



This Presentation Courtesy of the
International SOA Symposium
 October 7-8, 2008 Amsterdam Arena
www.soasymposium.com
info@soasymposium.com

Founding Sponsors



Platinum Sponsors



Gold Sponsors



Silver Sponsors



**Contracts
 Policies and Services
 Oh My**




SYSTEMATIC THOUGHT LEADERSHIP FOR INNOVATIVE BUSINESS

Ümit Yalçınalp, Ph.D.
 SAP Labs, Palo Alto
umit.yalcinalp@sap.com



Outline



- Basics: Policies as a contract
- Policy Specifications
- Concepts in WS-Policy
- Useful Patterns for Providers and Consumers
- Gotchas
- Conclusions

SAP RESEARCH October, 2008 SOA Symposium 3

- This talk is about principles
 - Why is it this way? What is it good for? Not lots of pointy brackets
- Best Practices and some gotchas
- Overview of Policy chapters *in WS-Contract Design and Versioning Book*
- Cheat sheet for understanding the reasoning behind the specs

Policies in SOA



- **Critical component of SOA**
 - Consumers may not be able to use services without explicit policies
 - Service providers may not be able
 - provide QoS without policies
 - enforce behaviors from consumers
 - deploy services
- Defined as an expression of constraints and capabilities
- Result in a set of behaviors for providers and consumers
- Apply typically to interactions with a service

Importance of Understanding Policy Frameworks



- **Providers**
 - Define the policies for their services, specifically QoS
 - Determine the policies that may apply
 - Decide whether multiple policies may apply to the same set of messages
 - Should support the alternatives if provided
- **Consumers**
 - Determine whether they can use the service based on a contract
 - Likely to have their own policies
 - Must determine which policies will apply when they use a service
 - Choose among alternative ways of using a service


You may be a provider, a consumer or be in both roles
You may be designing new expressions or new policy
vocabularies

The capabilities and limitations help in

The SAP logo, consisting of the letters 'SAP' in white on a blue square background.

- Designing better contracts
- Avoiding pitfalls in policy design


Two Prominent Players



- WS-Policy
- SCA Policy Framework

SAP RESEARCH October, 2008 SOA Symposium 9

WS-Policy Framework



- Crucial component of a contract using Web Services
 - Expression syntax
 - Compositions with expressions
 - Help define where to use policies
 - Defines how to use with WSDL
 - Guidelines for Authoring Policies
- Established set of specifications
<http://www.w3.org/2002/ws/policy/>
- Targets WS-*, for defining and using *concrete* policies
- Focus in this talk

SAP RESEARCH October, 2008 SOA Symposium 10

Basics of WS-Policy: Assertion



- It is a QName

<umit:speakEnglish>

<umit:takesnotes>

- Basic block in for a policy expression
- Has specific semantics
- Has specific schema
- Define a behavior
- Defined by a governance body, specification, a company, department...
- May contain other assertions by *nesting* other assertions
- Typically defined in a document + schema
- Specifies where it can be used as a contract (in WSDL)
- May specify related behaviors, compositional behavior

Standardized Policy Assertions



- WS-Addressing Metadata (w3c)

<http://www.w3.org/TR/ws-addr-metadata>

- WS-Security Policy 1.2 (OASIS)

<http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.html>

- WS-Reliable Exchange (OASIS) v1.1 and v1.2

<http://docs.oasis-open.org/ws-rx/wsrmp/200702/wsrmp-1.1-spec-os-01-e1.html>
<http://docs.oasis-open.org/ws-rx/wsrmp/200702>

- WS-Atomic Transaction (OASIS)

<http://docs.oasis-open.org/ws-tx/wstx-wsat-1.1-spec-errata-os/wstx-wsat-1.1-spec-errata-os.html>

Commonality:

- Describe wire-level protocols
- Target message exchanges with endpoints

Basics of WS-Policy: Policy Expressions



- *Operators* for conjunctions (`wsp:All`) and disjunctions (`wsp:ExactlyOne`)

```
<wsp:All>
  <umit:speakEnglish>
  <umit:takesnotes>
</wsp:All>
```

```
<wsp:ExactlyOne>
  <umit:speakEnglish/>
  <umit:speakGerman/>
</wsp:ExactlyOne>
```

- Are used to construct an expression using multiple assertions

WS-Policy Policy Expressions Equivalence



- Expressions can be composed or simplified with applying operators
 - Idempotent
 - Symmetric
 - Distributive (`wsp:All` over `wsp:ExactlyOne`)
- Full *Calculus*: Chapter 10

Policy Alternative



- A set of policy assertions is an alternative
- Example: Two alternatives

```
<wsp:ExactlyOne>
  <wsp:All><umit:speakEnglish/></wsp:All>
  <wsp:All><umit:speakGerman/></wsp:All>
</wsp:ExactlyOne>
```

wsp:optional attribute



- Shorthand for creating alternatives in an expression


```
<umit:speakEnglish wsp:optional="true"/>
```
- Is equivalent to


```
<wsp:ExactlyOne>
  <wsp:All><umit:speakEnglish/></wsp:All>
  <wsp:All/>
</wsp:exactlyone>
```
- *Implies two alternatives, thus two different behaviors*
 - an alternative where umit speaks English
 - an alternative where there are no behaviors implied
- **This is a pattern to indicate a *capability supported but not required***

Assertion Nesting



- A relatively new concept for WS-Policy
- A policy assertion may be nested in another assertion
 - By including it in a policy expression as its child
 - Expression is created by wrapping with `wsp:Policy` element
 - Its semantics depend or qualify the parent assertion
 - It is usually defined by governance body that specifies the parent assertion


Example: WS-Addressing Metadata



- `wsam:Addressing`
 - `wsam:AnonymousResponses`
 - `wsam:NonAnonymousResponses`
- You can use one or the other child assertion, *not both*

```
<wsam:Addressing>
  <wsp:Policy>
    <wsam:AnonymousResponses/>
  </wsp:Policy>
</wsam:Addressing>
```

Parametric Assertions:



An assertion with embedded content

```
<wsp:Policy>
  <umit:Speaks>
    <umit:language>English
  </umit:language>
</wsp:Speaks>
</wsp:Policy>
```


Compare with

```
<wsp:Policy>
  <umit:Speaks>
    <wsp:Policy>
      <umit:EnglishLanguage/>
    </wsp:Policy>
  </umit:Speaks>
</wsp:Policy>
```

- The content of a parametric assertion is **NOT** guaranteed to be understood by all frameworks!
- More details in Chapter 16

SAP RESEARCH October, 2008 SOA Symposium 19

Associating Policies with WSDL



- Policy Subjects: *Collective Attachment points* in WSDL
 1. Service (`wsd11:service` or `wsd120:service` element)
 2. Endpoint (`wsd11:binding` or `wsd120:binding`,
`wsd11:port` or `wsd120:endpoint` elements,
`wsd11:portType` or `wsd120:interface` element)
 3. Operation (*operation* elements in *binding*,
operation elements in `wsd11:portType` or
`wsd120:interface` elements)
 4. Message (`wsd11:message` element or
input | *output* | *fault* elements in
operation elements)
- 4 different *effective* policies that may simultaneously apply

SAP RESEARCH October, 2008 SOA Symposium 20

Assertions specify their attachment points



Example: `wsam:Addressing`

- Can be attached to `wsdl11:binding/wsdl20:binding` or `wsdl11:port/wsdl20:endpoint` elements
- Can not be attached to `wsdl11:portType` or `wsdl20:interface` elements

Discussion: Why can't we put this assertions in the `portType` or `interface`?

Some Points to Consider



- The design of WSDL + 4 simultaneous effective policies
 - Service Effective Policy governs all endpoints and protocols
 - Endpoint Effective Policy govern all messages for an endpoint
 - Policy for a `portType` or `interface`
 - must be applicable to all endpoints
 - must be abstract
- Avoid attaching incompatible policies in a hierarchy
- Avoid breaking the contract definition itself
 - Most standardized protocol assertions apply to Endpoint Policy Subject
 - Do not use them with `portType` or `interface` when not abstract

Attachment Patterns with WSDL



■ Inclusion patterns with WSDL:

Insert policy expressions in attachment points directly as extensibility

```
<binding>
  ...
  <operation name="simpleCommunicate">...<input>...
</operation>
  <wsp:Policy><wsam:Addressing/></wsp:Policy>
</binding>
```

Use partial centralization



- Define *named* policies in `wsd11:descriptions` or `wsd120:definitions` elements

```
<wsdl20:definitions>
  ...
  <wsp:Policy wsu:Id="communicationsPolicy"/>
    <wsam:Addressing wsp:optional="true"/>
    <myservice:LoggingEnabled/>...
  </wsp:Policy>
</wsdl20:definitions>
```

- Refer from attachment points

```
<wsdl20:binding>
  ...
  <wsdl20:operation name="simpleCommunicate">...
</wsdl20:operation>
  <wsp:PolicyReference URI="#communicationsPolicy"/>
</wsdl20:binding>
```

Policy Centralization Pattern



- Decouple WSDLs from Policy Documents
 - Create (a) policy document(s) independent of WSDL
 - Use references to WSDL elements in the document to refer to attachment points
 - *WSDL fragment identifiers* (URIs) are used to refer where you want to attach
 - More details in Chapter 16

Comparison



- Centralization decouples the contracts and assures reuse of WSDLs
- Centralization allows evolution and change
- Centralization require vendor specific solutions, repositories, independent management
- Inclusion is easy for testing and standard
- Be aware of what your toolkit supports



Why should you care as policy designers?



- Design time policies ultimately affect runtime policies
 - Policies may be attached to different places in WSDL
 - Using a service is governed by endpoints/ports and the messages exchanged
 - Understanding “complete” policy that applies is necessary
- A complex policy expression may be reduced to a simpler equivalent form
 - Compact form (written by users)
 - Normal form (utilized FOR runtime)
- The rules of composition and rewriting ain't that hard! 😊

Three Amigos of Policy Expressions Help To Answer *Which policies apply*



- **Merging**
 - Used to find the combined policy that will apply
 - Use `wsp:All` to combine different policy expressions
 - **For finding the effective Policy**
- **Normalization**
 - Used to find distinct alternatives (choices) given an expression
 - Only one alternative will apply in a given situation
 - Create an expression with no nested alternatives
 - Essentially translates into *disjunctive normal form*
 - **For finding the alternatives and determining which one ultimately applies**
- **Intersection**
 - Given two different normalized policy expressions, finds the common set of assertions
 - **Compatibility** of assertions is used to find the union of assertions that apply to both
 - **For determining which alternative to use in an interaction**

wsp:ignorable **attribute**



- A behavior that is engaged by user of the assertion
- Changes the way an *intersection* works with policies
- Consumer interaction with service may not need to change due to policy
- Consumers may decide not to use the service based on its presence
- *Implies single behavior*

```
<omit:TakesNotes wsp:ignorable="true"/>
```

- **Ignorable and optional are not the same!**

More on Intersection



- Regarded as guideline, not requirement toolkits
- Can utilize lax or strict modes
 - Lax ignores the ignorable policy
 - Strict must consider the ignorable policy
- Compatibility is based on QNames, not content
 - Parametric vs nested assertions play a role
 - Domain specific processing can change which assertions are compatible!
- See Chapter 17 for more on the tree amigos

Service Provider: What is my overall policy in WSDL?



Goal: Compute the policy for a *message exchange*

Do:

- Use *Merge* of the policies to obtain 4 effective policies
- Normalize the effective policy
- Make sure that runtime policy can be identified
 - Distinct alternatives may exist, only one will apply
 - Consumer should be able to use the alternative
 - Must be deterministic
 - If not, distinct ways of identification must be used
 - Use distinct ports or endpoints for each alternative
 - See **chapter 16**: Concurrent Policy Enabled Contract
 - **No policy negotiation available with standards**
- **Design time effects runtime !!!**

Service Consumer: How do I use the policy?



- Problem: Can I use this service? If so, which of the policies can I use?
 - Determine to ignore or use providers ignorable policy (lax or strict intersection)
 - Compute an intersection with provider's policy
 - If the intersection does yield zero alternatives, it fails
 - If multiple alternatives exist, choose one
 - Communicate with provider using the chosen policy
 - The determinism considerations apply here as there is no negotiation!

Define Your Own Assertion:



- The Assertion document
 - Namespace
 - Semantics of the behavior
 - Protocol
 - Headers
 - XML Schema
- Applicability
 - Roles
 - Both Parties: Protocol
 - Provider only
- The interactions with others
 - Does it affect/contradict other assertions,
 - Guidelines in composition
- The attachment points in a WSDL

Existing Assertions



- Protocol Specific
 - Enabling a Protocol or protocol capability (i.e. WS-Addressing)
- May be a capability (with `wsp:optional = "true"`)
- Most complicated is WS-Security Policy
 - A good reference for security concerns
Developer's Guide to SAP NetWeaver Security
<http://www.sap-press.de/katalog/buecher/titel/gp/titelID-1656>

Gotchas



- Parameterized Assertions
 - QName is used for compatibility testing, *not XML contents*

```
<me:speak>English</me:speak>
```

is *compatible with*

```
<me:speak>German</me:speak>
```


unless domain specific logic is applied
 - Policy calculus, transformation will need to be extended
 - Affects how you should define author your own policies
- Avoid ambiguous runtime policy
- Non-wire policies should utilize different endpoints
- Do not assign protocol policies to abstract WSDL
- Be Aware of Toolkit Limitations

Emerging Policy Specification: SCA Policy Framework



- Policy Framework for **Service Component Architecture**
- An Emerging Specification in OASIS for SOA
- Targets defining
 - *Abstract* policy vocabularies
 - Relationship of vocabularies to a set of policies
- Independent of specific implementation mechanism of components and services (EJBs, WS, etc).
- Addresses **interactions** and **implementation** policies
- Concrete policies may materialize differently depending on deployment
- Defers to WS-Policy or other policy languages for concrete realization


Conclusions



- WS-Policy Framework is established
- Contract designers should utilize guidelines and be aware of limitations
- SCA Policy Framework is emerging
- Thank You For Listening!

SAP RESEARCH October, 2008 SOA Symposium 37

Conclusion



Ümit Yalçinalp
SAP Research, Office of the CTO

umit.yalcinalp@sap.com

SAP RESEARCH October, 2008 SOA Symposium 38