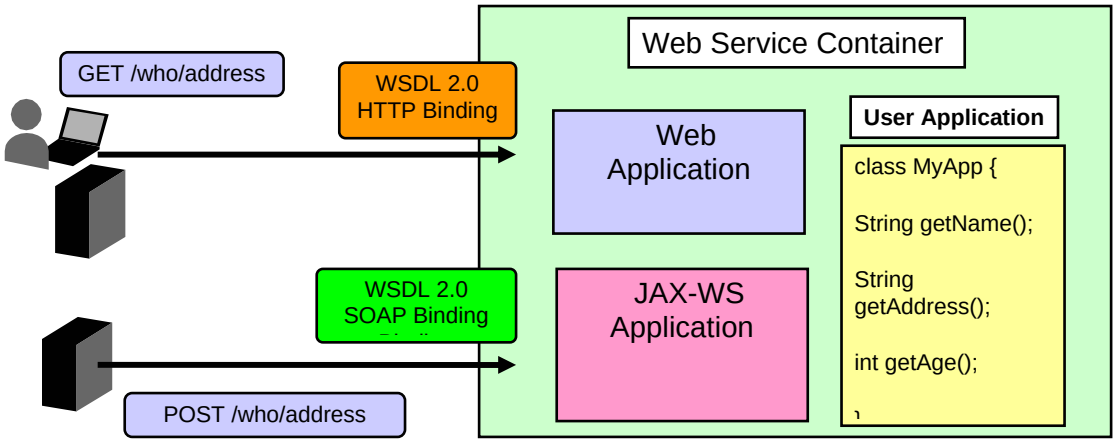


Figure 1. WSDL 2.0 Transport Independent Bindings

### 4.3. Message exchange pattern

Some believe that the only message exchange pattern implied by HTTP is request-response (in-out) only. Despite the fact that HTTP is inherently request-response it can be used to affect both one-way and solicit-response message exchange patterns. Note that the HTTP GET verb is inherently solicit-response in nature, in the sense that the GET request message contains only the identifier of the resource for which a representation is to be retrieved. HTTP PUT and DELETE are effectively one-way operations for which there is a response that contains a simply a status message (200 OK, 404 Not Found, etc.).

WSDL 2.0 defines a number of message exchange patterns, such as one-way (in-only) and notification (out-only) as well as request-response. Although the WSDL 2.0 specification does not specify such bindings, some transport protocols, such as SMTP and Message Queuing, have native support for one-way and notification. It is expected that specifications of those bindings would map the WSDL defined message exchange patterns to those protocols. The WSDL 2.0 specification's HTTP binding extension

section specifies how to implement a one-way message exchange pattern that uses the HTTP response message with a status code 202 Accepted (See, WSDL 2.0 Part 2 Section 6.5.1). Also, the notification message exchange pattern on HTTP may be implemented by using comet-like technique [23].

Unlike WSDL 2.0, which is intended to be transport independent (see section 2.2), WADL is only used to describe HTTP-based web applications. As such, its message exchange patterns (although WADL does not define or use such a term) are those supported by HTTP.

### 4.4. Stateless vs. Stateful

Although the REST architectural style requires that interactions be stateless, there is often confusion as to what is meant by the term “stateless”.

A resource (the thing identified by a URI) in a REST context isn’t stateless in and of itself. Rather, the statelessness of REST has to do with the protocol interaction with the resource identified by a URI. This means that all of the information necessary to fulfill a request is carried in that request message. Resources themselves are (often) inherently stateful. A resource that is a bank account balance is certainly stateful. We can assign a URI to that resource. Performing an HTTP GET on that resource URI will return a different value over time, reflecting the change in the state of that resource as the account balance increases and decreases as a result of activity against that account. However, if the GET is truly stateless (e.g. no abuse of cookies) then any HTTP get on that resource URI, by any user will always return a representation of that resource, and only that resource. Statelessness in the protocol interaction enables the caching feature of the REST architectural style. A caching proxy can be interposed, between the client and the server, that can cache the response to an HTTP GET on a URI and subsequently return a cached version of that resource in response to any other HTTP GET on that URI so long as the origin server has assigned appropriate cache pragmas on the original response message, indicating when the cache should be invalidated, etc.

There is one aspect of the HTTP protocol that can be abused from a RESTful perspective, thus violating the stateless constraint in the REST architectural style: HTTP cookies. Roy Fielding, the author of the REST thesis, has called HTTP cookies anathema and has said that if he had to do it over again, he would not have permitted their introduction into the protocol. While cookies *can* be used in a RESTful manner, they are often abused (from a REST perspective) as a means of associating session state without which the interaction becomes meaningless.

As an example, if a cookie is used to associate with some session state such as the user’s postal code that is then used to define the search criteria for a weather service, the URI would not necessarily identify the same resource (the current weather in the original user’s postal code area) when used in an HTTP GET by someone else with a different postal code. A RESTful use of a cookie in this context might be for tracking the user’s interaction, but use of the cookie to associate some pre-established session state with a request is inherently non-RESTful.

Although the WADL specification does not require that a web application (or interactions with the resources described by WADL) be stateless, the specification does not provide for any of the stateful features of HTTP, such as cookies. On the other hand,

the WSDL 2.0 HTTP binding specifically provides for use and description of HTTP cookies (See, WSDL 2.0 Part 2 Section 6.10).

### 4.5. Authentication

Typically, HTTP authentication is handled automatically because the HTTP protocol provides for standard status responses that the server can return in the event that the client has not provided the required credentials for a request. Such status responses (e.g. a 401 Unauthorized) can be used by the client to trigger the collection of credentials. Adding details of the authentication credentials required to access a resource to the description of a service is a useful means of optimizing the interaction such that the client can pre-collect the requisite credentials and present them with the original request, rather than to have an original request be pre-processed by the server, only to be returned with a response that indicates that authentication credentials are needed to fulfill the request. WSDL 2.0 supports HTTP authentication (See, WSDL 2.0 Part 2 Section 6.11), while WADL does not.

### 4.6. URL encoded data

Many web applications make use of URL encoded data (media type: “application/x-www-form-urlencoded”) as a means of “composing” the URI for a resource based upon some parameterized criteria, such as information collected from a web form. Currently, there exists no standard language for describing URL encoded data, although there are a number of proposals to define a URI template language.

#### 4.6.1. URI template

Both WADL and the WSDL 2.0 HTTP binding use a form of URI template to specify query string parameterization of URIs. WADL refers an internet draft named “URI template” [24] whose author is the same author of WADL. On the other hand, WSDL 2.0 defines an original URI template in the WSDL 2.0 HTTP binding extension specification (See, WSDL 2.0 Part 2 Section 6.8.1).

#### 4.6.2. Percent encoding in URL encoded data

WADL does not mention character encoding in URL encoded data. WSDL 2.0 HTTP binding explicitly specifies how to handle non-ASCII characters. (See, WSDL 2.0 Part 2 Section 6.5.3).

### 4.7. Schema language for describing message content

Both WADL and WSDL 2.0 provide a means by which the content of messages exchanged are described using a schema language.

WADL specifically provides for the ability to include both W3C XML Schema [17] and RelaxNG [18] schema documents that describe the content of messages exchanged.

WSDL 2.0 only specifies how to include a W3C XML Schema document in a description. The WSDL 2.0 specification does not provide an explicit use of RelaxNG to describe the content of messages. WSDL 2.0 does not require the use of XML Schema since the content of the wsdl:type element is an extensibility point. However, because the WSDL 2.0 specification is a product of the W3C, it is only natural that they would give preference to and explicitly specify usage of XML Schema.

## 5. Conclusion

WADL and WSDL 2.0 HTTP binding are similar but do have some differences. Each specification has both its pros and cons. In short, WADL is simple and has limited scope. By design, WADL is limited to describing HTTP applications and does not address features such as security. On the other hand, the WSDL 2.0 HTTP binding is more feature rich, at the cost of increased complexity, yet still lacks a true resource-centric model.

It will be interesting to watch as each of these technologies matures and gains broader adoption.

## Appendix A. WSDL 2.0 example for the Yahoo News Search Examples

In this section, we present WSDL 2.0 example for existing Yahoo New Search service. Note that you will be able to find a WADL description example for the same service in the WADL specification, Section 1.3 [21].

### A.1. WSDL 2.0 example

The following listing shows an example of a WSDL 2.0 description for the Yahoo News Search application. We write this example for comparison.

```
<?xml version="1.0" encoding="utf-8" ?>
<description
     xmlns="http://www.w3.org/ns/wsdl"
     targetNamespace="urn:yahoo:yn"
     xmlns:tns="urn:yahoo:yn"
     xmlns:yn="urn:yahoo:yn"
     xmlns:ya="urn:yahoo:api"
     xmlns:xs="http://www.w3.org/2001/XMLSchema"
     xmlns:whttp="http://www.w3.org/ns/wsdl/http" >

  <types>
    <xs:import namespace="urn:yahoo:yn"
                schemaLocation="http://search.yahooapis.com/NewsSearchService/V1/NewsSe
archResponse.xsd"/>
    <element name="search" xmlns="http://www.w3.org/2001/XMLSchema">
      <complexType>
        <sequence>
          <element name="appid" type="xs:string" />
          <element name="query" type="xs:string" />
          <element name="type" minOccurs="0">
            <simpleType>
              <restriction base="xs:NMTOKEN">
                <enumeration value="all"/>
```

```
            <enumeration value="any"/>
            <enumeration value="phrase"/>
          </restriction>
        </simpleType>
      </element>
      <element name="results" type="xs:int" minOccurs="0" />
      <element name="start" type="xs:int" minOccurs="0" />
      <element name="sort" minOccurs="0">
        <simpleType>
          <restriction base="xs:NMTOKEN">
            <enumeration value="rank"/>
            <enumeration value="date"/>
          </restriction>
        </simpleType>
      </element>
      <element name="language" type="xs:string" minOccurs="0" />
    </sequence>
  </complexType>
 </element>
</types>

<interface name="YahooNewsSearchInterface">
 <operation name="YahooNewsSearchOperation"
           pattern="http://www.w3.org/ns/wsdl/in-out">
  <input messageLabel="In" element="tns:search" />
  <output messageLabel="Out" element="yn:ResultSet" />
  <outfault messageLabel="Out" element="ya:Error" />
 </operation>
</interface>

<binding name="YahooNewsSearchHTTPBinding"
   interface="tns:YahooNewsSearchInterface"
   type="http://www.w3.org/ns/wsdl/http" whttp:methodDefault="GET">

 <operation ref="tns:YahooNewsSearchOperation"
   whttp:location="newsSearch/" />
</binding>

<service name="YahooNewsSearchService"
 interface="tns:YahooNewsSearchInterface">

 <!-- HTTP 1.1 GET End Point -->
 <endpoint name="YahooNewsSearchEndpoint"
   binding="tns:YahooNewsSearchHTTPBinding"
   address="http://api.search.yahoo.com/NewsSearchService/V1/"/>
</service>
</description>
```

## References


[1] Web Application Description Language (WADL), Marc J. Hadley, Sun Microsystems Inc. November 9, 2006. Available at https://wadl.dev.java.net/wadl20061109.pdf

[2] Representational State Transfer (REST), Roy Thomas Fielding, Author. 2000. Available at http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[3] Web Services Description Language (WSDL) Version 2.0 Part 0: Primer, W3C Recommendation 26 June 2007. Available at http://www.w3.org/TR/wsdl20-primer/

Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C Recommendation 26 June 2007. Available at http://www.w3.org/TR/wsdl20/

Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts, W3C Recommendation 26 June 2007. Available at http://www.w3.org/TR/wsdl20-adjuncts/

[4] JSR 224: JavaTM API for XML-Based Web Services (JAX-WS) 2.0. Available at http://jcp.org/en/jsr/detail?id=224

[5] JSR 311: JAX-RS: The JavaTM API for RESTful Web Services. Available at http://jcp.org/en/jsr/detail?id=311

[6] java.net, wadl: Web Application Description Language (WADL) - Specification and Tools. Available at https://wadl.dev.java.net/

[7] XForms 1.0 Third Edition, W3C Recommendation 29 October 2007.

Available at http://www.w3.org/TR/2007/REC-xforms-20071029/

[8] http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626/#adv-service-references

[9] http://dev.w3.org/cvsweb/2002/ws/desc/test-suite/documents/good/ServiceReference-1G/

[10] http://www.w3.org/TR/2007/REC-wsdl20-20070626/#wsdlx-references

[11] http://www.w3.org/TR/2007/REC-wsdl20-adjuncts-20070626/#http-binding

[12] http://www.w3.org/TR/2007/REC-wsdl20-adjuncts-20070626/#_operation_iri_style

[13] http://www.w3.org/TR/2007/REC-wsdl20-aduncts-20070626# http operation location cited ser

[14] http://dev.w3.org/cvsweb/2002/ws/desc/test-suite/documents/good/FlickrHTTP-1G/

[15] http://dev.w3.org/cvsweb/2002/ws/desc/test-suite/documents/good/W3CBugzillaHttp-1G/

[16] RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1.

Available at http://www.ietf.org/rfc/rfc2616.txt

[17] XML Schema Part 0: Primer Second Edition, W3C Recommendation 28 October 2004. Available at http://www.w3.org/TR/xmlschema-0/

XML Schema Part 1: Structures Second Edition, W3C Recommendation 28 October 2004.

Available at http://www.w3.org/TR/xmlschema-1/

XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004.

Available at http://www.w3.org/TR/xmlschema-2/

[18] RELAX NG Specification, OASIS Committee Specification, 3 December 2001.

Available at http://www.relaxng.org/spec-20011203.html

[21] News Search Documentation for Yahoo! Search Web Services.

Available at http://developer.yahoo.com/search/news/V1/newsSearch.html

[22] Web Services Transfer (WS-Transfer), Web Services Transfer (WS-Transfer).

Available at http://www.w3.org/Submission/WS-Transfer/

[23] Alex Russel, “Comet: Low Latency Data for the Browser”.

Available at http://alex.dojotoolkit.org/?p=545

[24] URI Template, Internet-Draft, Nov 26, 2007.

Available at http://www.ietf.org/internet-drafts/draft-gregorio-uritemplate-02.txt

[25] Web Services Description Language (WSDL) 1.1, W3C Member Submission, 15 March, 2001

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

[26] Apache Woden Project

Available at http://incubator.apache.org/woden/dev/devprocess.html

[27] WS-I Basic Profile Version 1.1, The Web Services-Interoperability Organization (WS-I), 10 April, 2006

Available at http://www.ws-i.org/Profiles/BasicProfile-1.1.html

## About the Authors

**Toshiro Takase** is a researcher at IBM Research, Tokyo Research Laboratory. Since he joined IBM Japan in 2000, his main research interests have been in the fields of Web Services, and XML performance. He received his B.E. and M.E. in information sciences from the Kyoto University, Japan, in 1998 and 2000.

**Chris Ferris** is an IBM Distinguished Engineer and CTO of Industry Standards in the Software Group Standards Strategy organization. He has been involved in the architecture, design, and engineering of distributed systems for most of his 28+ year career in IT and has been actively engaged in open standards development for XML and Web services since 1999. Chris is former chair of the WS-I Basic Profile Working Group, that is responsible for the development of the WS-I Basic Profile. He currently represents IBM on that WG and serves as editor for the WS-I Basic Profile 1.2 and 2.0. He co-chairs the W3C Web Services Policy WG and serves as chair of the W3C XML Protocols WG. He represents IBM on the OASIS WS-RX TC. He is a former elected member of the OASIS Technical Advisory Board (TAB). Additionally, he is an author and editor of the WS-Reliable Messaging specification and the IBM RAMP profile.

**Ken Ueno** is an Advisory Software Engineer and the expert on WebSphere related performance areas with the IBM Software Services for WebSphere in Yamato Software Development Laboratory. Ken spent last 4 years of his career working on the area of Web Services, Web Services Security, SOA, Web 2.0 and Appliance technology at IBM Tokyo Research Laboratory in Japan. He has many years of experience in WebSphere Application Server development, especially in performance areas. He has been involved in a lot of performance-related client engagements and consulting with the WebSphere products. Ken is co-author of WebSphere V3.5 Handbook and joint translator of Web Services Platform Architecture.

**Satoshi Makino** is a software developer working on Lotus Domino Web Access development team at Yamato Software Development Laboratory in Japan. As of this article's writing, he was a researcher at IBM Tokyo Research Laboratory working on Web 2.0 security.

**Shinya Kawanaka** joined IBM Tokyo Research Laboratory in 2006 as a researcher. He has learned and made researches on XML for several years. He is working for SOA and Web service security in IBM Tokyo Research Laboratory.

**Arthur Ryman** is a Distinguished Engineer in the Rational® Division of IBM. He works at the IBM Toronto Laboratory where he has developed tools such as VisualAge for Java, WebSphere Studio Application Developer, and Rational Application Developer, since 1982. He was a leader of the open source Eclipse Web Tools Platform project and is a coauthor of a recent book on that topic (Addison-Wesley, 2007). He led the development of tools for the early Web service specifications (SOAP, WSDL, UDDI, and WS-I) and later contributed to the WSDL 2.0 specification at W3C and its reference implementation in the open source Apache Woden project. Arthur holds a Ph.D. in mathematics from Oxford University and is a Senior Member of the IEEE. Contact him at ryman@ca.ibm.com