Web Services Security and More: The Global XML Web Services (GXA) Initiative

Joseph M. Chiusano

Booz | Allen | Hamilton

Open Source for National and Local eGovernment
Programs in the U.S. and EU

Washington, DC
March 17, 2003

What is the Global XML Web Services Architecture?

- An **application-level protocol framework** built on the foundation of **XML and SOAP** that is designed to provide a **consistent model** for building **infrastructure-level protocols** for Web services and applications

- Defines a family of **pluggable infrastructure protocols** that provide applications with **commonly needed services** such as security, reliability, and multi-party agreement
  - To “fill the gap” in the current Web services stack

- Specifications authored by Microsoft, IBM, Verisign, BEA Systems, RSA Security and SAP

- Growing need for **consistent support of more secure Web services**, especially at the levels of inter-enterprise trust, security, and business policy agreement

GXA Milestones

April 2001	*October 2001*	*April 2002*	*July 2002*	*December 2002*	*2003*
W3C Web Services Workshop *Microsoft and IBM outlined a road map for Web services security*	**Initial GXA specifications released**	**Joint White Paper** *Microsoft and IBM release "Security in a Web Services World: A Proposed Architecture and Roadmap"*	**WS-Security Specification moved into OASIS**	**6 new specifications released**	**3 new specifications released *(best guess)***

GXA defines several Design Principles by which its specifications are designed

1. **Decentralization and Federation** – GXA protocols are designed with “constrained agreement” in mind

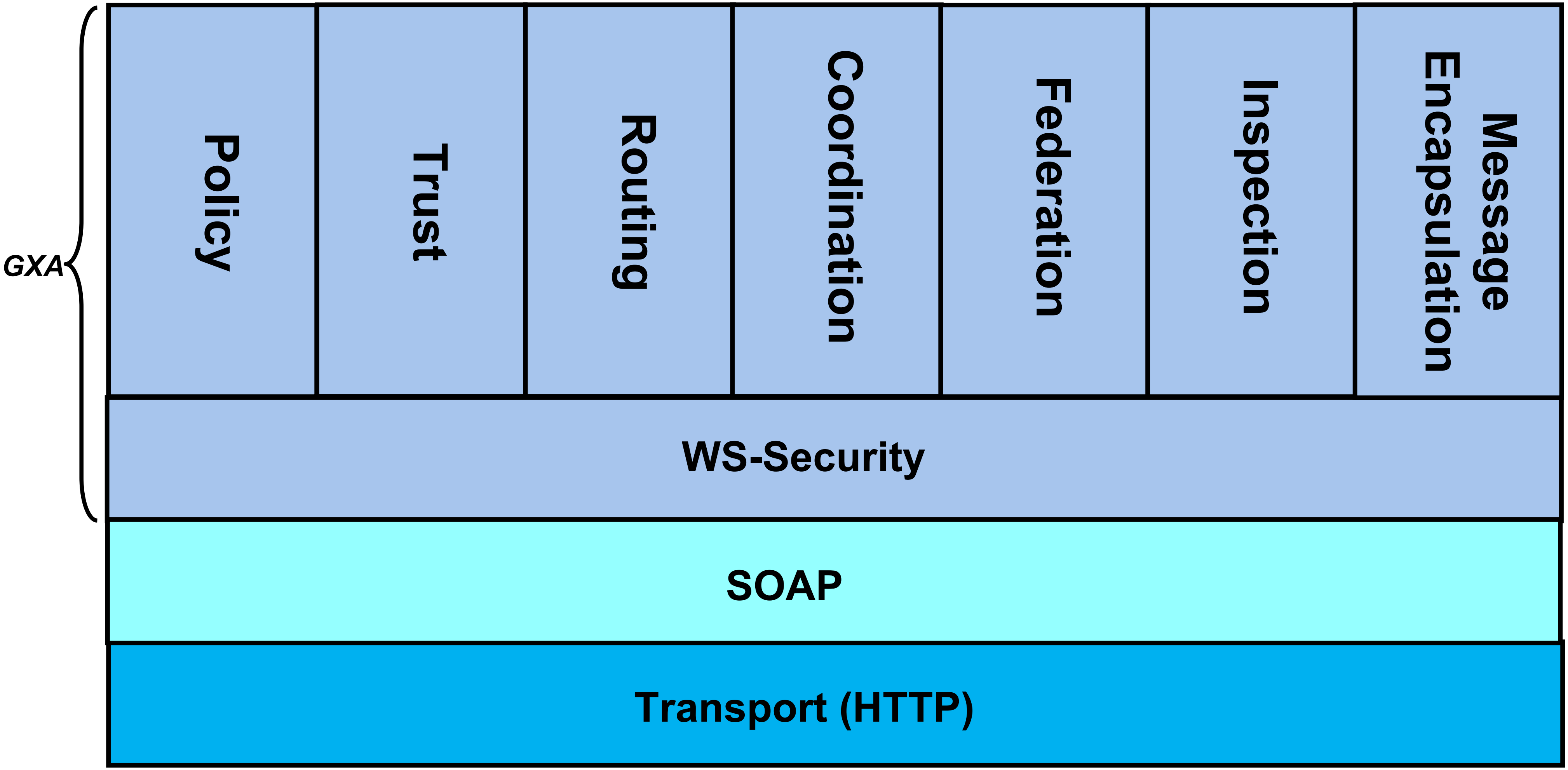
2. **Modularity** – GXA architecture is built on modular components rather than large, monolithic specifications that offer end-to-end functionality

3. **XML-Based Data Model**

4. **Transport Neutrality** – GXA is specified entirely at the SOAP level

5. **Application Domain Neutrality** – GXA protocols are general-purpose solutions to broad problems that span application domains

Web Services Stack: Where GXA Fits



GXA

Policy

Trust

Routing

Coordination

Federation

Inspection

Message Encapsulation

WS-Security

SOAP

Transport (HTTP)

The GXA specifications include 7 main concentrations

Concentration	*Specification*	*Description*
Security	WS-Security	The Cornerstone of GXA
Policy/Trust	WS-Policy	Expressing Enterprise Security Policies
	WS-SecurityPolicy	Policy-Related Extensions to WS-Security
	WS-PolicyAssertions	Message-Related Assertions
	WS-PolicyAttachment	Policies Applied
	WS-Trust	Managing Trust Relationships
	WS-Privacy*	Stating Privacy Requirements and Preferences
Routing	WS-Routing	Application-Level Routing
	WS-Referral	Dynamic Routing
Coordination	WS-Coordination	Coordination Requirements
	WS-Transaction	Transactional Properties

not yet released

The GXA specifications include 7 main concentrations

Concentration	***Specification***	***Description***
Federation	WS-SecureConversation	Establishing Security Context
	WS-Federation* WS-Authorization*	Constructing Federated Trust Scenarios Specification and Management of Access Policies
Inspection	WS-Inspection	Web Services Inspection Language
Message Encapsulation	DIME WS-Attachments	Direct Internet Message Encapsulation Attachments in DIME

***not yet released**

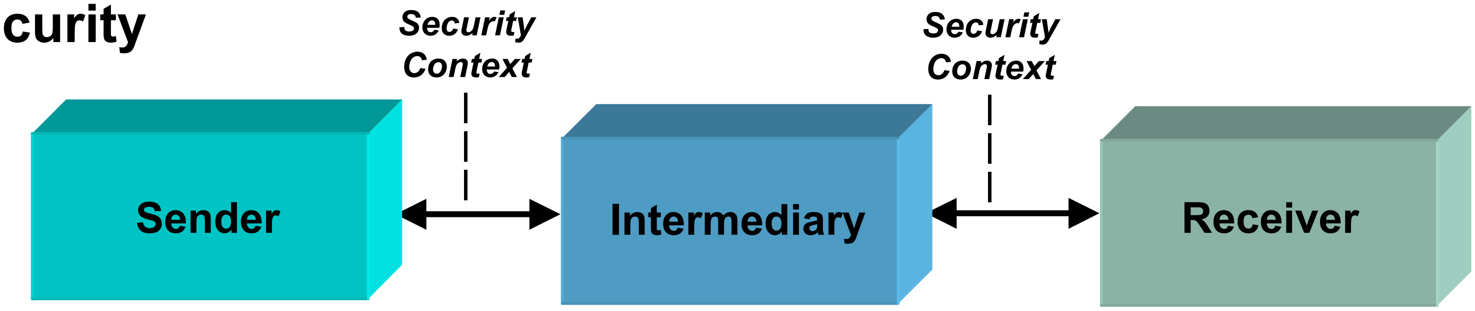
WS-Security

WS-Security defines a standard set of SOAP extensions that enable applications to construct secure SOAP message exchanges

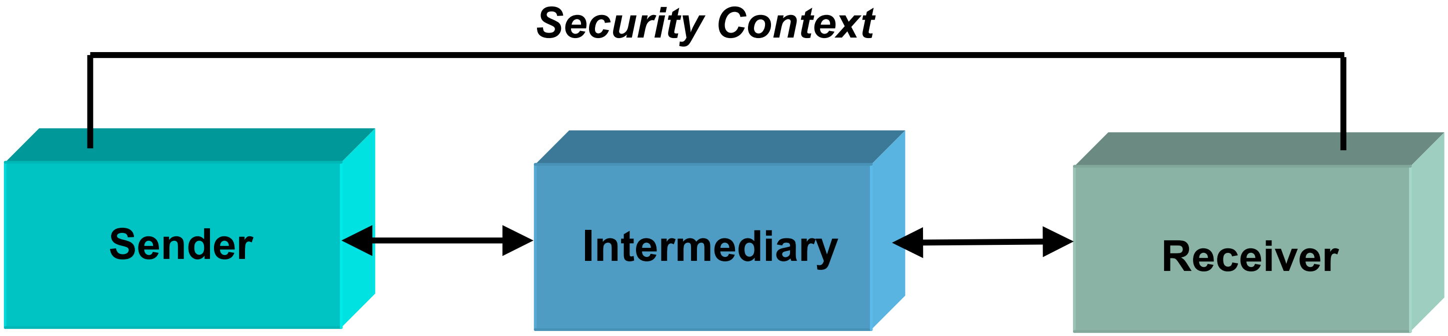
- Enables implementation of **credential exchange**, **message-level integrity** and **confidentiality**
- Original specification released **October 2001** by Microsoft, IBM, Verisign
- Leverages **existing standards and specifications** such as ITU-T X.509, XML Encryption and XML Signature

WS-Security addresses end-to-end security, where trust domains need to be crossed

- HTTP and its security mechanisms *(SSL/TLS)* address **only point-to-point security**



- WS-Security addresses how to **maintain a secure context over a multi-point message path**



Some XML Examples

- ***Example #1*** - Direct Trust Using Username/Password:

```
<?xml version="1.0" encoding="utf-8"?>
<S:Envelope
     …namespace declarations go here…>
     <S:Header>
          <wsse:Security>
            <wsse:UsernameToken wsu:Id="MyID">
                   <wsse:Username>Zoe</wsse:Username>
                   <wsse:Password>MyPassword</wsse:Password>
                   <wsse:Nonce>FKJh...</wsse:Nonce>
                   <wsu:Created>2001-10-13T09:00:00Z</wsu:Created>
            </wsse:UsernameToken>
            ……………
          </wsse:Security>
     </S:Header>
     <S:Body wsu:Id="MsgBody">
                 ……………
     </S:Body>
</S:Envelope>
```

This is the standard <Security> header, which contains the Username and Password

Some XML Examples

- ***Example #2*** - Digital Signature *(Integrity):*

```
<?xml version="1.0" encoding="utf-8"?>
<S:Envelope
     <S:Header>
          <wsse:Security>
             <wsse:BinarySecurityToken
              ValueType="wsse:X509v3"
              EncodingType="wsse:Base64Binary"
                   wsu:Id="X509Token">
                 MIIEZzCCA9CgAwIBAgIQEmtJZc0rqrKh5i...
              </wsse:BinarySecurityToken>
              <ds:Signature>
                      ................
                 <ds:SignatureValue>BL8jdfToEb1l/vXcMZNNjPOV...
                  </ds:SignatureValue>
                 <ds:KeyInfo>
                    ................
                 </ds:KeyInfo>
             </ds:Signature>
          </wsse:Security>
     </S:Header>
     <S:Body wsu:Id="MsgBody">
            ................
     </S:Body>
</S:Envelope>
```

This is the base64-encoded digital signature

In Summary

- Can also perform the following functions:
  - Message Encryption *(Confidentiality)*
  - Message Expiration *(Timestamps)*
- Specification:

  http://www.oasis-open.org/committees/wss
- Currently under OASIS

Potential E-Government Applicability

- May have applicability to E-Government initiatives (such as E-Authentication) as an **“authentication gateway” mechanism**
  - *Ex’s:* Username/password verification, digital certificate verification, etc.
- Incorporation of an open standard could allow **more seamless interaction** with an authentication gateway by participating systems, and therefore **potentially greater usage**

WS-SecurityPolicy

WS-SecurityPolicy defines how to describe policies related to features defined in WS-Security

- Specification released **December 2002** by Microsoft, IBM, Verisign, and RSA Security
- Example of policy:
  - “This Web service accepts X.509 certificates and Kerberos tickets, but **you must choose exactly one of these** and X.509 certificates are the preferred mechanism”
- *Policy Assertion* – represents an individual **preference, requirement, capability**, or other property
  - “This Web service accepts X.509 certificates”
  - “This Web service accepts Kerberos tickets”

WS-SecurityPolicy defines several types of assertions

- Types of assertions:
  - ***SecurityToken assertion:*** Specifies **security token types required/accepted** by a Web service
  - ***Integrity assertion:*** Specifies that **specific portions of a message must be signed**, and specific algorithms/keys to be used *(ex: SHA-1 algorithm, RSA key)*
  - ***Confidentiality assertion:*** Specifies that **specific portions of a message must be encrypted**, and a specific algorithm to be used *(ex: AES, 3DES)*
  - ***Visibility assertion:*** Indicates portions of a message that **must be visible to an intermediary or endpoint** *(i.e. unencrypted)*
  - ***Message age assertion:*** Specifies the **acceptable time period** before messages are declared "stale" and discarded

An XML Example

- SecurityToken assertion:

```
<wsse:SecurityToken TokenType=”wsse:X509v3”
      wsp:Usage="wsp:Required" wsp:Preference=”50”/>
```

An X.509 certificate is accepted by this Web service

- Specification:

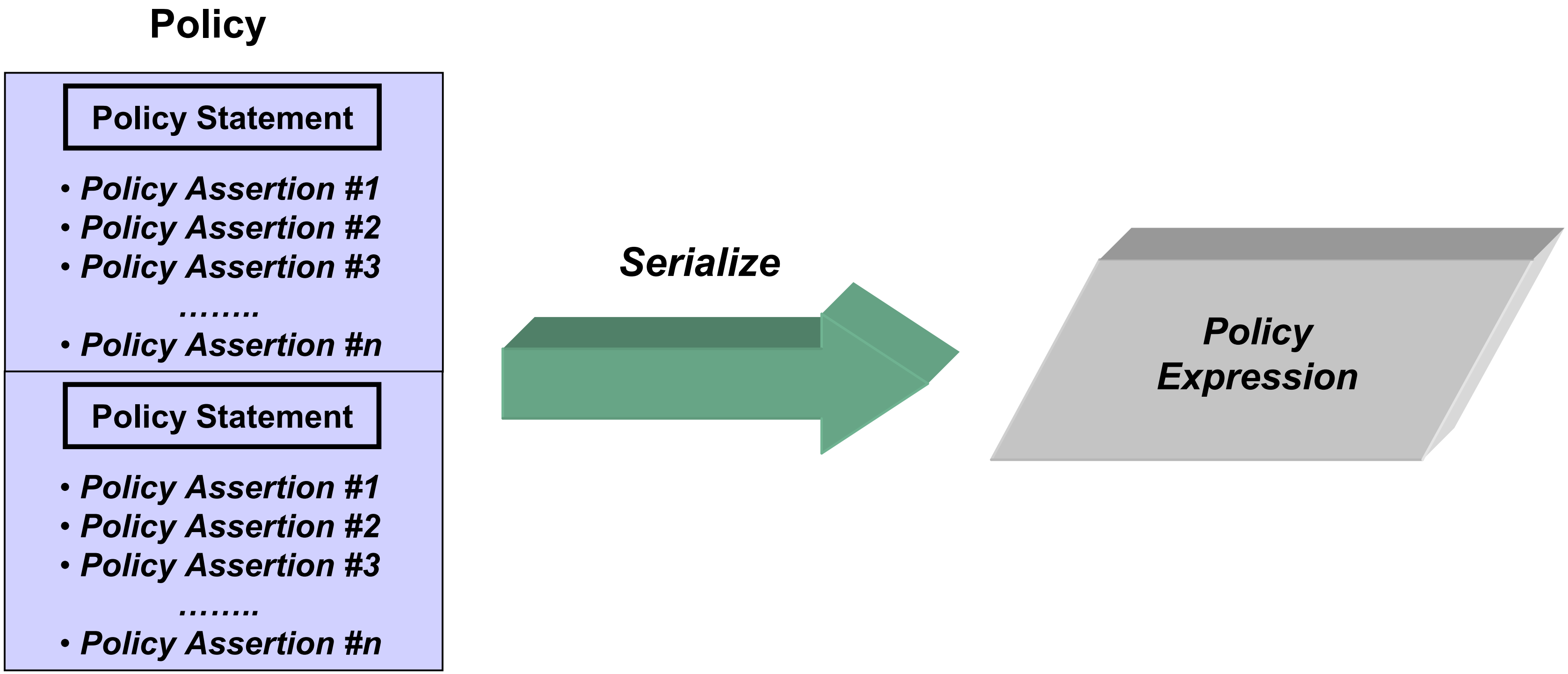
http://msdn.microsoft.com/ws/2002/12/ws-security-policy

WS-Policy

WS-Policy provides a framework for specifying and discovering the capabilities and requirements of a Web service

- Defines a framework and model for the **expression of these capabilities and requirements as policies**
- Specification released **December 2002** by Microsoft, IBM, BEA Systems, and SAP
- Terms:
  - *Policy Statement* – a group of **policy assertions**
  - *Policy* – a set of domain-specific **policy statements**
  - *Policy Expression* – an **XML serialization** that represents one or more **policy statements**

A policy is serialized into an XML representation, a "Policy Expression"



An XML Example

X.509 certificates and Kerberos tickets are accepted by this Web service, with X.509 certificates preferred

- Policy Expression using SecurityToken assertions:

```
<wsp:Policy>
   <wsp:ExactlyOne>
           <wsse:SecurityToken TokenType=”wsse:X509v3”
           wsp:Usage="wsp:Required" wsp:Preference=”50”/>
           <wsse:SecurityToken TokenType=”wsse:Kerberosv5TGT”
              wsp:Usage="wsp:Required" wsp:Preference="10"/>
   </wsp:ExactlyOne>
</wsp:Policy>
```

- Specification:

http://msdn.microsoft.com/ws/2002/12/Policy

Potential E-Government Applicability

- May have applicability to E-Government initiatives (such as E-Grants) for **defining capabilities and requirements as policies**
    - *Ex:* Specify **accepted security tokens** and **preference levels**

WS-PolicyAssertions

WS-PolicyAssertions defines general message-related assertions for use with WS-Policy

- Specification released **December 2002** by Microsoft, IBM, BEA Systems, and SAP

- Types of assertions:
  - ***TextEncoding assertion:*** Indicates which **character encodings** *(e.g. ISO-8859-1, UTF-8, UTF-16)* are supported by a Web service
  - ***Language assertion:*** Specifies supported **natural languages**
  - ***SpecVersion assertion:*** Indicates which **versions of a specification** a Web service supports
  - ***MessagePredicate assertion:*** Expresses **predicates (pre-conditions)** to which a message must conform

An XML Example

- ***MessagePredicate assertion:***

```
<wsp:MessagePredicate wsp:Usage="wsp:Required">
   count(wsp:GetHeader(.)/wsse:Security) = 1
</wsp:MessagePredicate>
```

Messages to which this assertion applies must contain exactly one WS-Security <Security> header element

- Specification:

http://msdn.microsoft.com/ws/2002/12/PolicyAssertions

WS-PolicyAttachment

WS-PolicyAttachment defines how to associate policy expressions with WSDL type definitions and UDDI entities

- Specifically, it defines:
  - How to reference policies from **WSDL definitions**
  - How to associate policies with **specific instances of WSDL services**
  - How to associate policies with **UDDI entities**
- Specification released **December 2002** by Microsoft, IBM, BEA Systems, and SAP

An XML Example

- Associating a policy expression with a **WDSL endpoint**:

```
<wsp:PolicyAttachment>
   <wsp:AppliesTo>
        <wsp:EndpointReference>
            <wsp:ServiceName Name="InventoryService"/>
            <wsp:PortType Name="InventoryPortType"/>
            <wsp:Address URI="http://www.xyz.com/acct"
        </wsp:EndpointReference>
   </wsp:AppliesTo>
   <wsp:PolicyReference Ref="http://www.xyz.com/acct-
                               policy.xml"/>
</wsp:PolicyAttachment>
```

This policy expression applies to all output resources of a service that implement the specified PortType

- Can also associate policy expressions with ***wsdl:message*** and ***wsdl:part*** elements

Implementations may register a specific WS-Policy expression in a UDDI registry as a distinct tModel

- Can associate WS-PolicyAttachment–based policy expressions with **entities in a UDDI registry**

- ***An XML Example*** - Associating a policy expression with an entity in a UDDI registry **using a predefined tModel**:

```
<tModel tModelKey="uuid:bd3966a8-faa5-416e-9772-
                     128554343571">
      <name>http://schemas.xmlsoap.org/ws/2002/07/
                                       policytmodel</name>
      <description>WS-PolicyAttachment policy
                                  expression</description>
</tModel>
```

Another XML Example

- Can associate a policy expression with a **businessService** using the service's categoryBag:

```
<businessService>
    <name>MyService</name>
    <description>This is a service that…</description>
    <bindingTemplates>
            ……………
    </bindingTemplates>
    <categoryBag>
        <keyedReference
          tModelKey="uuid:bd3966a8-faa5-416e-9772-
                        128554343571"
          keyName="http://schemas.xmlsoap.org/ws/
                        2002/07/policytmodel"
          keyValue="http://www.example.com/
                        myservice/policy"/>
        </keyedReference>
    </categoryBag>
</businessService>
```

The "tModelKey" represents the categorization system, while the "keyValue" contains the actual categorization

- Specification:

  http://msdn.microsoft.com/ws/2002/12/PolicyAttachment

Potential E-Government Applicability

- May have applicability to E-Government initiatives (such as GovBenefits) as mechanism for **associating policies with the WSDL endpoints that identify their services**, as well as the WSDL messages associated with those endpoints
  - Policies could range from **natural language requirements** (that a message must support Spanish) to **security policies**

WS-Trust

WS-Trust defines protocols for issuing security tokens and managing trust relationships

- *Trust* – “The characteristic that one entity is **willing to rely upon** a second entity to **execute a set of actions** and/or **make a set of assertions** about a set of subjects and/or scopes” - *WS-Trust Specification*

- Specification released **December 2002** by Microsoft, IBM, Verisign, and RSA Security

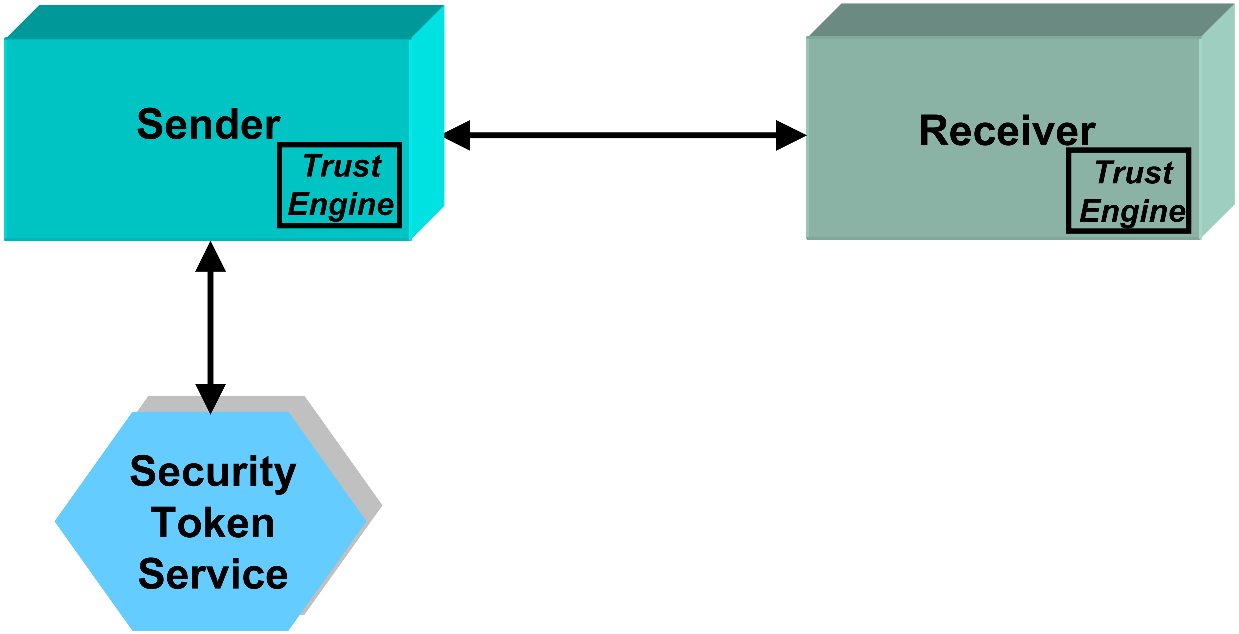
- In order to secure a communication between 2 parties, the 2 parties must **exchange security credentials** *(either directly or indirectly)*
  - However, each party needs to **determine if they can “trust”** the asserted credentials of the other party

A “Trust Engine” is a conceptual component of a Web service that evaluations the security-related aspects of a message

- A Trust Engine performs the following functions:
  - Verifies that the claims in the token are **sufficient to comply with the policy** and that the **message conforms to the policy**
  - Verifies that the attributes of the claimant are **proven by the signatures**
  - Verifies that the issuers of the security tokens are **trusted to issue the claims they have made**

A “Security Token Service” is a Web service that issues security tokens based on trust

- Transmission using Trust Engine and Secure Token Service:



Some XML Examples

- Requesting/returning a security token:

```
 <wsse:RequestSecurityToken>
         <wsse:TokenType>wsse:X509v3</wsse:TokenType>
         <wsse:RequestType>wsse:ReqIssue</wsse:RequestType>
</wsse:RequestSecurityToken>
```

Request for X.509 certificate

```
<wsse:RequestSecurityTokenResponse>
         <wsse:RequestedSecurityToken>
             <wsse:BinarySecurityToken
                   ValueType="wsse:X509v3"
                   EncodingType="wsse:Base64Binary">
                MIIEZzCCA9CgAwIBAgIQEmtJZc0...
             </wsse:BinarySecurityToken>
        </wsse:RequestedSecurityToken>
</wsse:RequestSecurityTokenResponse>
```

Response with certificate

In some cases, a Security Token Service may choose to challenge the requestor of a security token

- For example, the recipient **does not trust the nonce and timestamp** and issues a **<RequestSecurityTokenResponse>** message with an embedded challenge

- May also challenge the **signature:**

```
<wsse:SignChallenge>
    <wsse:Challenge>…Describes message parts that must be
                       signed…</wsse:Challenge>
     <wsse:SecurityTokenReference>...
    </wsse:SecurityTokenReference>
</wsse:SignChallenge>
```

- Specification:
  http://msdn.microsoft.com/ws/2002/12/ws-trust

Potential E-Government Applicability

- May have applicability to E-Government initiatives (such as Federal Asset Sales) for **issuance of security tokens to users** based on trust requirements
  - *Ex:* State Agencies for Surplus Property (SASP) that receive donated property

WS-Routing

WS-Routing is a simple, stateless, protocol for routing SOAP messages over a variety of transports such as TCP, UDP, and HTTP

- Entire path for a SOAP message *(as well as its return path)* can be **described directly within the SOAP envelope**
- Specification released **October 2001** by Microsoft
- Protocols such as HTTP and SMTP **define their own message path models and message exchange patterns** that differ from the SOAP message model
    - Not possible to use these protocol bindings alone to describe the exchange of a SOAP message from one point to another
- ***SOAP Router*** – a SOAP node that **exposes SOAP message relaying as a Web service**, either as a standalone service or in combination with other services

An XML Example

- Specifying intermediaries:

Messages from A to D will pass through B and C

```
<SOAP-ENV:Header>
   <wsrp:path>
       <wsrp:action>http://www.im.org/chat</wsrp:action>
       <wsrp:to>soap://D.com/some/endpoint</wsrp:to>
       <wsrp:fwd>
          <wsrp:via>soap://B.com</wsrp:via>
          <wsrp:via>soap://C.com</wsrp:via>
       </wsrp:fwd>
       <wsrp:from>soap://A.com/some/endpoint</wsrp:from>
       <wsrp:id>uuid:84b9f5d0-33fb-4a81-b02b-
                                 5b760641c1d6</wsrp:id>
   </wsrp:path>
</SOAP-ENV:Header>
```

- Specification:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-routing.asp

WS-Referral

WS-Referral is a stateless protocol for inserting, deleting, and querying routing entries in a SOAP router

- Enables dynamic route configuration
- Specification released **October 2001** by Microsoft
- While WS-Routing defines a message path *("send message from A to C via B")*, WS-Referral **enables route configuration** – i.e. how does A know about B?

A Referral Statement is an XML-based structure that describes a routing entry along with a set of conditions under which the statement is satisfied

- Each Referral Statement contains 5 parts:
  - A set of **SOAP actors** for which a statement is intended
  - A set of **conditions** that have to be met for a statement to be satisfied
  - Descriptive information
  - A **statement identifier**
  - A set of **SOAP routers** that a statement is referring to as part of the delegation

Some XML Examples

- Referral Statement:

```
<r:ref xmlns:r="http://schemas.xmlsoap.org/ws/
                2001/10/referral">
    <r:for>
         <r:prefix>soap://b.org</r:prefix>
    </r:for>
    <r:if>
         <r:ttl>43200000</r:ttl>
    </r:if>
    <r:go>
         <r:via>soap://c.org</r:via>
    </r:go>
    <r:refId>mid:1234@some.host.org</r:refId>
</r:ref>
```

For any SOAP actor starting with the specified prefix, if the referral is less than 12 hours old, then go via "soap://c:org"

Dynamic Routing: WS-Referral

- Referral query/response:

```
<S:Body>
    <wsr:query>
        <wsr:for>
             <wsr:prefix>soap://a.org</wsr:prefix>
        </wsr:for>
    </wsr:query>
</S:Body>
<S:Body>
   <wsr:queryResponse>
        <wsr:ref>
             …referral statement appears here…
        </wsr:ref>
   </wsr:queryResponse>
</S:Body>
```

Request referral statement for "soap://a.org"

Response with referral statement

WS-Referral can be useful in multiple cases

- For example:
  - “DNS-like” services
    - To notify other Web services that a Web service’s network address has changed
  - Load balancing
    - A SOAP router is too busy to handle the message – can reroute
  - Message path optimization
    - A “better” path suddenly exists
  - Delegation/message forwarding
- Specification:

  http://msdn.microsoft.com/webservices/understanding/gxa/default.aspx?pull=/library/en-us/dnglobspec/html/ws-referral.asp

Potential E-Government Applicability

- May have applicability to E-Government initiatives (such as E-Travel) for load balancing
  - *Ex:* Can **automatically/seamlessly reroute users** to another SOAP node when necessary for load balancing purposes

WS-Transaction

WS-Transaction specifies transactional properties of Web services

- Specification released **August 2002** by Microsoft, IBM and BEA Systems
- Utilizes 2 ***Coordination Types:***
  - Atomic Transaction
  - Business Activity
- *Atomic Transaction* – used to coordinate activities **having a short duration** and executed within limited trust
  - Has an **"all or nothing"** property
- *Business Activity* – used to coordinate activities that are **long in duration** and desire to apply business logic to handle business exceptions
  - Actions are **applied immediately and are permanent** because the long duration prohibits locking data resources

A Web services application can include both Atomic Transactions and Business Activities

- Each Coordination Type can have **multiple *Coordination Protocols***
  - Each is intended to **coordinate a different role** that a Web service plays in the activity
- Examples of Coordination Protocols:
  - *Completion* – a single participant tells the Coordinator to either **try to commit the transaction or force a rollback**
  - *2PC (2 Phase Commit)* – a participant such as a resource manager *(ex: database)* registers for this, so that the Coordinator can **manage a commit/abort decision** across all resource managers
  - *PhaseZero* – Coordinator notifies a participant **just before a 2PC protocol begins**
    - May need to write **cached updates** to a database prior to 2PC

A “Coordination Service” propagates/coordindates activities between services

- Messages exchanged between parties carry a **Coordination Context**
  - Contains information necessary to **link the various activities**
- Example of Coordination Context:

```
<S:Header>
   <wscoor:CoordinationContext>
          <wsu:Expires>
             2002-06-30T13:20:00.000-05:00
          </wsu:Expires>
           <wsu:Identifier>
              http://abc.com
          </wsu:Identifier>
          <wscoor:CoordinationType>
              http://schemas.xmlsoap.org/ws/2002/08/wstx
          </wscoor:CoordinationType>
          <wscoor:RegistrationService>
              <wsu:Address>
                   http://xyzregistrationservice.com
              </wsu:Address>
          </wscoor:RegistrationService>
     </wscoor:CoordinationContext>
  .....
</S:Header>
```

The CoordinationType “wstx” denotes an Atomic Transaction. The Registration Service will be discussed shortly.

A Coordination Service consists of several components

- Coordination Service consists of:
  - Activation Service – allows a **Coordination Context** to be created
  - Registration Service – allows a Web service to **register** to participate in a Coordination Protocol
  - A set of **Coordination Protocol Services** for each supported Coordination Type *(Completion, 2PC, etc.)*

Abbreviated Example – Atomic Transaction Process

- App1 **sends a *CreateCoordinationContext* message** to its local Activation Service to create an Atomic Transaction
- App1 **receives a Coordination Context** containing the following information:
  - Transaction Identifier
  - Coordination Type
  - Coordinator Port Reference
- App1 **registers with the Coordinator** for the "Completion" Coordination Protocol
- App1 **sends a message to App2** containing the Coordination Context

Abbreviated Example – Atomic Transaction Process

- App2 is an **application that caches data** – it registers with the Coordinator for the **“PhaseZero”** Coordination Protocol

- App2 **sends a message to App3** containing the Coordination Context

- App3 is a **resource manager** – it registers with the Coordinator for the **“2PC”** Coordination Protocol

At this point the Coordinator knows all the participants and what Coordination Protocols they expect to use

- Specification:

  http://msdn.microsoft.com/webservices/understanding/gxa/default.aspx?pull=/library/en-us/dnglobspec/html/ws-transaction.asp

Potential E-Government Applicability

- May have applicability to E-Government initiatives (such as Pay.gov) for **transactional processing**
  - *Ex:* Ensure that activities (such as payments) are **carried out in an atomic ("all-or-nothing") manner**

Remaining Specifications

Remaining Specifications

- WS-Coordination:
  - Defines **Coordination Types** used in WS-Transaction
  - Specification: http://msdn.microsoft.com/webservices/understanding/gxa/default.aspx?pull=/library/en-us/dnglobspec/html/ws-coordination.asp
- WS-Inspection:
  - Defines a **Web Services Inspection Language** for inspecting a Web site for available services
  - Specification: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-inspection.asp

Remaining Specifications

- WS-SecureConversation:
  - Defines mechanisms for establishing security context **using session keys, derived keys, and per-message keys**
  - Specification: http://msdn.microsoft.com/ws/2002/12/ws-secure-conversation/
- DIME *(Direct Internet Message Encapsulation)*:
  - Defines a **binary packaging format** for SOAP messages with attachments
  - Specification: http://www.ietf.org/internet-drafts/draft-nielsen-dime-02.txt

Remaining Specifications

- WS-Attachments:
  - Defines how DIME packaging can be used to **provide the attachment capabilities** needed by Web services
  - Specification: http://www.ietf.org/internet-drafts/draft-nielsen-dime-soap-01.txt
- WS-Privacy *(Pending)*
- WS-Federation *(Pending)*
- WS-Authorization *(Pending)*

Conclusions

The Global XML Web Services Architecture is poised to play a ***major role*** *in advancing the adoption of Web services through its* ***robust specification*** *of mechanisms for Web services such as* ***security, policy, coordination, federation, and routing.***

Several GXA specifications (WS-Transaction, WS-Coordination) appear to be ***plausible likely candidates*** *for inclusion in W3C's upcoming* ***Web Services Choreography Language Specification.***

QUESTIONS?

Contact Information

Joseph M. Chiusano

Booz | Allen | Hamilton
McLean, VA
(703) 902-6923
chiusano_joseph@bah.com