Policy Management and Web Services

Tim Gleason
Oracle Corporation
224 Strawbridge
Moorestown, NJ 08054
tim.gleason@oracle.com

Kevin Minder
Oracle Corporation
224 Strawbridge
Moorestown, NJ 08054
kevin.minder@oracle.com

Greg Pavlik
Oracle Corporation
224 Strawbridge
Moorestown, NJ 08054
greg.pavlik@oracle.com

ABSTRACT
We maintain that the representation syntax of specific Web services policies is secondary to the general problem of policy management in the Web services space. We outline a broad view of the policy space in middleware systems, discuss emerging solutions for the Web services environment, and explain critical aspects of policy management that are required for taking Service Oriented Architectures (SOAs) to the next level.

Categories and Subject Descriptors
D.2.12 [**Software Engineering**]: Interoperability

General Terms
Management, Standardization, Languages.

Keywords
Web services, Policy Framework, Policy Management, Policy Enforcement.

1. Background
The term 'Policy' in distributed systems typically refers to an externally consumable statement of system constraints, capabilities or requirements that effect the interaction between a consumer and a service. In some cases, the policy may simply impact the decision to make use of a service; in other cases, the policy may place constraints on the interaction itself. An example of the former is a privacy policy, which, if deemed unacceptable, will cause the consumer to forgo use of a service altogether. An example of the latter is a policy that dictates that the service be used in the context of a transaction. In this case, interactions with the service must somehow be scoped as part of a larger unit of work.

Systems that are designed primarily with human users as principal actors in the consumer role tend to advertise policies that revolve around the decision to use a service. The archetypical example of such a system is the Web. Policies for the Web tend to fall into several classes

Policies designed to encourage use

Users may consider it desirable for a Web site to maintain strict rules about how information about site users is managed. For example, users are more likely to use a Web site if they have confidence the site owner will not distribute personal information and will guarantee an adequate level of protection for credit card data.

Though formal syntax is not always used to express policies of this nature, the Platform for Privacy Preferences (P3P) specification [1] describes a policy language for expressing the privacy rules adhered to by an organization in machine readable and human interpretable form. These policies generally assume a level of trust; in the Web environment, this is typically gained through a combination of certification by an independent authority and perhaps more commonly by reputation.

Policies designed to constrain access

Rules surrounding the access rights for a Web server are an example of this kind of policy. Typically, authentication and authorization procedures are integrated with the Web site's human user interface; in this case, the communication mechanism is relatively ad hoc and presented via HTML or similar markup languages.

Policies about availability

These are policies that declare under what terms a service is available. This information is typically communicated in quality of service agreements, as maintenance notices, or general information about a Web site. Examples of this kind of policy are notices of administrative practices requiring downtime for maintenance or payment requirements for use. These policy statements are important mechanisms for managing user expectations; in some cases, users may decide not to use a site based on conflicting availability requirements. Availability may apply not only to network presence of the service, but also to secondary business functions. For example, a Web site may be available on a 24X7 basis but may not have order processing available on weekends. Policies dealing with availability are also typically expressed through markup and interpreted by users.

These policy categories are not mutually exclusive. For example, a Web site may have policies that are intended to encourage the use of a site by a restricted class of users. The salient feature of Web policies is that they tend to be heavily oriented toward direct consumption by human users, assuming that users will find the policies and interpret them satisfactorily. In many cases, the policies are expressed in written statements on Web sites. Policies for Web sites tend to apply to the broad aspects of the site, rather than individual resources. For example, a certain portion of a Web site may require payment for use. More specialized services that provide access to copyrighted digital assets often place constraints on classes of resources (for example, you must pay .99 USD to download a song from Apple's popular iTunes Web site).

Distributed systems that focus on machine-to-machine interoperability have traditionally provided policies reflecting low-level constructs familiar to programmers that build such systems. Taking CORBA [2] as a representative example, policies are typically based on local configuration that is in turn tied to specific object references exported into the user environment. Policies for system level functions like security or transactions are exposed as properties of the distributed object reference (CORBA IOR). This allows programs to analyze remote services dynamically to assure that appropriate quality of service semantics are maintained when the service is invoked.

These policies are in general different from the typical Web policies in that:

1) Middleware policies are intended to be interpreted and used by software systems rather than human users.
2) For the most part, middleware policies deal with defining the semantics of interactions with a service. These policies are very different from the kinds of policies that are defined for Web resources.
3) These policies are very tightly bound to specific service implementations. In the CORBA example, policies are expressed to clients of the service within each individual object reference. Typical CORBA programs are based on the object oriented design paradigm, which may encourage the use of very fine-grained policies.

Web services policies combine elements found in both traditional middleware for machine-to-machine interoperability and policies associated with Web resources.

2. Web Services Policy

A general breakdown of the Web services policy space today includes:

Policies that focus on enabling and exposing traditional middleware system services like message delivery guarantees, transaction semantics, and security requirements. The WS-Policy Framework [3] specification proposed by Microsoft and IBM is oriented heavily toward expressing this kind of policy. Its emphasis on selection and logical operators – which we believe is of limited utility in practice even for the case of system services – make it a poor choice for other kinds of policies. As a general rule, these policies will affect the message payload by the addition of SOAP [4] headers specific to the policy selection that has been made for a message exchange. For example, the use of a WS-Reliability [5] functionality in a message exchange will include SOAP headers that look something like the following:

```
<wsrm:Request
xmlns:wsrm="http://www.oasisopen.org/committees/wsrm/schema/1.1/SOAP1.1"
xmlns:SOAP="http://schemas/xmlsoap.org/soap/envelope/"
SOAP:mustUnderstand="1">
<wsrm:MessageIdgroupId="20041221-160154-022.9@nobody.oracle.com"/>
<wsrm:ExpiryTime>2005-04-16T09:48:34</wsrm:ExpiryTime>
<wsrm:ReplyPattern>
<wsrm:Value>Poll</wsrm:Value>
</wsrm:ReplyPattern>
<wsrm:AckRequested/>
<wsrm:DuplicateElimination/>
</wsrm:Request>
```

Information policies: in many cases these will be formalizations of the kind of Web polices outlined above. Web services will require structured mechanisms to express informational policies, but complex policies will continue to be provided in forms targeted for direct human consumption in the near term. We believe that higher-level protocols will need to be developed to allow clients to express their expectations about specific informational policies. Informational policies typically impact the decision to use a service rather than the specific content of a message exchange. For example, a P3P document may express policies about the maintenance of personal information that are unacceptable to some users.

Service level agreements guaranteeing some combination of commitments around the quality of the service itself and the underlying business processes it represents. These policies are often tailored to specific users or classes of users and may depend on complex business rules. These policies are often applied by leveraging specific information associated with the established identity of the message sender.

Aside from the classes of policies we identify above, we assume the following requirements for Web services policies:

1) More than one policy may be associated with a service. We believe that multiple policies, often representing very different kinds of policy domains, will be in effect for a single service. For example, a single service may include policies for security, privacy, and business agreements.
2) A single policy may be associated with more than one service. Large organizations expect to set global policies and assure normal constraints and rules for sets of services. End users seeking to create a SOA are looking for mechanisms to support policy normalization.
3) Policies associated with a service may change over the lifetime of a service. For example, new polices may be introduced after a service has been deployed or existing policies may evolve over time
4) Policies need to vary independent of WSDL: new policies should be managed and provisioned independently of the basic business function and message exchanges offered by a service implementation.

At the current time, the Web services policy space is murky and evolving. There are proprietary proposals that emphasize different aspects of policy requirements, but tend to support one class of policy types better than others. In addition, there is the general problem of business rules and semantics. So called Semantic Web services have garnered great interest in academic circles but have not made in-roads in practice in the software industry.

The first step for providing a policy management solution is to achieve a standardized policy framework capable of meeting the requirements we have outlined. Regrettably, the industry has not yet been able to reach this critical milestone; in fact, no widely accepted standard effort exists in this space at the time of this writing. As a result, policies are often created in ad hoc ways and communicated through mechanisms that are out of band with respect to the Web services architecture and model. For example, we know of organizations maintaining Word documents that are passed via email describing how their Web services should be used. We believe the following design goals should be accommodated in a viable policy framework standard.

First, a policy framework should be able to support for different domains and styles of policy expression. Services will be bounded by a range of policy types, each critical in its own regard. A framework for supporting policies for security, reliability and transactions is necessary but insufficient. On the other hand, these

kinds of policies should be able to be expressed in a simple and easy to process set of assertions. We believe that a useful policy framework should provide containers for domain expressions that may utilize their own syntax and express their semantic requirements in a domain specific manner. The outline of a framework that provides domain containers is described in [6]. Much of the either/or discussions about policies that utilize Semantic Web capabilities versus assertion-based model may miss the point: domains should be free to utilize the technologies that appear best suited for the specific problem space

Second, informational policies are processed by service consumers to determine if a service may be used. Since policies may evolve independent of service interfaces, consumers should be able to express their expectations about informational policies that are believed to apply to a service. A SOAP header with a mustUnderstand="1" attribute could be used to convey expectations about specific informational policies; services that are not observing the policy expectation should return a fault rather than process the SOAP message carrying unsatisfied expectations.

Third, a policy document will be associated with a Web service. The standard should ensure that policies are not required to be included within WSDL documents or constructs so that the two may evolve freely. To support this model, we advocate extensions to WSDL indicating that a policy is enforced and how it may be obtained.

3. Policy Management

The classes of polices and general requirements for policies in the Web services environment, taken together, directly help to define the scope of a Web services policy management solution. Specifically, a Web services policy management solution needs to manage:

1) Policy Lifecycle

This includes the definition, maintenance and application of policies. The management of policies throughout their lifecycle combines problems of metadata management and organization as well as content management versioning and control facilities. Policies may be ad hoc or informal and should also be supported within the system: another motivator for dividing policy expressions into independent domains Many Web services management products support a policy repository capability that supplies some or all of these features and some protocol to provision policies to enforcement points. At the present point in time, these functions are achieved by non-standard and proprietary mechanisms.

2) Policy Discovery/Access

End users need to have access to policies to make decision about whether to use and how to use a service. Regardless of how policy lifecycles are controlled, a policy management solution must allow for metadata retrieval and policy organization. Most solutions will provide an association of policies and services, generally organized with some logical structure, perhaps based on taxonomies. The UDDI specification [7] provides interoperable rules for service registries, which can also expose policies and associated resources. In some cases, the Web services platform on which a service is hosted will directly supply the policy in response to a specific query using the HTTP protocol or a specialized Web services protocol for metadata retrieval. The WS-MetadataExchange specification [8] is an example of the latter.

3) Enforcement of policies for individual and groups of services.

One mechanism that is emerging in practice to handle policy enforcement is gateway services that act as active intermediaries in the SOAP processing model. The gateways process SOAP messages and enforce policy constraints or resolve system-level instructions before the message is provided to the service implementation for processing. For example, a gateway service may manage authentication and authorization based on policies defining the access control rules for a service or group of services (policy normalization). We believe that Web services intermediaries will prove to be fundamental to Service Oriented Architecture (SOA) deployments; we discuss this area in more detail below.

A policy management solution is foundational to a SOA: it provides a global model for an organization to understand and control the services within an organization. While application servers provide hosting platforms for individual services, a policy management solution provides visibility and control over a SOA topology and its characteristics. From this perspective, policies for organizations may be most effectively managed in centralized repositories that allow for businesses to set global policies and store information about how a service may be used. Individual service deployments can extend and specialize policies based on their specific requirements; this implies that well-defined rules must be in place for how policy domain expressions may be combined. Again, we believe this is largely a domain specific problem. Managing and storing metadata about services is largely a data management problem and amenable to storage in metadata containers built on standard relational database solutions.

Somewhat more problematic is the enforcement of managed policies, since services typically rest on a heterogeneous set of application server technologies. We believe that the following methods of policy enforcement are viable solutions for the Web services environment: local agents and gateways.

Agents that reside at service endpoints.

Agents allow processing logic to be inserted directly at service endpoints. This can occur via interception of the carrier protocol stream or within application server specific extensibility points specific to the Web services environment, such as JAX-RPC [8] Handlers. In either case, agents need to receive current policy definitions from the management repository.

Gateway-type active intermediaries.

Active intermediaries in the SOAP processing model can often be used to spread the processing logic of ultimate message recipients across multiple servers. A gateway can be configured to transparently enforce policies that are expressed as properties of the Web service. While the archetypical use case for Web services gateways is enforcement of security policies, almost any policy can be enforced or observed via a gateway architecture by organizing a pipeline of policy enforcement steps required for the service. Since these intermediaries may combine global and service specific policies, composition rules should be well-specified and isolated to overlapping domains.

Both enforcement mechanisms can be used to provide data about

policy enforcement to systems management consoles. This combination of a well-factored policy framework, policy provisioning, access, and enforcement mechanisms, and monitoring capabilities provide a compelling solution for the Web services environment.

One area that requires special care is the provisioning of policies between centralized repositories and enforcement points: it is important that policies are applied consistently, particularly in replica-based cluster environments. This can be a significant challenge in agent-based systems and is an area that is rife for interoperability research proposals and ultimately standardization.

4. Conclusion

A complete Policy framework needs to accommodate the requirements for different classes of policies and the solution architecture that is emerging for the management of policies. We do not believe that current proposals meet the full range of requirements that exist for a complete Web services policy solution. In particular, current proposals are not tailored to the emerging requirements, organization and deployment topologies of Web services networks and policy management solutions that are required for a coherent SOA deployment.

5. Acknowledgements

Special thanks to Ashok Malhotra and Jon Maron for their insightful comments. Thanks also to the Oblix CoreSV product team for sharpening our understanding of Web services management in commercial practice.

6. References

[1] Cranor, Lorrie et al. The Platform for Privacy Preferences 1.0 Specification. (April 2002) http://www.w3.org/TR/P3P/

[2] Common Object Request Broker Architecture: Core Specification. (March 2004) http://www.omg.org/docs/formal/04-03-01.pdf

[3] Bajaj, Siddharth et al. Web Services Policy Framework. (September 2004) ftp://www6.software.ibm.com/software/developer/library/ws-policy.pdf

[4] Gugdin, Martin et al. SOAP Version 1.2 Part 1: Messaging Framework. (June 2003) http://www.w3.org/TR/soap12-part1

[5] Iwasa, Kazunori. WS-Reliability 1.1. (August 2004) http://docs.oasis-open.org/wsrm/2004/06/WS-Reliability-CD1.086.pdf

[6] Ashok Malhotra and Umit Yalcinalp. Position Paper for W3C Constraints and Capabilities Workshop. (August 2004) http://www.w3.org/2004/08/ws-cc/amuy-20040903

[7] Ballinger, Keith et al. Web Services Metadata Exchange (September 2004) ftp://www6.software.ibm.com/software/developer/library/WS-MetadataExchange.pdf

[8] Chinnici, Roberto and Hadley, Marc. Java API for XML based RPC (JAX-RPC) 2.0. (June 2004) http://jcp.org/aboutJava/communityprocess/edr/jsr224