W3C WORKSHOP ON CONSTRAINTS AND CAPABILITIES FOR WEB SERVICES

SAP Position Paper

1 September 2004

Author: Claus von Riegen, SAP AG

INTRODUCTION

While the core Web services standards for message exchange (SOAP), description (WSDL), and discovery (UDDI) have matured, higher-level protocols for mission-critical Web service applications have recently begun emerging in domains such as security and reliable messaging.

These protocols need to allow for certain flexibility in how they are used individually or in combination with other protocols. For example, a protocol that provides integrity for Web services by means of digital signatures, such as WS-Security, needs to be flexible in terms of the canonicalization algorithm that is to be used before a signature is calculated. However, it is not conceivable that all Web service providers and consumers are willing or able to use all possible features.

Thus, there is a requirement to describe the constraints and capabilities of Web services so that Web service providers and consumers can determine the subset that they are jointly capable and willing to use for a certain Web service interaction. A language for the description of Web service constraints and capabilities (policies) is needed.

REQUIREMENTS FOR A WEB SERVICES POLICY LANGUAGE

The requirements for a Web services policy language can be grouped in the following categories.

Expressiveness

Every kind of constraint and capability that determine a Web service interaction needs to be expressible. The most important aspects to consider in this regard are domain, scope, and complexity.

The domain is the type of behavior that needs to be described for a Web service. Typical examples are security, reliable messaging, and transactional behavior.

The scope is the subject to which a certain policy is bound and can range from concrete, low-level (e.g. the security settings required for messages of a certain message definition that a Web service offers as an input message) to more abstract artifacts (e.g. the frequency with which messages need to be sent to a Web service so that, in a certain business context, a partner agreement can be fulfilled).

The complexity relates to the fact that while some constraints and capabilities have a boolean character in that they are simply required/offered or not (e.g. the WS-Security protocol is offered by a certain Web service), others are highly parameterized and combine aspects that have an interdependency (e.g. a certain canonicalization algorithm is to be used in combination with a certain security token type when digitally signing a message).

Robustness

In order to express the policy of a Web service, a policy language is needed that, while supporting all aspects of the previous section, should not change whenever new domain-specific policy aspects need to be described. Therefore, the most logical separation of concerns is to provide a core policy language that provides extensibility points for domain-specific policies. Thus, the core language should focus only on aspects that are common across the various domains. Additionally, the core language needs to focus on the description of Web service policy and should be composable with regard to policy exchange mechanisms (see below).

Composability and Reuse

Composability of domain-specific policies

Taking domain-specific policies into account, the core policy language needs to describe both combinations of policies from different domains (e.g. support of a certain reliable messaging protocol in combination with WS-Security) and alternatives between policies from one domain (e.g. use of either X.509 or user name security tokens).

The core language also needs to provide a concept for referencing externally defined policies in order to allow reuse of well-known policy patterns.

Composability within the Web service architecture

The core policy language needs to fit into the Web services architecture

Firstly, policies need to be associated with Web service artifacts described by means of, for example, XML Schema interface definitions, WSDL artifacts, WS-Addressing endpoint references, UDDI entities or BPEL processes. While this can already be achieved by using the various extensibility mechanisms (global attributes and elements, WSDL and BPEL extensibility points, and UDDI category systems), there is a need to provide guidance in terms of a) the meaning such an association has and b) the alternatives that should be used in favor of others in order to achieve utmost interoperability. For example, the following questions need further investigation:

- When publishing a Web service definition or deployed Web service to a UDDI registry, should the policies be associated with the respective WSDL artifact, the corresponding UDDI entity, or both?
- When associating a policy with a WSDL artifact, should this be done
  a) by adding it inline via WSDL element extensibility (note that WSDL 1.1 does not provide element extensibility for all artifacts),
  b) by adding references to policies that are external to the WSDL document, or
  c) by attaching it to a WS-Addressing endpoint reference (for WSDL ports)?

Secondly, policies need to be exchanged between parties involved in Web service interactions in order to

- know the alternatives provided by each party,
- derive the common subset of possible alternatives,
- jointly negotiate or individually select the alternative that is actually to be used, and
- configure each party's application accordingly.

Possible exchange mechanisms are

- sending policies via SOAP messages as a SOAP header
- retrieving policies from a Web service broker such as a UDDI registry, and
- retrieving policies from the Web service itself by means of emerging protocols such as WS-MetadataExchange.

Non-goals

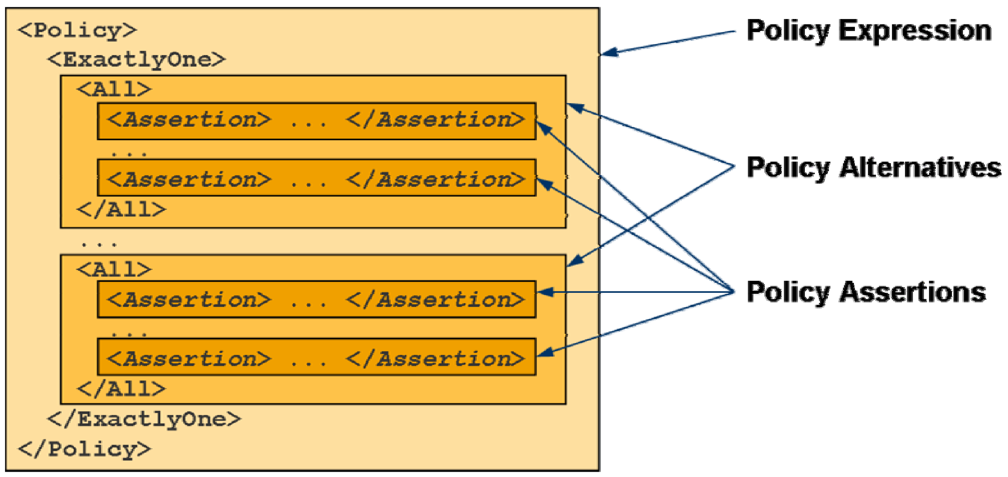
Domain-specific policies
The definition and standardization of domain-specific policies should be done by domain experts, since they know best which constraints and capabilities need to be described. However, the domain experts certainly need some guidance in terms of the effects a certain policy modeling approach has on the usability within the core policy language.

Policy exchange mechanism
The core policy language should allow both brokered and peer-to-peer exchange mechanisms and should not rule out approaches that may be developed in the future.

WS-POLICY FEATURES

WS-Policy and WS-Policy Attachment (**[WS-Policy]**, **[WS-PolicyAttachment]**) have been developed to cover the requirements described in the previous section. While WS-Policy can be seen as the core policy language for Web services, WS-PolicyAttachment addresses the requirements for associating Web service policy with Web service artifacts such as WSDL artifacts and UDDI entities.

Model



```
<Policy>
  <ExactlyOne>
    <All>
      <Assertion> ... </Assertion>
      ...
      <Assertion> ... </Assertion>
    </All>
    ...
    <All>
      <Assertion> ... </Assertion>
      ...
      <Assertion> ... </Assertion>
    </All>
  </ExactlyOne>
</Policy>
```

Figure 1: WS-Policy Information Model (Normal Form)

Figure 1 illustrates the information model proposed by WS-Policy. While a policy expression is a combination of admissible policy alternatives, of which one can be selected for a certain Web service interaction, a policy alternative is a combination of policy assertions that collectively constrain the Web service interaction. A policy assertion is the atom within a policy expression and represents a domain-specific, potentially complex, set of constraints and capabilities. It is opaque from the WS-Policy Framework point of view and can't be decomposed further.

Policy Types

WS-Policy allows the description of various policy assertion types.

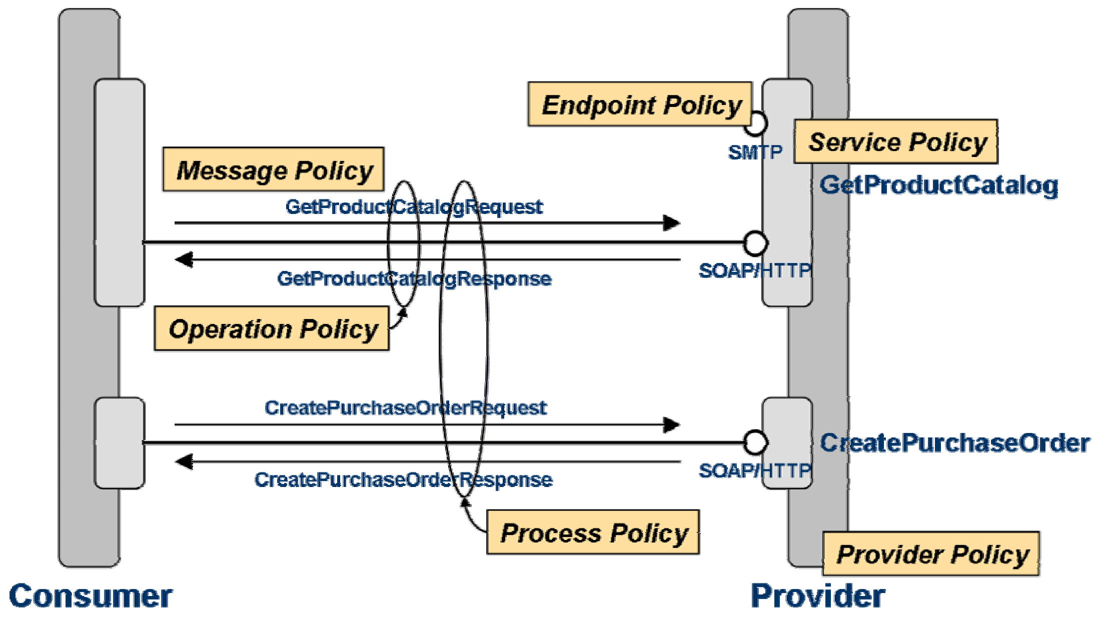


Figure 2: Policy Types

Policy assertions need to be categorized by the type of Web service artifact they are scoped to. For example, most policies regarding confidentiality, integrity, and reliable messaging apply to messages, i.e. they can differ between input and output messages of an operation. Another example is the priority a certain executable process definition has related to other processes, i.e. this policy is scoped to the process construct.

Policy Attachment

WS-PolicyAttachment describes, amongst other things, how to associate WS-Policy expressions with Web service artifacts such as WSDL artifacts and UDDI entities.

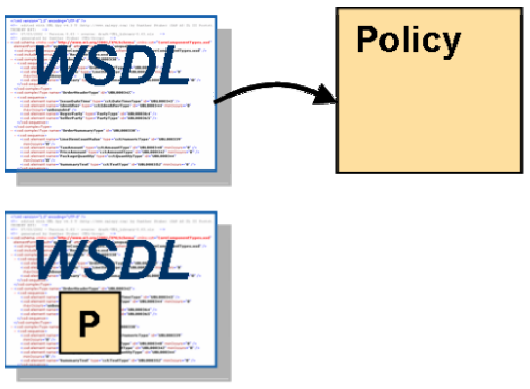


Figure 3: Associating Policy with WSDL

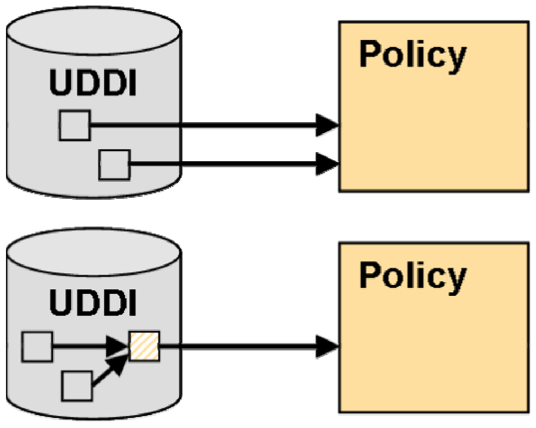


Figure 4: Associating Policy with UDDI

WS-PolicyAttachment makes use of WSDL's extensibility mechanisms to associate a policy expression with a WSDL artifact (see Figure 3), either by use of extensibility elements and/or internal references or by use URI-based external references.

WS-PolicyAttachment makes use of UDDI's extensibility mechanisms to associate a policy expression with a UDDI entity (see Figure 3), either by categorization with the policy expression URI or by categorization with the tModel that represents the policy expression.

USE CASES

The use case outlined by the Call for Position Papers[1] can be addressed by attaching policy assertions to WSDL[2] as illustrated by Figure 5.

```
<definitions name="StockQuote"
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/08/policy"
    xmlns:sv="http://schemas.xmlsoap.org/ws/2002/12/policy"
    xmlns:wsat="http://schemas.xmlsoap.org/ws/2003/09/wsat"
    xmlns:wsrm="http://schemas.xmlsoap.org/ws/2004/03/rm"
    xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd"
    ... >
 <wsp:UsingPolicy wsdl:Required="true" />
 ...
 <binding name="StockQuoteSoapBinding"
               type="tns:StockQuotePortType" >
  <wsoap:binding style="document"
                 transport="http://schemas.xmlsoap.org/soap/http" />
  <wsp:Policy wsu:Id="SRT" >
    <p:SpecVersion URI="http://schemas.xmlsoap.org/ws/2002/07/secext" />
    <wsp:ExactlyOne>
     <wsse:SecurityToken>
      <wsse:TokenType>wsse:X509v3</wsse:TokenType>
     </wsse:SecurityToken>
     <wsse:SecurityToken>
      <wsse:TokenType>wsse:UsernameToken</wsse:TokenType>
     </wsse:SecurityToken>
    </wsp:ExactlyOne>
    <p:SpecVersion URI="http://schemas.xmlsoap.org/ws/2004/03/rm" />
    <wsrm:SequenceCreation />
    <wsrm:BaseRetransmissionInterval Milliseconds="3000" />
   </wsp:Policy>
  <operation name="GetLastTradePrice" >
   <wsoap:operation soapAction="http://example.com/GetLastTradePrice" />
   <input>
    <wsoap:body use="literal" />
    <wsoap:header message="tns:SubscribeToQuotes"
             part="subscribeheader"
             use="literal"/>
   <wsp:Policy wsu:Id="EncryptSubscribeHeader" >
    <wsse:Confidentiality>
     <wsse:Algorithm Type="wsse:AlgEncryption"
         URI="http://www.w3.org/2001/04/xmlenc#3des-cbc" />
     <wsse:KeyInfo>
      <xenc:EncryptedKey />
     </wsse:KeyInfo>
     <wsse:MessageParts
       Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part" >
       p:Header(tns:SubscribeHeader)
     </wsse:MessageParts>
    </wsse:Confidentiality>
   </wsp:Policy>
   </input>
   ...
  </operation>
 </binding>
 ...
</definitions>
```

Figure 5: Reliable Messaging and Confidentiality policies

[1] A Web service wishes to stipulate that clients are required 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.

[2] The example makes use of sample WSDL fragments taken from the WSDL 1.1 specification.

Line (02) indicates that the WSDL is extended using WS-Policy; in this particular example, each of the provided bindings uses WS-Policy, and the WSDL should not be processed without understanding WS-Policy, so @wsdl:Required = "true".

Line (03) is a typical binding to SOAP (Line 04). Line (05) lists WS-Policy that applies to the binding as a whole; this element is named using @wsu:Id for convenient reference in other bindings (not shown).

Line (06) to (11) cover security. Line (06) is a generic assertion that indicates support for a specification, identified by URI, is required; in this case, WS-Security is required. Line (08) is a choice between assertions defined in WS-SecurityPolicy that indicate an X.509 security token (Lines 08-09) or a username and password (Lines 10-11).

Line (12) is another instance of that assertion, indicating support for WS-ReliableMessaging is required; Line (13) is an assertion that indicates that the RM destination is responsible for creating RM sequences; @wsp:Optional = "true" indicates this is not required. The assertion in Line (14) is a parameterized assertion indicating that the RM source will wait 3 seconds for an acknowledgement before retransmitting Messages.

Line (20) lists WS-Policy that applies to a specific input message; it too is named using @wsu:Id for use on other messages (not shown). The assertion in Line (21) is defined in WS-SecurityPolicy and indicates encryption using DES (Line 22), with a symmetric key protected by a token (Lines 07-11), of the Subscribe SOAP header (Line 25). (It’s assumed that the Subscribe SOAP header allows for encrypted content.)

In order to associate a P3P policy with a deployed Web service, a possible approach would be to publish the P3P policy to a UDDI registry in order to indicate that it is a service provider policy valid for all offered services as illustrated by Figure 6.

```
(01) <uddi:businessEntity businessKey="uddi:example.com">
(02)   <uddi:name>Example.com</uddi:name>
(03)   <uddi:categoryBag>
(04)     <uddi:keyedReference
(05)       tModelKey="uddi:schemas.xmlsoap.org:remotepolicyreference:2003_03"
(06)       keyName="P3P company policy for Example.com"
(07)       keyValue="http://example.com/p3p" />
         ...
(08)   </uddi:categoryBag>
       ...
(09) </uddi:businessEntity>
```

Figure 6: P3P policy

The example indicates that the service provider (businessEntity) with the businessKey "uddi:example.com" (line (01)), which is named "Example.com" (line (02)) has a policy expression attached to it (line (05)) that is identified by the policy expression URI "http://example.com/p3p" (line (07)).

REMAINING ISSUES

The following list illustrates some of the issues that remain open and need to be addressed by future work.

- In order to allow reuse and simplify intersection, it seems to be quite useful to define *patterns* of policy alternatives that combine multiple policy assertions, in particular for parameterized assertions.
- Web service artifacts that represent interface definitions, such as XML Schema definitions, WSDL portTypes/interfaces, and UDDI tModels, need to express their policy in an abstract way. Therefore, *abstract* policy assertions (such as “reliable messaging required”) and a binding to concrete policy assertions (such as “WS-ReliableMessaging *is a* reliable messaging protocol”) needs to be defined.
- Since the behavior of a Web service may change over time, an update to the corresponding policy expressions may be needed. Transitioning between different versions of policy expressions, expressing time dependencies and models for the reliable and timely exchange of policies requires further work.

RECOMMENDATION

SAP believes that WS-Policy and WS-PolicyAttachment are compelling candidates for addressing the need for both, a core Web services policy language and related conventions for associating Web service policy with core Web service artifacts.

REFERENCES

[WS-Policy], D. Box, et al, Web Services Policy Framework (WS-Policy), September 2004

[WS-PolicyAttachment], D. Box, et al, Web Services Policy Attachment (WS-PolicyAttachment), September 2004