IBM Position Paper for W3C Constraints/Capabilities

Statement of Goals

A survey of the terms and technologies cited in this workshop call can produce various interpretations of the goals of this workshop. The challenges before us, as a community attempting to address business requirements in the 21st century are significant and wide ranging. We are always interested in participating in discussion of these challenges, but also believe discussion needs to be balanced by a clear set of goals. IBM believes the following near-term issues must be resolved for web services to continue their successful evolution:

- The communication of metadata related to conditions and constraints expressed between and about Web service endpoints (between requestor and provider, between service and hosting environment).
- The declarative description of domain specific conditions and constraints such as those for enterprise privacy.

Consequently IBM's goals in submitting this paper for this workshop are:

- To discuss the advantages of WS-Policy as a technology for Web services polices and determine the interest level in this type of activity at the W3C
- To acknowledge the ongoing efforts that we believe need focus in the next several years.

In addition, we believe that the set of issues to emerge in the next 3 to 5 years will include:

- Business modeling as an effective technology for composing Web services into solutions
- Self-managing systems as a way to reduce the administrative costs associated with increasing system complexity.

Introduction

One of the remaining challenges in the evolution of Web services technology is the expression, exchange and processing of the conditions governing the interactions between Web service endpoints. WSDL and XML Schema provide metadata describing *what* a service does by defining the business interface (that is, business operations like openAccount, processDeposit, etc.) for a Web service. *How* the service implements the interface and what it expects or provides to requestors and what is expected or provided in a hosting environment, is equally important. Is the service transactional? Must callers sign messages? To achieve the promise of a Service-Oriented Architecture it is important to extend the current Web service interface and message definitions to include the expression of constraints and conditions on the Web service.

While it is possible to use XML and WSDL extensibility to achieve some of these goals, it is preferable to provide a common framework for Web services constraints and conditions that allows a clear articulation of the available options. The common framework must enable constraints and conditions defined for various domains/disciplines to be combined together consistently so web service providers and consumers are not burdened with many domain-specific mechanisms. A common framework provides support for determining valid intersections of constraints and conditions when multiple choices are possible.

It is also important that Web services rely on a declarative model for conditions and constraints. The program logic implementing a Web service could explicitly implement the conditions/constraints. A declarative model factors the conditions/constraints out of business logic, allowing for automated implementation by middleware and operating systems. Using a declarative model enables a better user experience and better reuse of application code by the organizations that deploy and support Web Service instances.

There are many valid approaches to implementing a Web service's business functions. Programming languages, BPEL4WS and business rules are some examples. There are many approaches to rule based programming, which include decision tables, decision trees, if … then … sets or inference engines. The IBM approach is to view business rules as one of many ways to implement a Web service. To support customization and tailoring, designers can use a strategy pattern based on Web services. One service, A, provides the intrinsic implementation. Service A delegates onto another customizing service, service C, for dynamic, changing computations. Rules are one way to implement C.

It is IBM's belief that there is a natural separation of concerns with regard to classification of the varied set of information often referred to as "policy". Web service *policies* are consumable declarative expressions of conditions and constraints for a particular Web service. The goal is for the Web service endpoints (requestor and provider) to communicate any requirements or agreements that affect either endpoint when providing a web service. There may also be policies that a Web service implementation declares to express requirements on a hosting environment ("the container").

For Web services, the W3C has played a fundamental role in standardizing the core components of WSDL and SOAP and this has allowed the industry to achieve a high degree of interoperability both at design time (tool) and runtime. IBM believes there is a natural extension to this model in the form of WS-Policy and WS-Policy Attachments[1] in the near term.

WS-Policy [WS-Policy] is an extensible framework for defining Web services policy. WS-PolicyAttachments [WS-PolicyAttachment] offers a flexible way of associating policy expressions with existing (and future) Web services artifacts. The WS-Policy specification defines a common framework for services to annotate their interface definitions to describe their service assurance qualities and requirements in the form of a machine-readable expression containing combinations of individual assertions. The WS-Policy framework also allows for algorithms that determine which concrete policies to apply when the requestor, provider and container support multiple options.

Web Services Policy in the Web Services Protocol Stack

There is an increasing body of specifications and standards that improve the fidelity and quality of Web service interactions. These specifications require the ability to annotate Web services with metadata describing valid and required options.

IBM has participated with other industry partners in the development of these Web services specifications, among them Web Services Security, Web Services Reliable Messaging, Web Services Addressing, Web Services Transactions, Web Services Coordination, and the Business Process Execution Language for Web Services (BPEL4WS). These specifications introduce three kinds of artifacts:

- Architected function enablement (infrastructure) specific headers that augment messages with function specific information, e.g. transaction context, message sequence numbers.
- Distinguished Web services that participate in the implementation of the enablement functions, e.g. Coordinator, Trust Authority.
- Endpoint protocols for implementing the enablement functions, e.g. message retransmission, transaction protocol sets.

The result is significant step forward towards a robust secure, reliable, and transactional Web services infrastructure. But there is more to be done.

In this context of rich interactions and multiple service assurance protocols it is important for a Web service to be able to indicate in a consistent and declarative way what it is capable of supporting, including the types of protocols that it supports, as well as what requirements it places on potential requesters. The service implementation should also be able to document requirements on a hosting environment since much of the implementation of the protocols and functions occurs in middleware and OS functions supporting the service's implementation. Thus, there must be a way to describe the constraints/conditions derived from the environment hosting a particular web service.

Modeling these properties in separately standardized discipline specific XML languages allows a developer and deployer to incrementally augment the sophistication of a Web service description. It allows basic services to work today and it allows for the evolution of the sophistication of these services over time. It also allows the partitioning

---

[1] http://www.ibm.com/developerworks/library/specification/ws-polfram/, http://www.ibm.com/developerworks/library/specification/ws-polatt/

of the work into units where the subject matter experts can proceed on expressing their discipline requirements independently thus allowing for parallelism of development.

WS-Policy proposes a framework that extends the features already provided through WSDL, including tool interoperability but also service discovery. More refined service descriptions, qualified by specific WS-policies, support more accurate discovery of compatible services. In a service registry (such as the UDDI registry), queries of WS-Policy described services enable the retrieval of services supporting the appropriate policies in addition to the correct business interface. For example a query may request all services that support the purchaseOrder WSDL interface (port type) *and also* use Kerberos for authentication and have an explicitly stated privacy policy. This allows a service requestor to select a service provider based on the quality of the interaction used to deliver their business contracts.

Service registries are important components of some Web Service environments. However, it is often important to address the direct request of service information. WS-Policy, as well as other metadata relevant to the service interaction (such as XML Schema and WSDL descriptions) can also be dynamically exchanged between interacting endpoints using the Web Services Metadata Exchange protocol. WS-MetadataExchange allows a service requester to directly ask a service provider for all or part of its metadata, without the mediation of a third party registry. Using the WS-MetadataExchange protocol service endpoints can exchange policies at runtime to bootstrap their interaction with information about the settings and protocols that apply. This is useful when not all policy information is in a repository, or when a requestor receives a reference to a service through some mechanism other than registry query. The direct dynamic exchange of policies also supports the customization of each specific interaction based for example on the identity of the other endpoint or any other aspect of the context under which it takes place. With this flexibility, Web Services can then be designed to offer different qualities of service to different targeted audiences.

Detailed Example

This call for papers has asked that each paper provide an illustration of the use case given. To illustrate our positions stated above, we have taken the W3C provided scenario[2] and used it to demonstrate the practical application of WS-Policy.

WS-Policy defines a policy to be a collection of policy alternatives, where each policy alternative is a collection of policy assertions. Some policy assertions specify traditional requirements that will be reflected in the messages that are exchanged (e.g., tokens used for authentication, digital signatures or encryption). Other policy assertions may have no manifestation in the messages exchanged yet are critical to proper service selection and usage (e.g., site privacy policy). WS-Policy provides a single policy grammar to allow both kinds of assertions to be defined. A valid interpretation of the policy expression below would be that an invocation of a Web service makes a choice from two policy alternatives (Lines 3 to 8) defined in WS-SecurityPolicy[3] to use either Username tokens or X509v3 tokens for authentication. Additionally an assertion for encrypting a header is included (lines 10 to 18).

```
01 <wsp:Policy xml:base=http://www.fabrikam123.com/policies  wsu:Id="SEC">
02   <wsp:ExactlyOne>
03     <wsse:SecurityToken>
04       <wsse:TokenType>wsse:UsernameToken</wsse:>
05     </wsse:SecurityToken>
06     <wsse:SecurityToken>
07       <wsse:TokenType>wsse:X509v3</wsse:TokenType>
08     </wsse:SecurityToken>
09   </wsp:ExactlyOne>
10    <wsse:Confidentiatlity>
```

[2] A Web service wishes to stipulate that clients are required support a reliable messaging protocol, and encrypt a specific SOAP header using OASIS Web Service Security with 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.

[3] http://www-106.ibm.com/developerworks/webservices/library/ws-secpol/

```
11      <KeyInfo>
12          <wsse:Reference URI="#ENCKEY" />
13      </KeyInfo>
14     <wsse:AlgEncryption URI=”http://www.w3.org/2001/04/xmlenc#3des-cbc/>
15        <MessageParts Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part >
16          <wsp:Header() >
17         </MessageParts>
18    <wsse:Confidentiality>
19 </wsp:Policy>
```

To facilitate interoperability, discipline authors of a Web service policy will need to partition and standardize assertions into non-overlapping disciplines. For example, adding an approach to distributed message trace might include defining specific assertions describing trace levels Generally, it is recommended that assertions are grouped by a namespace tag indicating the discipline. Within a namespace, there are generally two types of assertions. Simple assertions that contain only a Qname which can be used to find a match by taking two policies and intersecting them to find a common alternative. Simple assertions are generally seen to have self defining semantics. The second general type of assertion is a parameterized assertion in which additonal attributes are provided along with a qname. The additional values are used to clarify the semantics of the base assertion Qname. To illustrate the given use case further, let’s say the reliable messaging experts define the following simple (SequenceCreation) and parameterized (BaseRetransmissionInterval) assertions for RM:

```
01 <wsp:Policy xml:base=http://www.fabrikam123.com/policies  wsu:Id="RM">
02 <wsrm:SequenceCreation />
03 <wsrm:BaseRetransmissionInterval Milliseconds="3000"/>
04 </wsp:Policy>
```

The final step in the example, then is to associate them with a subject, in this case a WSDL document. WS-Policy Attachment defines two general-purpose mechanisms for associating policies with one or more Policy Subjects. The first (which we use to illustrate the use case given) allows XML-based descriptions of resources (represented as XML elements) to associate Policy as part of their definition (i.e. WSDL). The second allows Policies to be associated with arbitrary Policy Subjects independently from their definition (i.e., EPRs). The box below shows how policies may be defined in WSDL/1.1. WSDL 1.1 can be used to associate Policies with four different types of Policy Subject, identified as the Service Policy Subject, the Endpoint Policy Subject, the Operation Policy Subject and the Message Policy Subject. These subjects should be considered as nested, due to the hierarchical nature of WSDL.

```
<xml version="1.0"?>
<definitions name="StockQuote"
targetNamespace="http://example.com/stockquote.wsdl"
          xmlns:tns="http://example.com/stockquote.wsdl"
          xmlns:xsd1="http://example.com/stockquote.xsd"
          xmlns="http://schemas.xmlsoap.org/wsdl/"
           xmlns:ps="http://fabrikam123.com/policies">
    <wsp:UsingPolicy wsdl:Required="true" />
       …
    <portType name="StockQuotePortType"
        wsp:PolicyURIs="http://www.fabrikam123.com/policies#RM">
        <operation name="GetLastTradePrice">
           <input message="tns:GetLastTradePriceRequest"
           <output message="tns:GetLastTradePriceResponse"/>
        </operation>
    </portType>
    <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
     <wsp:Policy>
        <wsse:Integrity>
         <wsse:Algorithm  Type="wsse:AlgCanonicalization"
              URI=" http://www.w3.org/2001/10/xml-exc-c14n#"/>
            <wsse:Algorithm Type="wsse:AlgSignature"
                        URI=" http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
        <MessageParts Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part >
```

```
            <wsp:Body() >
          </MessageParts>
           <wsse:SecurityToken>
                <wsse:TokenTypeType>wsse:X509v3</wsse:TokenType>
        </wsse:SecurityToken>
        <wsse:Integrity>
    </wsp:Policy>
        <soap:binding style="document"
             transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetLastTradePrice">
           <soap:operation
           soapAction="http://example.com/GetLastTradePrice"/>
           <input>
                <wsp:PolicyReference
                     URI="http://www.fabrikam123.com/policies#SEC"/>
               <soap:body use="literal"/>
           </input>
           <output>
               <soap:body use="literal"/>
           </output>
        </operation>
    </binding>
</definitions>
```

One of the remaining issues for a policy framework to address is the requirement for additional XPath-based domain expressions to be used in conjunction with the <PolicyAttachment> mechanism. These new expressions will allow the external attachment of policy assertions to subjects other than service endpoints such as XML Schema definitions, WSDL definitions including operations, messages (inputs, outputs and faults) and message parts as well as specific sections of XML document instances. Using these new domain expressions it would be possible to address the last aspect of the proposed W3C use case……how to associate a existing P3P policy with a service via a SOAP header or to an existing WSDL file.

Summary

Web service standards like WSDL, XML and SOAP have achieved widespread and growing adoption. These specifications allow runtime interoperability between Web services, independently of implementing organization, hosting company or enabling infrastructure. WSDL, XML and BPEL4WS provide support for interoperability between different development tool suites to support collaborative design, and support documenting Web services in registries like UDDI. Most Web service scenarios are incrementally requiring more powerful functions like security, transactions and Reliable Messaging. There is an evolving set of specifications and standards defining such functions, and a growing set of middleware automating the implementation which is the most pressing problem critical to the growth of Web Services.

The evolution of these capabilities introduces a requirement for describing supported, expected, and required options. IBM feels that a framework built on XML and complimenting WSDL is the proper approach for solving this problem. The WS-Policy framework meets the requirements. WS-Policy

- Allows for independent specification of the XML languages for policies in different domains of enablement functions
- Provides composition of independently authored policy assertions into compound document that define valid combinations of choices
- Offers a simple, algorithmic approach for selecting the right policy setting between multiple options
- Allows documenting the valid policy associated with a Web service in registries, and also supports binding time (run time) exchange of the information through protocols like WS-MetadataExchange.

The space of “rules,” “policies,” “semantic information,” “ontologies,” etc. is a vast field. Rules and the Strategy Pattern are approaches to implementing services. Tackling the general “semantic” problem is a large undertaking, which will take years of trial and error. IBM feels that the next logical, achievable and most important problem is

supporting a framework for policy that compliments WSDL, XML, SOAP and the evolving set of standards for service assurance (transactions, security, etc.).

Contributors

Ali Arsanjani, Francisco Curbera, Donald F Ferguson, Allen Gilbert, Christopher B Ferris, Steve Graham, Maryann Hondo(editor; mhondo@us.ibm.com), David L Kaminsky, Anthony Nadalin, Chris Sharp, John Sweitzer, Tony Storey

References

[WS-Policy]

"Web Services Policy Framework (WS-Policy)," Siddharth Bajaj, Don Box, Dave Chappell, Francisco Curbera, Glen Daniels, Phillip Hallam-Baker, Maryann Hondo, Chris Kaler, Dave Langworthy, Ashok Malhotra, Anthony Nadalin, Nataraj Nagaratnam, Mark Nottingham, Hemma Prafullchandra, Claus von Riegen, Jeffrey Schlimmer (Editor), Chris Sharp, John Shewchuk, September 2004.

[WS-PolicyAttachment]

"Web Services Policy Attachment (WS-PolicyAttachment)," Siddharth Bajaj, Don Box, Dave Chappell, Francisco Curbera, Glen Daniels, Phillip Hallam-Baker, Maryann Hondo, Chris Kaler, Ashok Malhortra, Hiroshi Maruyama, Anthony Nadalin, Mark Nottingham, David Orchard, Hemma Prafullchandra, Jeffrey Schlimmer, Chris Sharp (editor), Claus von Riegen, and John Shewchuk, September 2004.