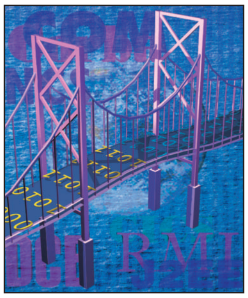

WS-Addressing Metadata

Steve Vinoski • *IONA Technologies*

Last time, I described issues related to using WS-Addressing endpoint references (EPRs) as Web service references (Jan./Feb. 2005, pp. 94–96). Since that column's publication, the W3C's WS-Addressing working group (of which I'm a member) has adopted changes to the draft WS-Addressing specification that largely assuage the concerns I raised — mainly that

- EPRs didn't support multiport services, which are common within multiprotocol enterprise systems; and
- EPRs didn't help with service availability issues, given that they could contain only a single service address.

In this column, I review those concerns and explore how the recent additions to the draft specification resolve them.

EPRs and Metadata

Any Web services endpoint, or more generally, virtually any distributed systems endpoint, incorporates various forms of metadata. For example, an endpoint typically supports some kind of interface, accepts certain forms of data as inputs, and produces the same or different forms of data as outputs. Endpoints might have requirements related to security or transactions, or they might support particular quality-of-service (QoS) guarantees. Although such metadata has traditionally been hard-coded directly into applications, leading to tightly coupled systems, numerous research efforts are currently exploring how applications might dynamically discover and use metadata in practical ways that let them be more loosely coupled and flexible.

The original WS-Addressing member submission to the W3C specified several EPR metadata properties.[1] The `selected-port-type` and `service-port` properties, for example, were intended to convey information about the Web Services Description Language (WSDL) definition associated with a given endpoint. Meanwhile, the `policies` property was intended to be a general container for Web Services Policy Framework (WS-Policy)[2] information associated with an endpoint. Unfortunately, these metadata properties have some problems.

First, WS-Policy is currently only a specification, not an actual standard. When the draft WS-Addressing standard becomes final (currently scheduled to occur sometime in late 2005) it will no longer be able to normatively reference WS-Policy. The WS-Addressing working group realized this shortly after its inception in October 2004 and quickly eliminated the `policies` property from the working draft.

Second, in WSDL, a *port type* specifies a collection of input and output messages — essentially, an interface — that a Web service supports, and a WSDL *binding* associates a port type with a particular protocol and message format (SOAP bindings are commonly used, for example). A WSDL *service* combines a binding (and thus a port type) with an actual service address or location. In the original WS-Addressing specification, the `selected-port-type` property let an EPR creator specify an endpoint's WSDL port type or interface, and the `service-port` property let the creator specify the endpoint's WSDL service. Unfortunately, applications can't use these properties unless they're already aware of the endpoint's WSDL definition. What's more, the properties don't let EPRs include information about multiple WSDL bindings that a Web service might support.

Although these properties enhance certain types of flexibility, they limit others. The fact that the `selected-port-type` and `service-port` properties assume that applications already know the Web service's complete WSDL definition, or can discover it by some other means, implies specific requirements on a Web services system architecture. Specifically, this assumption means that the applications comprising the system must continue to incorporate hard-coded service metadata and accept the brittleness and tight coupling inherent to that approach, or else the system must provide some service or facility that allows for metadata discovery. Last time, I mentioned a specification for one such metadata-retrieval service, WS-MetadataExchange,[3] which specifies messages that let applications request and retrieve service metadata.

Given that the original WS-Addressing specification describes the EPR as "a lightweight and extensible mechanism," we can assume that the authors intended to keep it as minimal as possible. This could explain why they included only the `selected-port-type` and `service-port` WSDL properties. Of course, a truly minimal EPR would consist of nothing more than a service address in the form of a URL, but in that case, wrapping a URL with some XML just for the sake of calling it an EPR wouldn't be worthwhile. Rather, an EPR's ability to convey not only an address but also *endpoint metadata* is what makes it more useful than a simple URL. If this is the primary reason for having an EPR construct in the first place, forcing it to be "lightweight" by specifically limiting what metadata it can carry is somewhat shortsighted. After all, what's considered lightweight for one system might be much too heavy for another or "super lightweight" for a third. Furthermore, the assumption that applications have only two choices when it comes to metadata — hard-coding it or relying on a metadata-retrieval service such as WS-MetadataExchange — eliminates a viable third option: avoiding hard-coded metadata without requiring the introduction of new metadata-retrieval services. This option allows for more flexibility in the EPR itself, such that each application can determine how much metadata its EPRs should carry.

The EPR Metadata Element

In February 2005, several members of the WS-Addressing working group, including IBM, IONA Technologies, Microsoft, Oracle, SAP, and Sun Microsystems, coauthored a proposal (which the full working group subsequently adopted) to alleviate these shortcomings by introducing a general metadata element to the EPR construct. This element, which we often referred to as a "metadata bucket" in our working group discussions, provides an extensible container for endpoint metadata.

This metadata element effectively replaces the `policies` property, which the working group dropped because it was based on a nonstandard specification. Assuming that its authors eventually submit the WS-Policy specification for standardization, they'll now be able to specify precisely how policy metadata should appear within an EPR's metadata container. In general, the metadata element provides an extensibility point that other standards can use to add metadata to the EPR.

More important, though, is the fact that the metadata element breaks the original artificial limitations on how much WSDL-related metadata can appear within an EPR. An EPR creator can now, when appropriate, include a Web service's entire WSDL definition within the metadata container. A WSDL definition typically contains

- type definitions for data sent to and returned from a service,
- message and operation definitions,
- port type definitions, which collect operations into endpoint interfaces,
- bindings, which tie port types to concrete protocols and message formats,
- ports, which combine bindings and service endpoint communication details, and
- a service definition consisting of one or more ports.

This approach has several benefits. First, it opens up the third option for metadata retrieval without disallowing the other two. This means services can export their own metadata directly in their EPRs, which is especially important for legacy applications that developers can't modify and redeploy just to support a solution such as WS-MetadataExchange. Instead, developers can write WSDL definitions to describe those services as they already exist, and then simply embed that WSDL metadata within the services' EPRs. This lets providers make those applications available to new service consumers without any modifications.

An EPR's ability to convey not only an address but also *endpoint metadata* is what makes it more useful than a simple URL.

Another benefit is that the metadata element lets services advertise multibinding access capabilities. To maximize integration possibilities, services often make themselves available over multiple means of access. For example, a developer might want to take a service originally built against some middleware architecture and update it so that it's also available via SOAP over HTTP. This allows original clients to continue using the legacy

```
<wsa:EndpointReference
  xmlns:wsa="http://www.w3.org/2005/03/addressing">
  <wsa:Address>http://www.somecompany.com/my-service/svc/</wsa:Address>
  <wsa:Metadata>
    <wsdl11:definitions targetNamespace="http://www.somecompany.com/"
      xmlns:wsdl11="http://schemas.xmlsoap.org/wsdl/"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns:iiop="http://www.omg.org/iiop/"
      xmlns:scomp="http://www.somecompany.com/">
      <wsdl11:import namespace="http://www.somecompany.com/"
        location="http://www.somecompany.com/my-service/my.wsdl"/>
      <wsdl11:service name="MyService">
        <wsdl11:port name="port1" binding="scomp:MyServiceSoapBinding">
          <soap:address
            location="http://www.somecompany.com/my-service/svc/"/>
        </wsdl11:port>
        <wsdl11:port name="port2" binding="scomp:MyServiceIIOPBinding">
          <iiop:address location="IOR:…"/>
        </wsdl11:port>
      </wsdl11:service>
    </wsdl11:definitions>
  </wsa:Metadata>
</wsa:EndpointReference>
```

Figure 1. Example endpoint reference. This EPR contains multiport WSDL metadata, specifying multiple mechanisms for accessing a single service.

application unchanged via its original protocol and message format, even as new Web services client applications start communicating with it via SOAP messages. The original EPR design provided no way to express that consumers could access a single service via multiple message formats and protocols. However, the revised EPR definition allows for this because service providers can describe in WSDL all the message formats and protocols that a service supports and include them within the EPR metadata element. Figure 1 (next page) illustrates what a multiport EPR might look like.

This example shows an EPR for a Web service called `MyService`. It specifies a service address using the `wsa:Address` element and also includes a `wsa:Metadata` element. Within the latter, a full WSDL definition specifies two ports for the service – one using a SOAP binding, and the other using a hypothetical binding for the Corba Internet Inter-ORB Protocol (IIOP). (For brevity, I don't show the bindings and port types, but instead assume they are imported via the `wsdl11:import` directive.) Assuming it understands both SOAP and IIOP, an application receiving this EPR could choose to use either protocol to interact with the service.

In addition to providing multiple means of accessing a service, being able to specify multiple ports per service might also help us solve availability problems. If a service EPR included information for accessing the service over multiple ports, an application that failed to reach the service over one port might succeed using a different one. Of course, this isn't the only way to address availability concerns, but it's always nice to have another option.

Using the metadata element to specify a service's WSDL definition doesn't introduce any significant new interoperability problems. Because it relies on WSDL, this approach is effectively just as interoperable as WSDL itself. And for those developers who don't care to use WSDL, the metadata element is completely optional; it need not appear at all within an EPR.

The Debate Continues

Although achieving agreement on this proposal in the WS-Addressing working group wasn't difficult, it hasn't helped resolve a debate regarding Web services that's continued unabated for the past several years. This debate, between Web services proponents and proponents of representational state transfer (REST)[4] – the architecture underlying the World Wide Web – is about the different approaches to Web services that each camp embraces.

One of the most fundamental disagreements revolves around Web service interfaces. The EPR metadata element allows, among other things, a Web service to include a definition of its interface in its own reference. REST proponents, however, argue that such interface information is unnecessary. Instead, they recommend using a *uniform interface* for all services. They base this recommendation on the success of the Web, which has scaled to its present proportions, in part, because all Web servers support the same interface – the HTTP verbs `GET`, `PUT`, `POST`, and `DELETE`. Nonuniform interfaces don't scale as well because of the higher degree of coupling they introduce between systems.

Web services proponents accept uniform interfaces' superior scalability but argue that they aren't ideal for all situations. Within enterprise intranets, for example, tighter coupling isn't necessarily unworkable because the same organizations often own both the service applications and the applications that use them. Done correctly, the tighter coupling of a specialized interface can lead to higher system performance. Even Fielding's REST thesis acknowledges that "the REST interface is designed to be efficient for large-grain hypermedia data transfer, optimizing for the common case of the Web, but resulting in an interface that is not optimal for other forms of architectural interaction."[4]

Depending on the scale and usage of the services under debate, both sides are right to a certain extent. Given that a significant percentage of Web services usage is occurring within the enterprise, however, the need to support WSDL-style interfaces won't disappear anytime soon. The EPR metadata element neither hinders nor improves the applicability of non-REST Web services to very large-scale problems, but it definitely enhances Web services' applicability to thorny integration problems in today's heterogeneous enterprises.

References

1. D. Box et al., "Web Services Addressing (WS-Addressing)," W3C member submission, Aug. 2004; www.w3.org/submission/2004/subm-ws-addressing-20040810.
2. S. Bajaj et al., *Web Services Policy Framework (WS-Policy)*, joint specification by IBM, BEA Systems, Microsoft, SAP, Sonic Software, and VeriSign, Sept. 2004; http://specs.xmlsoap.org/ws/2004/09/policy/ws-policy.pdf.
3. K. Ballinger et al., *Web Services Metadata Exchange (WS-MetadataExchange)*, joint specification by BEA Systems, Computer Associates, IBM, Microsoft, SAP AG, Sun Microsystems, and WebMethods, Sept. 2004; http://www-106.ibm.com/developerworks/library/specification/ws-mex.
4. R.T. Fielding, "Architectural Styles and the Design of Network-Based Software Architectures," doctoral dissertation, Dept. of Computer Science, Univ. of California, Irvine, 2000.

Steve Vinoski is chief engineer of product innovation for IONA Technologies. He's been involved in middleware for more than 17 years. Vinoski is coauthor of *Advanced Corba Programming with C++* (Addison Wesley Longman, 1999), and he has helped develop middleware standards for the Object Management Group (OMG) and the World Wide Web Consortium (W3C). Contact him at vinoski@ieee.org.

ADVERTISER / PRODUCT INDEX MAY/JUNE 2005

Advertiser	Page Number
EFF	9
JavaOne	**Cover 4**
Open Source Initiative	9

Boldface denotes advertisements in this issue.

Advertising Personnel

Marion Delaney
IEEE Media, Advertising Director
Phone: +1 212 419 7766
Fax: +1 212 419 7589
Email: md.ieeemedia@ieee.org

Marian Anderson
Advertising Coordinator
Phone: +1 714 821 8380
Fax: +1 714 821 4010
Email: manderson@computer.org

Sandy Brown
IEEE Computer Society,
Business Development Manager
Phone: +1 714 821 8380
Fax: +1 714 821 4010
Email: sb.ieeemedia@ieee.org

Advertising Sales Representatives

Mid Atlantic (product/recruitment)
Dawn Becker
Phone: +1 732 772 0160
Fax: +1 732 772 0161
Email: db.ieeemedia@ieee.org

New England (product)
Jody Estabrook
Phone: +1 978 244 0192
Fax: +1 978 244 0103
Email: je.ieeemedia@ieee.org

New England (recruitment)
Robert Zwick
Phone: +1 212 419 7765
Fax: +1 212 419 7570
Email: r.zwick@ieee.org

Connecticut (product)
Stan Greenfield
Phone: +1 203 938 2418
Fax: +1 203 938 3211
Email: greenco@optonline.net

Midwest (product)
Dave Jones
Phone: +1 708 442 5633
Fax: +1 708 442 7620
Email: dj.ieeemedia@ieee.org
Will Hamilton
Phone: +1 269 381 2156
Fax: +1 269 381 2556
Email: wh.ieeemedia@ieee.org
Joe DiNardo
Phone: +1 440 248 2456
Fax: +1 440 248 2594
Email: jd.ieeemedia@ieee.org

Southeast (recruitment)
Thomas M. Flynn
Phone: +1 770 645 2944
Fax: +1 770 993 4423
Email: flynntom@mindspring.com

Southeast (product)
Bill Holland
Phone: +1 770 435 6549
Fax: +1 770 435 0243
Email: hollandwfh@yahoo.com

Midwest/Southwest (recruitment)
Darcy Giovingo
Phone: +1 847 498-4520
Fax: +1 847 498-5911
Email: dg.ieeemedia@ieee.org

Southwest (product)
Josh Mayer
Phone: +1 972 423 5507
Fax: +1 972 423 6858
Email: jm.ieeemedia@ieee.org

Northwest (product)
Peter D. Scott
Phone: +1 415 421-7950
Fax: +1 415 398-4156
Email: peterd@pscottassoc.com

Southern CA (product)
Marshall Rubin
Phone: +1 818 888 2407
Fax: +1 818 888 4907
Email: mr.ieeemedia@ieee.org

Northwest/Southern CA (recruitment)
Tim Matteson
Phone: +1 310 836 4064
Fax: +1 310 836 4067
Email: tm.ieeemedia@ieee.org

Japan
Tim Matteson
Phone: +1 310 836 4064
Fax: +1 310 836 4067
Email: tm.ieeemedia@ieee.org

Europe (product)
Hilary Turnbull
Phone: +44 1875 825700
Fax: +44 1875 825701
Email: impress@impressmedia.com