Position Paper: W3C Workshop on Constraints and Capabilities for Web Services

Jan Alexander, Systinet
Toufic Boubez, Layer 7 Technologies
Luc Clément, Systinet

Introduction

One of the fundamental concepts in Service Oriented Architectures is that of loose coupling. The goal of loose coupling is to eliminate unnecessary dependencies between a service and its consumers at design time and make late binding between them possible. While this is harder to achieve for application semantics, it should be relatively more achievable for service deployment parameters: In this context, what a service does is not a concern - how it is made available is.

There are two kinds of information that need to be conveyed in order for a client to access a service successfully, both important, but fundamentally different: WSDL conveys a programmatic API, the signature or the semantics of the service that is being described. That information alone is necessary but not sufficient. There are numerous other aspects of a Web service that need to be communicated to a Web service consumer either through a process of introspection or discovery in order for it to successfully access the service: These include credentialing mechanisms, security parameters such as encryption or signature of the message or parts of it, preferred transport mechanisms, reliability requirements, transactional requirements. Some other elements are informational in nature, and help the client decide whether to use the service: privacy policies, logging policies, etc. Finally, some information might be required for deploying the service, but not public in nature, such as information about endpoint addressing for routing purposes. In essence, while WSDL deals mostly with the `<Body>` element of the SOAP envelope, there is no metadata language to deal with the `<Header>` element, which is the domain of the information required for a true vision of "secure, reliable, transacted Web services" [SRTWS]. By the same token, this metadata needs to be discoverable to allow an enterprise to build dynamic business services.

We see this set of metadata, which we'll call *Policy*, as an essential component and requirement for the governance of Web services deployments. Policy artifacts have to be exchanged between parties and enacted both on the service and the client sides; for this exchange to occur, discovery of the artifacts needs to be provided for. Currently and historically, this has been done through out-of-band communications, through non specific documents such as text documents and through coding. It is extremely critical therefore that the software community agrees on establishing a standard framework for expressing, discovering, and exchanging this metadata.

Use Cases

This section will discuss the use case proposed in the Call for Participation (http://www.w3.org/2004/06/ws-cc-cfp.html) and propose some other use cases of interest to be discussed at the workshop.

CFP Proposed Use Case

A Web service wishes to stipulate that clients are required to support a reliable messaging protocol, and encrypt a specific header with WS-Security using a X.509 or user name security token in order to send an acceptable request message. Furthermore, the service has a P3P policy associated with its operations. Such constraints and capabilities might be associated with the Web service via a SOAP header or a WSDL file.

Discussion
One issue with the proposed use case is the ambiguity around the terms "a reliable messaging protocol". Without getting into issues of semantics that are sure to be described in other papers, we believe that the use case should have been stated and conceptualized more appropriately as the service proposing a list of acceptable protocols available to the client. An acceptable alternative would be to state unambiguously what "a reliable messaging protocol" is and point through a reference to a list of protocols that have already been defined by some standardization body like the W3C. In either case, the service (or its developer) could generate an XML policy document of the following form (Note: while this position paper is not proposing any particular syntax for the metadata, a format similar to the WS-Policy framework [WSP] in the example):

```
<Policy>
   <AND>
      <OR>
         <Reliability>w3c:SomeProtocol</Reliability>
         <Reliability>w3c:SomeOtherProtocol</Reliability>
      </OR>
      <OR>
         <Encrypt>
            <XpathExpression xpathExpressionValue="included">
               <Expression stringValue="SomeXPathExpression"/>
            </XpathExpression>
            <wsse:SecurityToken>
               <wsse:TokenType>wsse:X509v3</wsse:TokenType>
            </wsse:SecurityToken>
         </Encrypt>
         <Encrypt>
            <XpathExpression xpathExpressionValue="included">
               <Expression stringValue="SomeXPathExpression"/>
            </XpathExpression>
            <wsse:SecurityToken>
               <wsse:TokenType>wsse:Username</wsse:TokenType>
            </wsse:SecurityToken>
         </Encrypt>
      </OR>
      <P3P policyref="SomeURL">
   </AND>
</Policy>
```

Though the use case identifies the need for expressing constraints and capabilities, along with the need to exchange this metadata, it does not go far enough and state the additional requirement to make this information discoverable. By making policy discoverable a consumer can select for example between otherwise identical Web services the instance that exhibits the constraints for which the consumer has the capability of supporting.

The use case as presented does not recognize the dynamic nature of SOA-based deployments nor does it enact what is required to govern an enterprise's SOA. Distributing constraints and capabilities in the form of policy expressed exclusively at the end points is not scalable and manageable for an enterprise. While the end point typically represents the policy enforcement point motivating the need for the ability to support the means of exchanging this policy with consumers and the author of the policy, the policy must be discoverable for consumers. The inability to support discoverability of policy by consumers significantly reduces the value of the proposition. Furthermore value for the enterprise in terms of governance would be greatly enhanced by supporting the means of effecting a global policy change that can be pushed to or pulled by end points; for this to be scalable a Web services registry is required to support the need of discovering Web services and their policies.

Therefore, the authors propose a broadening of the use case to encompass discoverability both from the perspective of the consumer as well as the producer that enacts policy asserted by the enterprise. The use case and the context in which it is presented needs to leverage registry not only to promote visibility of the Web services and their artifacts but also as a key SOA governance enabler, this given the highly variable nature of business and IT, and the commensurate needs to adapt to change.

We are therefore supplementing the use case by providing a description below that leverages registry – namely UDDI – for this purpose. That said, the XML policy document presented above is thus attached to the UDDI entry for the service in a manner consistent with the mapping described in the *Using WSDL in a UDDI Registry, Version 2.0* [WSDL-UDDI] Technical Note. Additionally, a reference to this document may be included in the WSDL document as an extension to the `<Service>` or `<Operation>` elements (see discussion in Open Issues below). Or the document can be made available by the endpoint itself by means similar to the WS-MetadaExchange specification [WSME]. Note that this implies that the policy can be evaluated as a logical expression, with informational elements always returning `TRUE`, and no preference to the ordering of the elements. Clients are assumed to be in conformance to the policy if the expression evaluates to `TRUE` (see more discussion in Open Issues below).

Exploring some additional use cases is also of value if the solution is to address a broad set of requirements.

Differentiated Access Use Case

A Web service wishes to stipulate that clients from within a particular group of identities are required to use SSL and provide credentials using HTTP Basic-Auth and their requests will not be logged, whereas clients belonging to another group of identities have to use the WS-Security user name security token for authentication, and their requests will be logged.

Refresh/Fault Use Case

Typically, policies will probably be cached at the client and re-used. A Web service may wish to stipulate that its usage policy has to be refreshed at each invocation, or it may wish to provide its clients with an expiration timestamp for the policy document, forcing a refresh. Additionally, specific policy related fault codes may be provided. These faults are different in purpose and usage from the application specific faults that can be specified in a WSDL document.

In addition to this, corporate governance may need to push notification of changes to enact a policy update across multiple services rather than expecting a client to refresh cached information or process a fault.

Private Policy Use Case

In addition to the publicly stated requirements and capabilities expressed in the previous use cases (encryption, transport, etc), the administrator of a Web service wishes to specify different endpoints and different logging mechanisms for different clients, based on their identities. These policy details are irrelevant to the client, and are of a private nature to the Web service and therefore are not to be conveyed to the client. They have to be stored in the same overall policy document that is attached to the Web service.

Open Issues

Processing model

It is unclear without more use cases whether the policy document should impose a processing model on the client and the service. In absence of a processing model, the policy expression can be viewed as a logical expression to be evaluated, with access to the service contingent on the expression evaluating to `TRUE` (assuming the credentials are valid and have access to the service, etc). This would imply logical operators such as `AND`, `OR` and `NOT`. A processing model, on the other hand, imposes some additional complexities but allows more flexibility in expressing policy elements. In a processing model environment, ordering of policy elements will matter, so will side effects of evaluating these elements. This will allow the introduction of conditionals and branching, allowing content-based policy models to be expressed. Another aspect of a processing model is to enable the creating of policies that can alter the content of the message and be acted upon by intermediaries.

Scope (private/public)

We believe that there are different kinds of policy information that need to be stated, some of it to be made public and some to stay private, and that there needs to be a mechanism to differentiate between the two. For example, policy information relating to corporate security policies, access control lists, audit logging rules, and physical endpoint addressing and routing has a different audience than the client. It is therefore sometimes desirable or even necessary to write policy elements that are not meant to be made public, and to label them as such. Both public and private policies created on one system need to be exchangeable and implemented on another platform.

Attachment and Discovery

Since we already have a mechanism for registering and discovering other types of metadata for Web services in UDDI, it is therefore logical and indeed required for an enterprise governance perspective that we use the same mechanism to register policy metadata and make it discoverable. The example of handling WSDL in UDDI should be examined and the policy mechanism such as proposed by WS-PolicyAttachments should be investigated.

Policy distribution and enforcement

Enterprise configurability and adaptability needs of SOA advocates in favour of the need to provide the means of supporting automatically configured frameworks based on the received policy, requirement of searchable policy repository integrated with the registry (UDDI), policy dispersing and enforcement from the centrally managed repository. These things would be necessary to make the constraints based configuration and management work; as previously stated, distribution of this information solely at the end points will not scale for most enterprises.

Interoperability

One of the undeniable trends in the marketplace is the emergence of both Web services security and Web services management categories, with vendors in both categories having some claim on managing policies associated with services in their realm. It is important therefore that any proposed standard accommodate both categories and allow for interoperability between them.

Negotiation

Although most of the emphasis on constraints and capabilities for Web services has been focused on the services, it is just as important to keep in mind the constraints and capabilities of service consumers. A mechanism that allows consumers to discover a service's policies and negotiate terms acceptable to both parties would further enhance the loose coupling in SOA deployments.

About the authors

Jan Alexander: Jan Alexander is Chief Architect at Systinet Corporation. At Systinet Jan has been instrumental in designing the Systinet Server and Systinet Registry Web services products. He has been active at standards efforts at W3C, OASIS and WS-I.
Toufic Boubez: Toufic Boubez is the CTO of Layer 7 Technologies. Toufic has been involved in Web services since 1999 when he was the lead Web services architect for IBM and for the IBM Web services toolkit (WSTK). He's a co-author of the UDDI V1 specification, and of the WS-Trust and WS-SecureConversation specifications.
Luc Clément: Luc Clément is a senior program manager at Systinet. He is a former Microsoft UDDI program manager, the general program manager for UDDI.org, an established UDDI expert, and co-author and editor of the UDDI specification.

Layer 7 Technologies: Layer 7 Technologies has a very strong and critical interest in policy standards for Web services. We have had released products in that space since July 2003. In developing the products, and more importantly during our customer deployments and interactions, we have encountered some important issues and gained some experience that we hope we can contribute to the discussion.

Systinet: Systinet sees UDDI and policy as a critical component of SOAs as it enables businesses to standardize the way they organize, discover, reuse, manage and govern an SOA built using Web services. By adopting the means of expressing, discovering and exchanging policy, enterprises will attain what they have for so long longed for: enterprise-wide insight, control and economic leverage of its SOA business assets that, increasingly over time, will be realized and used by customers, suppliers, products, and employees as dynamic business services.

References

[SRTWS] D. F. Ferguson, T. Storey, B. Lovering, J. Shewchuk. "Secure, Reliable, Transacted Web Services". http://www-106.ibm.com/developerworks/webservices/library/ws-securtrans/
[WSDL-UDDI] “Using WSDL in a UDDI Registry, Version 2.0” - http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm
[WSME] "Web Services Metadata Exchange (WS-MetadataExchange)". http://ftpna2.bea.com/pub/downloads/WS-MetadataExchange.pdf
[WSP] D. Box, F. Curbera, M. Hondo, C. Kaler, D. Langworthy, A. Nadalin, N. Nagaratnam, M. Nottingham, C. von Riegen, J. Shewchuk. "Web Services Policy Framework (WS-Policy)" http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-policy.asp