Reference: w3c-ccws-9 Issue 3 (1st September 2004)

Publishing Web Service Policies

Position Paper for the W3C Constraints and Capabilities Workshop

This paper has been prepared in response to the call for participation for the W3C Workshop on Constraints and Capabilities for Web Services[1] to be held October 12-13, 2004, Oracle Conference Center, Redwood Shores, CA, USA.

This paper reflects the author's best understanding of the interests and requirements of BT in the "Constraints and Capabilities for Web services" debate, in particular requirements regarding the description and publishing of Web service policies. The paper should not be taken as an authoritative, formal or official company statement.

Copyright

Document history

Revision	Author	Date	Notes
1	PSD	11/8/04	First Issue, Paul Downey, Web Services Integration, BT Exact
2	PSD	13/8/04	Updated following comments from Dr Simon Thompson, Intelligent Systems Lab, BT Exact
3	PSD	1/9/04	Updated following comments from Chris Hipson, Web Services Integration, BT Exact

Distribution

Distribution will be controlled by:

Paul Downey

pp D3001,
BT Westside,
Apsley,
Hertfordshire,
HP3 9YF

paul.downey@bt.com

tel: +44(0) 1442 296260

Overview

As a service provider, BT requires an interoperable method for publishing attributes and other meta-data associated with a Web service. Typical attributes associated with a Web service may include, but are not restricted to:

- Authentication– endpoint is protected using user name or certificates.
- Message level security – which parts of a message should be signed, which should be encrypted. Hash and encryption algorithms to be used.
- Service status – development, test, in life, deprecated.
- Service availability – hours of service, recovery following failure.
- Quality of service – expected response times.
- Cost – the financial cost of invoking an operation.
- Conformance claims – the service is WS-I Basic Profile compliant.
- Privacy – user auditing, data retention.
- Loading – number of requests which may be sent in a given period, length of time which a request may be queued.

In some cases these attributes change the contents and function of messages exchanged. In other cases they are simply meta-data about the service and the environment in which the service expects to operate.

The purpose for describing these attributes in a formal policy framework is so that an autonomous agent may:

- Correctly send security and other, possibly out-of-message, information required by a service
- Select between different operations and endpoints offering a similar service. This is of particular interest when combining two or more services into a workflow constrained by a total cost or minimum response time.

Policy Framework

We would expect a policy framework to be layered as follows:

Policy Assertions

Here a uniform mechanism allows for expressing tuples of policy information. In essence an assertion is a property value pair, “service-status is live”. The property name would be unique and scoped within a namespace URI. This syntax may be rendered in multiple forms and attached to documents, not just SOAP or WSDL 2.0. Policy assertions may be made against several different layers of Web service description including an interface, a binding and endpoint.

Policy Vocabularies

For interoperability, the policy assertions should reuse existing names and namespaces for a given policy area. Here we would expect application specific vocabularies to be formed by interested parties, e.g. there would be a namespace and a set of names for describing WS-Security (WSS) attributes.

Policy Combinations

These combine a set of individual policy assertions into compound rules. Here the challenge is to define a language able to express combinations, but simple enough to be well implemented by agents. We would expect a policy framework to consider reusing an existing generic composition mechanism such as OWL-S.

Alternatively it is possible that specialised languages built upon standard Policy Vocabularies would emerge to target specific problem domains, e.g. P3P. Either approach would be acceptable to BT.

Policy Environment

We would expect a policy framework to exhibit the following non-functional features:

Policy Evolution

Some policies will change over a period of time. For example the status of a service may change from 'in life' to 'deprecated'. So the ability to communicate the life span of an individual policy along with a mechanism for communicating changes in policy is essential.

Policy Negotiation

Non-functional elements may be the subject of negotiation. A service should be able to exchange and negotiate policy assertions with an endpoint at run time, for example a sender may wish to assert an expectation regarding privacy in a message. It may be useful for a service to describe the negotiation mechanism to be employed, e.g. the service uses a cost versus expected response time trade-off function.

Policy Semantics

The commitment with which each policy is asserted should also be describable. What are the semantics behind publishing a policy such as the cost of calling a service? Is a policy assertion advisory or a contractual commitment?

Use-case

The use-case in the workshop call for participation exposes the current difficulties in publishing Web service policies:

> "A Web service wishes to stipulate that clients are required support a reliable messaging protocol, and encrypt a specific header with WS-Security using an 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."

There are specific concerns when addressing this use-case:

- Reliable messaging protocol is a requirement – the policy framework should have a 'required' or 'mustUnderstand' mechanism.
- The specific header element name to be encrypted needs to be identified, as does the encryption mechanism to be employed.
- The service accepting both X.509 and user name security means that a composition mechanism is required to express the combination of these two separate assertions. Which of the two security methods is preferred? Can both methods be applied to a single message?
- Where should each of these policies be published – against a WSDL interface, binding, endpoint or carried inside a SOAP message?

We would characterise the most obvious of the current options for publishing Web service policies as follows:

WS-Policy

This proprietary family of specifications has been the subject of significant practical development and promotion by some important vendors (notably Microsoft, IBM, BEA

and SAP). The policy language is independent of SOAP and WSDL, and may therefore be applied using extensibility to other XML based languages. Since WS-Policy is not published by a standards organisation, it has yet to be adopted by another significant group of vendors. WS-Policy therefore does not currently meet BT's requirement for an interoperable and non-proprietary framework.

Features and Properties
Defined in the WSDL 2.0 Last Call and SOAP 1.2 Recommendation, Features and Properties appear to provide a timely standard mechanism for making policy assertions in a Web service description. Unfortunately Features and Properties has been a very divisive technology remaining the subject of a minority opinion for WSDL 2.0 (from Microsoft and IBM, backed by SAP[2]) and there are some technical issues yet to be resolved, in particular:

- How should policies published as Features and Properties in WSDL 2.0 be reused in other contexts such as WSDL 1.1 and other languages which allow for XML based extensibility?

- How should individual features and properties be ordered and combined? This is the subject of another minority objection (from IONA, Oracle, Sonic and Sun) in WSDL 2.0 for compositors[3]. We are concerned that expanding the WSDL language to include a full featured policy composition language, whilst meeting our requirement for a policy framework, will further risk support from a significant group of vendors as well as only serving to delay WSDL 2.0 becoming a W3C Recommendation.

RDF and OWL
Subject to a large amount of effort from the W3C and backed by academic rigour, the Semantic Web recommendations offer great promise as a mechanism for building policy languages. Policies maybe described using RDF assertions in any one of a number of syntaxes and formats, not just those based on XML, e.g. PDF and PNG. External parties may develop specialised vocabularies for RDF assertions such as Dublin Core and RSS. Higher level languages exist to express combining RDF assertions, e.g. OWL-S. There is also the possibility of mapping assertions from other policy languages such as P3P. Unfortunately we see a lack of support and little strategic direction from vendors towards the Semantic Web tools in current mainstream Web service products.

Conclusion
BT as a provider and user of Web services requires an interoperable mechanism for publishing policy information. Unfortunately no interoperable mechanism currently exists, or is likely to arrive without standardisation from a body such as the W3C. Lack of a single, well adopted and interoperable policy framework risks the widespread adoption of Web services, a technology in which BT is investing heavily. BT therefore expects the W3C to lead the industry towards an open, interoperable framework for publishing Web service policies.

< End of document >

[1] http://www.w3.org/2004/06/ws-cc-cfp.html
[2] http://lists.w3.org/Archives/Public/www-ws-desc/2004Jul/0375.html
[3] http://lists.w3.org/Archives/Public/www-ws-desc/2004Jul/0371.html