Published on ***WSO2 Oxygen Tank*** (http://wso2.org)

WS-Addressing in Action

By ***samisa abeysinghe***
Created ***2007-09-03 22:02***

Applies To

Apache Axis2/C	1.1.0
Environment	Any

Introduction

Web Services Addressing provides mechanisms to address Web services in a transport neutral manner. In others words, irrespective of whether you are using HTTP, SMTP, XMPP or any other transport to communicate with a service, WS-Addressing allows the messages to be uniquely addressed to Web service endpoints, independent of any transport characteristics. Being transport independent allows a given service to be consumed using the same syntax and semantics using different transports.

WS-Addressing can be considered as one of the corner stones of the WS-* stack, where many other Web services specifications make use of it. Hence it is rudimentary that any Web services engine that supports or plans to support the full Web services stack must have WS-Addressing support implemented.

Specifications and Status

In May 2006, the W3C Web Services Addressing working group [1 [1]] made available the version 1.0 recommendation [2 [1]] of WS-Addressing. Prior to that, they released a submission version [3 [1]] of the recommendation in August 2004. Version 1.0 has many enhancements over the submission version. There could be some service implementations out there that still use the submission version. However 1.0 can be considered mainstream. There has been an interop testing round based on a well defined test suite [4 [1]] and the results are available online.

Constructs of WS-addressing

There are two basic constructs in WS-Addressing:

- Endpoint references
- Message addressing properties

An endpoint is an entity to which a Web service message can be addressed. Endpoint references convey information required to address a Web service endpoint. The Endpoint References [5 [1]] section of the WS-

Addressing specification defines the information model and syntax of an endpoint reference.

The following is an example endpoint reference that references the endpoint at the URI "http://wso2.org/addressing/sample":

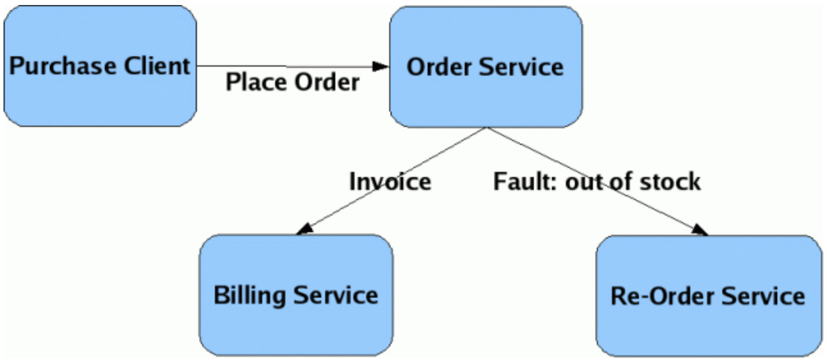
```
<wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing">
      <wsa:Address>http://wso2.org/addressing/sample</wsa:Address>
</wsa:EndpointReference>
```

Message addressing properties are used to convey various end-to-end message characteristics. The Message Addressing Properties section [6 [1]] of the WS-Addressing specification defines the information model and syntax of message addressing properties. The following table summarizes the basic set of message addressing properties. It is important to note that, out of these message addressing properties, only Action is mandatory; the others are optional.

	Message Addressing Property	**Description**
wsa:To	Provides the destination URI. This property indicates where the message will be sent to.	
wsa:From	Provides the source endpoint reference. Indicates where the message came from.	
wsa:ReplyTo	Provides the reply endpoint reference. Indicates where the reply for the request message should be sent to. If this is not present, usually the reply will be sent back to the endpoint where the request came from.	
wsa:FaultTo	Provides the fault endpoint reference. Indicates where the fault message should be sent to in case there is a fault. If this is not present, usually the fault will be sent back to the endpoint where the request came from	
wsa:Action	Conveys the action related to a message. For example, the wsa:Action property can be used to identify the operation to be invoked upon receiving a request message. This property must be provided in message addressing properties of a message.	
wsa:MessageID	Conveys the ID of a message that can be used to uniquely identify a message.	
wsa:RelatesTo	Conveys the message ID of a related message, along with the relationship type.	

Example: Order Processing

Now that we have discussed the basics of WS-Addressing, let's consider an example where WS-Addressing is in action. The following diagram portrays the general idea in this example.



The purchase client places an order to the order processing service. If the order is processed successfully, the invoice corresponding to the processed order will be sent to the billing service, which in turn will take the necessary actions. In case the order processing fails due to some error, the order would be queued with the re-order service, where the order will persist for later processing.

When applying WS-Addressing for this scenario, the purchase client will have to set the various message properties appropriately. The wsa:To will be set a URI to point to the order processing service. The wsa:ReplyTo property will be set to direct the reply for the placed order to the billing service and the wsa:FaultTo property will be set to point to the re-order service in case of an error.

The following is an example of a message with the WS-Addressing message properties, set in the header of the SOAP message corresponding to the place order request.

```
<soapenv:Envelope xmlns:soapenv='http://www.w3.org/2003/05/soap-envelope'>
<soapenv:Header xmlns:wsa='http://www.w3.org/2005/08/addressing'>
<wsa:To>http://localhost:9090/axis2/services/order_service</wsa:To>
<wsa:Action>http://wso2.org/wsf/c/addr/order</wsa:Action>
<wsa:ReplyTo>
<wsa:Address>http://localhost:9090/axis2/services/billing_service</wsa:Address>
</wsa:ReplyTo>
<wsa:FaultTo>
<wsa:Address>http://localhost:9090/axis2/services/reorder_service</wsa:Address>
</wsa:FaultTo>
<wsa:MessageID>a4dfb94a-593b-1dc1-36d2-000000000000</wsa:MessageID>
</soapenv:Header>
<soapenv:Body>
<ns1:order xmlns:ns1='http://wso2.org/wsf/c/addr/sample'>
<item>paper</item>
<quantity>100</quantity>
</ns1:order>
</soapenv:Body>
</soapenv:Envelope>
```

WS-Addressing with Axis2/C

Apache Axis2/C has full WS-Addressing support implemented. Apache Axis2/C supports both the submission version and the 1.0 version. The default version is 1.0. In the current implementation of Axis2/C, you have to engage the addressing module in the axis2.xml configuration file in order to use WS-Addressing. This can be done by adding the following line to the axis2.xml file.

```
<module ref="addressing"/>
```

Once the above line is added to the configuration file, the addressing module is engaged and ready to use.

WS-Addressing on the Server Side

On the server side, when the addressing module is engaged, there is a requirement for WS-Addressing to be activated. That is the incoming message has to use addressing. When this requirement is met, the addressing handlers will be invoked and the server will respond with addressing message properties in the response message.

In line with the WS-Addressing specification, if the incoming message has a wsa:Action header, the server side considers the incoming message to use addressing. This is justified because, as per the specification, the wsa:Action message property is mandatory.

The Axis2/C engine has provision for the user to map wsa:Action to the operation to be invoked. This is one of the common use cases for the wsa:Action message property. This can be done by configuring the service to map addressing actions to the operations to be invoked, in the services.xml file. The following example shows you how to configure the wsa:Action in services.xml.

```
<service name="order_service">
<parameter name="ServiceClass" locked="xsd:false">order_service</parameter>
<description>
This is the order service.
</description>
<operation name="order">
<parameter name="wsamapping" >http://wso2.org/wsf/c/addr/order</parameter>
</operation>
</service>
```

In the above example, the parameter element with the name attribute value of wsamapping specifies the wsa:Action for the operation 'order'.

The Axis2/C engine has a dispatcher named "Addressing based dispatcher" that uses the wsa:Action in the incoming message, and try to identify the operation to which the request has to be dispatched.

Specifying the wsa:Action in the services.xml file is the only requirement to use WS-Addressing in a user implemented service. Rest of the WS-Addressing related requirements such as sending the reply or the fault to the correct endpoint and generating and setting message IDs etc. are handled by the addressing module.

WS-Addressing on the Client Side

In order to use addressing on the client side, there are two pre-requisites. First the addressing module needs to be engaged, and second, the SOAP action to be used must be specified.

There are two ways to engage addressing on the client side. One way is to use the configuration file, and add the addressing module reference to the axis2.xml file.
The other way to engage addressing is do it programmatically, using the `axis2_svc_client_engage_module` function of the service client API.

```
axis2_svc_client_engage_module(svc_client, env, AXIS2_MODULE_ADDRESSING);
```

On the client side, the WS-Addressing action to be used can be set with options, using the function `axis2_options_set_action`

```
axis2_options_set_action(options, env, http://wso2.org/wsf/c/addr/order);
```

Engaging the addressing module and setting the WS-Addressing action are the compulsory pre-requisites for using addressing with Apache Axis2/C on the client side. The sample, samples/user_guide/clients/echo_blocking_addr.c demonstrate these basic requirements, and also the client sample works with the echo sample.

You can also set the FaultTo and ReplyTo endpoints using the options. You can use, `axis2_options_set_reply_to` and `axis2_options_get_fault_to` functions to set the endpoints for those headers.
Note that irrespective of whether you are using addressing or not, you have to always use `axis2_options_set_to`, to set the target endpoint to be consumed by the client.
Also note that you do not have to set the MessageID SOAP header when using WS-Addressing with Apache Axis2/C. As the MessageID SOAP header is mandatory when using WS-Addressing, and it also has to be unique across all calls, the Axis2/C engine takes care of setting a unique MessageID for each and every message when WS-Addressing is in use.

Conclusion

WS-Addressing is useful when addressing Web services messages and directing those messages to desired destinations. As it was explained in this article, WS-Addressing features can be leveraged to model message exchanges that is in sync with the business requirements of the organizations. The implementation that comes with Apache Axis2/C Web services engine [2] is a complete implementation of WS-Addressing that has been proved to interoperate with Java implementations a well as Microsoft .NET.

References

Apache Axis2/C [2]

[1] W3C Web Services Addressing Working Group [3]
[2] Web Services Addressing 1.0 [4]
[3] Web Services Addressing Submission [5]
[4] Web Services Addressing 1.0 Test Suite [6]
[5] WS-Addressing, Endpoint References [7]
[6] WS-Addressing, Message Addressing Properties [8]

Author

Samisa Abeysinghe - Software Architect WSO2 Inc. samisa at wso2 dot com.

Articles Intermediate

Source URL: http://wso2.org/library/2605

Links:
[1] http://wso2.org/library/2605#ref
[2] http://ws.apache.org/axis2/c/
[3] http://www.w3.org/2002/ws/addr/
[4] http://www.w3.org/TR/ws-addr-core/
[5] http://www.w3.org/Submission/ws-addressing/
[6] http://www.w3.org/2002/ws/addr/testsuite/
[7] http://www.w3.org/TR/ws-addr-core/#eprs
[8] http://www.w3.org/TR/ws-addr-core/#msgaddrprops