Web Services Advanced Topics
Beyond SOAP, WSDL, and UDDI (Part 1)

Kelvin R. Lawrence
CTO, Emerging Internet Software Standards
IBM Software Group
klawrenc@us.ibm.com

Related Talks This Week

- Other talks this week that are related to this one (there may be others) ...

- Arthur Ryman
  - Developing Web Services with Eclipse
  - What’s New in WSDL 2.0?

- Michael Perks (was Tony Nadalin)
  - SOA Security Programming Model
  - Model Driven Security Architecture

- Paul Freemantle
  - Apache Axis2 – The New Generation of Open Source Web Services

- Neil Graham
  - Who’s Afraid of XML Schema?

- Andre Tost
  - Practical Lessons Learned in Web Services Design and Implementation
  - Service Data Objects
Agenda (Parts 1 and 2)

- An overview of several new technologies for Web Services:
  - The Web Services “stack” of technologies
    - A quick update on the basic web services specs.
  - Detailed look at some advanced web services topics:
    - Security and the Security Roadmap
    - Policy
    - Trust, Secure Conversation and Federation
    - Addressing
    - Reliable Messaging
    - Transactions
    - Management
    - Business Process Modeling and Execution
  - Web Services and Interoperability
    - WS-I Status and work in progress
    - Industry profiles
# Web Services – a Simple View

## Business Processes

<table>
<thead>
<tr>
<th>Business Process Execution Language For Web Services (BPEL4WS)</th>
</tr>
</thead>
</table>

## Quality of Service

<table>
<thead>
<tr>
<th>Reliability</th>
<th>Transactions</th>
<th>Management</th>
<th>Security</th>
</tr>
</thead>
</table>

## Description

<table>
<thead>
<tr>
<th>Web Services Description Language (WSDL)</th>
</tr>
</thead>
</table>

## Messaging

<table>
<thead>
<tr>
<th>Simple Object Access Protocol (SOAP)</th>
<th>Extensible Markup Language (XML)</th>
<th>Other Protocols Other Services</th>
</tr>
</thead>
</table>

## Progress in 2005:

- **WS Reliable Exchange (WS-RX) TC Formed at OASIS, May 2005**
  - Reliable message exchanges between two Web Services
- **OASIS WS-Security Interop Event at Gartner conference, April 2005**
  - 14 companies demonstrated interoperable WS-Security implementations
- **WS Distributed Management approved by OASIS, March 2005**
  - Management of Web services & Management Using Web services
- **WS Trust, SecureConversation, SecurityPolicy, WS-AT, WS-BA, WS-C submitted to OASIS**
- **RAMP Profile published**
Aidon Jennery — Stop. Go. No, stop! How many ways are there to block a thread in Java?

Kelvin Lawrence — Web Services Advanced Topics: Beyond SOAP, WSDL, and UDDI (Part 1)
Technologies Discussed In This Session

- Business Process Execution Language (BPEL)
- WS-Coordination
- WS-TXanctions
- WS-Security family of specifications
- WS-Reliable Messaging
- WS-Distributed Management
- WS-Policy
- UDDI
- WSDL
- SOAP, SOAP Attachments
- Other protocols
- Other services
- XML, XML Infoset
- Transports
- Description and Discovery
- Messaging and Encoding
- Business Processes
- Quality of Service
Quick SOAP, WSDL & UDDI Update
Basic Web Services (SOAP, WSDL, UDDI)

**SOAP** uses XML messages for a request and response model of conversation between programs.

**WSDL** describes the interface a requester uses to invoke a service.

Development tools use the WSDL document to generate SOAP code automatically.

**UDDI** can be used to publish details of one or more services.
SOAP Message Structure

One way messages, or Request and Response style messages
- Request invokes a method on a remote object
- Response returns result of running the method

REMINDER: SOAP is not just about RPC

SOAP specification defines an "envelope"
- "envelope" wraps the message itself
- the “envelope” contains a header (optional) and a body
- message is a different vocabulary
- namespace prefix is used to distinguish the two parts

Application specific Vocabulary.

SOAP Envelope Vocabulary.
A SOAP Request Message

```xml
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Header>…</s:Header>
  <s:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>IBM</symbol>
    </m:GetLastTradePrice>
  </s:Body>
</s:Envelope>

SOAP Envelope

App. specific message
Status of SOAP, WSDL and UDDI

- **SOAP 1.1, WSDL 1.1 and UDDI 2.0** widely deployed today
  - Covered by WS-I Basic Profile 1.x (covered in detail in Part 2)

- W3C published the **SOAP 1.2 Recommendation**
  - June 2003
  - “Recommendation” status means finished, a W3C standard
  - Specs available at [http://www.w3.org/TR/soap](http://www.w3.org/TR/soap)
    - SOAP Version 1.2 Part 0: Primer
    - SOAP Version 1.2 Part 1: Messaging Framework
    - SOAP Version 1.2 Part 2: Adjuncts

- W3C Published **WSDL 2.0 Last Call document**
  - August 2005

- **UDDI 3.0** declared an OASIS Standard (2.0 already an OASIS standard)
  - February 2005
Web Services and Security
Web Services and Security

Why HTTPS Is Not Enough for Web Services

- HTTPS is transport-level security
  - Point-to-point: lasts only for duration of the connection
  - “All or nothing” encryption only
  - Weak integrity concept
  - Does not support other security mechanisms
Security Considerations with SOAP Messaging

- How to include security credentials in the message
- How to use element-wise encryption: expose some parts for routing, hide critical data from unauthorized parties
- How to use digital signatures
- Security must persist from originator to processing end-point, for the life of the transaction
- Security survives call to external business partner
- Use with, or instead of, protocol-level security
WS-Security: SOAP Message Security

- A foundational set of SOAP message extensions for building secure Web services
  - Defines new elements to be used in SOAP header for message-level security
- Defines the use of formerly incompatible proven and emerging security technologies:
  - Kerberos, PKI, HTTPS, IPSEC, XrML
  - XML Signature, XML Encryption, XKMS from W3C
  - SAML, XACML from OASIS
- OASIS WS-Security 1.0 standard
  - Widely supported in application servers and development tools from several vendors, including IBM, Microsoft, BEA, ...
WS-Security: SOAP Message Security

- Flexible, composable specification
  - Designed to be used as basis for securing Web services
  - Supports a wide variety of industry security models including PKI, Kerberos, and SSL

- Builds upon existing security technologies
  - XML DSIG
  - XML Encryption

- Provides support for:
  - multiple security token formats
  - multiple trust domains
  - multiple signature formats
  - multiple encryption technologies
SOAP Message Structure with Security Added

- The SOAP specification defines the “envelope” vocabulary
  - The "envelope" wraps the message itself
  - The message is a different vocabulary
  - A namespace prefix is used to distinguish vocabularies

- WS-Security defines the `<Security>` element, which allows security extensions to be placed in `<soapenv:header>`
  - Username/password
  - Encryption details
  - XML Signature
  - x.509 certificate
  - Kerberos ticket
  - Rights (REL)
  - SAML
The WS-Security Namespaces

In the following examples you will see the WS-Security namespaces used (wsse: or wsu: prefix):

- The OASIS namespace URLs are too long to fit in the examples cleanly, so for reference here they are:

  - WSSE (Web Services Security Extension)
    - http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-secext-1.0.xsd
  - WSU (Web Services Utility)
    - http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-utility-1.0.xsd
The WS-Security `<Security>` element

- The WS-Security specification defines a vocabulary that can be used inside the SOAP envelope. `<wsse:Security>` is the “container” for security-related information.

```
<S:Envelope xmlns:S="http://www.w3.org/2002/06/soap-envelope">
  <S:Header>
    <wsse:Security xmlns:wsse="...">
      Security information
    </wsse:Security>
  </S:Header>
  <S:Body>
    App-specific content
  </S:Body>
</S:Envelope>
```

Define and use WS-Security namespace

SOAP Envelope
Security Tokens for the <Security> Element

- A **Security Token** is a collection of one or more “claims”
  - A **claim** is a declaration made by some entity, such as name, identity, key, group, privilege, capability, etc.
  - “username” is an example of an unsigned security token

- A **Signed Security Token** is one that is cryptographically signed by a specific authority
  - An X.509 certificate is a signed security token
  - A Kerberos ticket is also a signed security token

- An **XML Security Token** is one that is defined with a separate XML schema, rather than simple or encrypted text
  - SAML and XrML are examples
  - Can be included directly in <wsse:Security> container
The `<UsernameToken>` element

- This element can be used to provide a user name within a `<wsse:Security>` element, for **Basic Authentication**

```xml
<S:Envelope xmlns:S="http://www.w3.org/2002/06/soap-envelope">
  <S:Header>
    <wsse:Security xmlns:wsse="...">
      <wsse:UsernameToken wsu:ID="myToken">
        <wsse:Username>kelvin</wsse:Username>
        <wsse:Password>elephant</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </S:Header>
  <S:Body>
    App-specific content
  </S:Body>
</S:Envelope>
```
The `<BinarySecurityToken>` element

- Signed security tokens, such as a Kerberos ticket or x.509 certificate, are binary content. They must be encoded for inclusion in the `wsse:Security` container.

```xml
<S:Envelope
 xmlns:S="http://www.w3.org/2002/06/soap-envelope">
 <S:Header>
  <wsse:Security
   xmlns:wsse="...">
   <wsse:BinarySecurityToken wsu:ID="myToken"
    ValueType="wsse:Kerberosv5ST"
    EncodingType="wsse:Base64Binary">
    XIFNWZz99UUbalqIEmJZc0
   </wsse:BinarySecurityToken>
  </wsse:Security>
 </S:Header>
 <S:Body>
  App-specific content
 </S:Body>
</S:Envelope>
```
XML Digital Signature Standard

- The XML Digital Signature standard defines rules for creating a digital signature and representing that signature as XML content
  - XML-Signature Syntax and Processing 1.0: W3C Recommendation, February 2002
  - [http://www.w3.org/Signature/](http://www.w3.org/Signature/)
  - Definition of schema for the signature (KeyInfo)
  - Procedures for computing and for verifying such signatures
  - Signature survives parsing/generation operations
  - Sign entire document, portions, or combinations of these
  - Can create multiple signatures with arbitrary keys

- Related specification: XML Exclusive Canonicalization
  - Specifies order of processing in computing a signature
  - [http://www.w3.org/TR/xml-exc-c14n/](http://www.w3.org/TR/xml-exc-c14n/)
XML Digital Signature

- Provides proof of **integrity** of XML content
  - The signed data has not changed since it was sent
  - Does NOT provide confidentiality
- Based on hash functions and encryption
  1. Generate a hash from the data to be signed
  2. Encrypt the digest to create the signature
  3. The signature is sent with original content for verification purposes
- To verify the signature
  1. Regenerate a digest of the original data that was signed
  2. Decrypt the first encrypted digest (*i.e.* the signature)
  3. Compare the two digests; a match verifies the content
- Along with Auditing, XML Digital Signature gives us Non-repudiation
- We’ll look at signatures from a Web services perspective in a moment
Hash Functions

- A **hash** or **message digest** function reduces an arbitrary stream of bytes to a fixed-size number
  - the number is usually 128 or 160 bits in length

- It has two important properties:
  1. Any change to the original input stream, even a small change, will produce a change in the hash code
  2. Given an input stream and its hash code, it’s practically impossible to find a second stream with the same hash code

![Diagram of hash function]

---

**Message**

**Message Digest Algorithm**

**Message Digest**
General Digital Signature Processing

1. Public and private keys belong to the sender.
2. Signature appended to message and sent.
3. Compute new digest, decrypt signature, compare, valid if equal.

Sender

Message

Message Digest

Asymmetric Signature Algorithm

Private Key

Asymmetric Key Pair Generation

Receiver

Message Digest

Signature

Asymmetric Verification Algorithm

Public Key

Message

May be retrieved from a key registry.
XML Digital Signature

- An XML digital signature is stored in a `<Signature>` element
- It has three main parts
  - `<SignedInfo>` – Information about what is signed
  - `<SignatureValue>` – The value of the digital signature itself
  - `<KeyInfo>` – The public key used to verify the signature

- Steps:
  - Calculate a `<DigestValue>` and create `<Reference>` elements for data to be signed
  - Add `<DigestValue>` and `<Reference>` elements into `<SignedInfo>`
  - Sign the entire `<SignedInfo>` element to create a `<SignatureValue>` element
  - Add `<SignedInfo>`, `<SignatureValue>`, and `<KeyInfo>` to `<Signature>`
Example: XML Signature (No SOAP)

```xml
<SignedInfo>
  <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
  <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
  <Reference URI="#wssecurity_body_id_2601212934311668096_1040651106378">
    <Transforms>
      <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    </Transforms>
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <DigestValue>AWQKpmksMpzzT4PxcizO980gVHw=/</DigestValue>
  </Reference>
</SignedInfo>

<SignatureValue>bNhT+DsNN9PR [binary data has been truncated]</SignatureValue>

<KeyInfo>
  <wsse:SecurityTokenReference>
    <wsse:Reference URI="#wssecurity_binary_security_token_id_1603091_4272645"/>
  </wsse:SecurityTokenReference>
</KeyInfo>

</Signature>
```
Using XML Digital Signatures with SOAP

- As we have seen, XML Digital Signatures tells us how to sign arbitrary XML content

- How do we use XML Signatures with SOAP messages?
  - WS-Security defines a new element in the SOAP header to hold XML Signature(s) on the content
  - Standardization of these elements allows implementations from different vendors to interoperate with signatures
  - WS-I Basic Security Profile (work in progress) specifies usage details to ensure interoperability.
    - The bulk of the profile work is now complete.
    - WS-I still has to finish up the work on the testing tools and sample applications to get comfortable that remaining issues have been found/ironed out.
Example: SOAP with XML Signature

```xml
<S:Envelope>
  <S:Header>
    <wsse:Security S:mustUnderstand="1"
      xmlns:wsse="...">
      <wsse:BinarySecurityToken EncodingType="wsse:Base64Binary">
        MIIDQTCC4ZzO7tIgerPlaid1q ... [truncated]
      </wsse:BinarySecurityToken>
    </wsse:Security>
  </S:Header>

  <S:Body>
    <m:OrderAircraft quantity="1" type="777" config="Atlantic"
      xmlns:m="http://www.boeing.com/AircraftOrderSubmission"/>
  </S:Body>
</S:Envelope>
```

...see XML Signature example for full content...

---

Kelvin Lawrence — Web Services Advanced Topics: Beyond SOAP, WSDL, and UDDI (Part 1)
XML Encryption

- The XML Encryption standard defines ways to encrypt all or part of an XML document
  - The encrypted information is replaced with a single `<EncryptedData>` element
  - You can encrypt different parts of the same document with different keys
  - You can encrypt the whole document, a single element, or just the text of an element
Symmetric Encryption

- The same secret key is used to both encrypt and decrypt the message

![Symmetric Encryption Diagram]

Sender

1. Secret Key

2. Plain text message

3. Symmetric Cipher Encrypt

4. Cipher text

5. Symmetric Cipher Decrypt

Receiver

Plain text message
Symmetric Encryption

- Fast

- Common algorithms are Triple DES (3DES), AES, ...

- Drawback: the key must remain secret, and it must be distributed securely to anyone we want to talk with
  - If we want secure conversations with $n$ partners, we have to distribute $n$ keys to them

- If the partner is local, we can hand them the key on any convenient digital media

- But if they are distant, this isn’t convenient, and we can’t safely send it to them using the Internet!
Asymmetric Encryption

- Each owner has a pair of complementary keys
  - They are different from each other
  - Encrypt with one, decrypt only with the other (in either direction)
  - We give one away (the Public key) and keep the other secret (the Private key)
  - If anyone encrypts a message with our public key, only we can decrypt the message (with our private key)
  - Conversely, if we encrypt a message with our private key, only our public key will decrypt it. So...
    - If a recipient successfully decrypts that message with our public key, they know we sent the message

- Drawback: asymmetric encryption is slower than symmetric encryption
Asymmetric Encryption

- Encrypt with the receiver's Public Key – only the receiver can decrypt the message

Public Key Cryptography, the basis for PKI
Hybrid Approaches

- Cryptography applications frequently combine these approaches
  - SSL is an example: It uses asymmetric encryption to setup the secret key, then uses symmetric encryption for the data transmission
- Most of the ISO security issues are solved with protocols based on combinations of symmetric encryption, asymmetric encryption, and hash functions
What’s in `<EncryptedData>`

- An `<EncryptedData>` element contains these elements
  - `<EncryptionMethod>` – The algorithm used to encrypt the data
  - `<KeyInfo>` – Information about the key used to encrypt the data
  - `<CipherData>` – Contains the
    - `<CipherValue>` element, which in turn
      - Contains the actual encrypted data

- As we'll see shortly, XML encryption in the context of Web services changes the format a little
W3C XML Encryption Specifications

- **Who:** W3C Working Group
  - http://www.w3.org/Encryption/
  - Started as joint proposal by IBM, Microsoft, Entrust

- **Purpose:**
  - Encrypting data and representing the result in XML
  - Can encrypt: an entire XML document, elements, element content, arbitrary data, or a combination of these
  - `<EncryptedData>` replaces encrypted element or content, or is the root of an encrypted document

- **Status:** W3C Recommendations, December 2002
  - XML Encryption Syntax and Processing 1.0
  - Decryption Transform for XML Signature 1.0

- **Availability:**
  - WebSphere 6
WS-Security Utilizes W3C XML Encryption

- `<EncryptedData>` element replaces the content being encrypted.
- It contains:
  - `<EncryptionMethod>` Algorithm used to encrypt the data
  - `<CipherData>`
    - `<CipherValue>` Element containing the encrypted data

- `<EncryptedKey>` element placed in security header, contains
  - `<EncryptionMethod>` Algorithm used to encrypt symmetric key
  - `<KeyInfo>` Identifier of key used to encrypt symmetric key
  - `<CipherData>`
    - `<CipherValue>` Encrypted symmetric key value
  - `<ReferenceList>` List of `<DataReference>`s to content encrypted with this symmetric key
Example: Entire `<body>` Contents Encrypted

```
<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PayBalanceDue>
```

Unencrypted original content
Red text is data to be encrypted
Green text is left unencrypted

```
<EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#' Type='http://www.isi.edu/in-notes/iana/assignments/media-types/text/xml'>
  <CipherData><CipherValue>A23B4C6</CipherValue></CipherData>
</EncryptedData>
```

"PayBalanceDue" element identity is hidden in encrypted form. We can't even see what kind of transaction it is!

(The real cipher would be longer than this)
Example: One Element and Sub-elements Encrypted

```xml
<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PayBalanceDue>

<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
    Type='http://www.isi.edu/in-notes/iana/assignments/media-types/text/xml'>
    <CipherData><CipherValue>A23B4C6</CipherValue></CipherData>
  </EncryptedData>
</PayBalanceDue>
```

Red text is data to be encrypted
Green text is left unencrypted

<CreditCard> group was replaced by <EncryptedData> element
Example: Element Text (Only) Encrypted

- `<PayBalanceDue xmlns='http://example.org/paymentv2'>`
- `<Name>John Smith</Name>`
- `<CreditCard Limit='5,000' Currency='USD'>`
  - `<Number>4018 2445 0277 5567</Number>`
- `<Issuer>Bank of the Internet</Issuer>`
- `<Expiration>04/02</Expiration>`
- `</CreditCard>`
- `</PayBalanceDue>`

Text was replaced by an EncryptedData element

Red text is data to be encrypted
Green text is left unencrypted
A More Complex WS-Security Example

- Demo: encryption2.xml
- Demo: Turning a Java class into a secured web service.
Status of WS-Security


- Latest WS-Security specifications and work in progress:
  - The OASIS Standard for WS-Security 1.0 was approved, April 2004
    - WS-Security (core) specification
    - Username Token Profile
    - X.509 Token Profile

  - The WSS Technical Committee has since produced
    - SAML Token Profile (OASIS Standard Dec 2004)
    - Rights Expression Language (REL) Token Profile (OASIS Standard Dec 2004)
    - WS-Security 1.0 Errata (non normative)

  - The WSS Technical Committee is working on
    - WS-Security 1.1 (errata, updates, new profiles)
      - Kerberos Token Profile
      - Soap With Attachments (SWA) Profile
    - Mobile Device (minimal) Profile
Welcome to the WSS TC home page.

Latest News  July 12th 2005

We have just begun a public review on the documents that will make up the 1.1 release of the specification. Full details are here.

OASIS Standard 1.0

- [Web Services Security, SOAP Message Security V1.0](#)
- [Web Services Security, Username Token Profile V1.0](#)
- [Web Services Security, X.509 Token Profile V1.0](#)
- Schema files V1.0 [1] and [2]

Note that all OASIS Standards documents are available from the standards page.
Web Services and Policy
The Policy specs have been evolving...

- WS-Policy has been simplified quite a bit.
- Some assertions are now defined in separate policies.
- Some of the elements and attributes that used to be in the spec have either been renamed or removed completely in the most recent drafts.
- The PolicyAssertions spec has been “deprecated”.
- WS-SecurityPolicy has been enhanced significantly.
- WS-RM Policy has evolved.
- The following slides now reflect these changes...
What Is a Policy?

- **A policy** is a set of capabilities, requirements, preferences, and general characteristics about entities in a system.

- The elements of a policy (**policy assertions**) can express:
  - Security requirements or capabilities
  - Various Quality of Service (QoS) characteristics
  - Any other kinds of policies that are required by a service

- **WS-Policy** defines a general purpose, extensible model and grammar ("framework") for describing policies in a Web services system.
  - Simple, declarative policies
  - More complex, conditional policies
WS-Policy defines the framework for policy definition

- The container element <Policy>
- The organizing operator elements
- The “Optional” attribute
- An inclusion / reuse mechanism

WS-Policy does NOT define:

- Any specific policy assertions. These are defined by, WS-SecurityPolicy, WS-RM Policy and others yet to be invented.
- The binding to a policy subject.
  - This is defined in WS-PolicyAttachment.
Old Policy Syntax (Earlier Spec Drafts)

```xml
<wsp:Policy xmlns:wsse="..." xmlns:wsp="...">
  <wsp:ExactlyOne>
    <wsse:SecurityToken TokenTypewsp:="wsse:Kerberosv5TGT"
      wsp:Usage="wsp:Required"
      wsp:Preference="100"/>
    <wsse:SecurityToken TokenTypewsp:="wsse:X509v3"
      wsp:Usage="wsp:Required"
      wsp:Preference="1"/>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Meaning:

1. A valid message must contain any of one the two SecurityTokens
2. Kerberos and X509 are supported authentication tokens
3. Kerberos is preferred over X509
New Policy Syntax (as of latest draft)

```xml
<wsp:Policy xmlns:wsse="..." xmlns:wsp="...">
  <wsp:ExactlyOne>
    <some-ns:assertion1 />
    <some-ns:assertion2 />
  </wsp:ExactlyOne>
</wsp:Policy>
```

Meaning:

1. Exactly one of the requirements must be met.
2. Note that there is no longer a way to express a preference.

**NOTE:** The security specific elements are being replaced/renamed by WS-SecurityPolicy (we’ll discuss this in a few minutes).
Policy Terminology

```
<wsp:Policy xmlns:wsse="..." xmlns:wsp="...">
  <wsp:ExactlyOne>
    <some-ns:assertion1/>
    <some-ns:assertion2/>
  </wsp:ExactlyOne>
</wsp:Policy>
```

1. **Policy assertion**: a preference, requirement, capability or other property
2. **Policy alternative**: a collection of policy assertions
3. **Policy**: a collection of policy alternatives
4. **Policy expression**: an XML infoset representation of a policy

A **policy subject** is an entity (e.g. an endpoint, object, or resource) to which a policy can be bound (see WS-PolicyAttachment).
Policy Operators

Operators can be **ExactlyOne**, or **All**. In this example:

- The *primary* operator **ExactlyOne**, is a policy statement (alternative)
- The *subordinate* operator **All**, groups two related policy assertions

```xml
<wsp:Policy xmlns:wsp="..." xmlns:wsse="...">
  <wsp:ExactlyOne>
    <wsp:All/>
    <some-ns:assertion1 />
    <some-ns:assertion2 />
  </wsp:All>
  <wsp:All/>
    <some-ns:assertion3 />
    <some-ns:assertion4 />
  </wsp:All>
</wsp:ExactlyOne>
</wsp:Policy>
```
Policy Preferences

```
<wsp:Policy xmlns:wsp="..." xmlns:wsse="...">
    <wsp:ExactlyOne>
        <wsp:All wsp:Preference="100">
            <wsse:SecurityToken TokenType="wsse:Kerberosv5TGT"/>
            <wsse:Algorithm Type="wsse:AlgSignature"
                URI="http://www.w3.org/2000/09/xmlenc#aes"/>
        </wsp:All>
        <wsp:All wsp:Preference="1">
            <wsse:SecurityToken TokenType="wsse:X509v3"/>
            <wsse:Algorithm Type="wsse:AlgEncryption"
                URI="http://www.w3.org/2001/04/xmlenc#3des-cbc"/>
        </wsp:All>
    </wsp:ExactlyOne>
</wsp:Policy>
```

- The Preference attribute indicates the preferred policy assertions among the listed choices.
- Larger numbers are preferred.
• The Usage attribute specifies how the assertion is used
  - Required – assertion must be applied to the subject; if not supported, fail.
  - Rejected – assertion is not allowed. If present, fail.
  - Optional – assertion may be made, but may not be applied.
  - Observed – assertion is applied, and requesters are informed that it is applied.
  - Ignored – assertion is processed, but ignored, and requestors are informed.

```xml
<wsp:Policy xmlns:wsse="..." xmlns:wsp="...">
  <wsp:ExactlyOne>
    <wsse:SecurityToken TokenType="wsse:Kerberosv5TGT"
                 wsp:Usage="wsp:Required"
                 wsp:Preference="100"/>
    <wsse:SecurityToken TokenType="wsse:X509v3"
                 wsp:Usage="wsp:Required"
                 wsp:Preference="1"/>
  </wsp:ExactlyOne>
</wsp:Policy>
```
The *Optional* Attribute

```
<wsp:Policy xmlns:wsse="..." xmlns:wsp="...">
  <wsrm:RMAssertion wsp:Optional="true"/>
</wsp:Policy>
```

Is shorthand for (and semantically equivalent to) …

```
<wsp:Policy xmlns:wsse="..." xmlns:wsp="...">
  <wsp:ExactlyOne>
    <wsp:All>
      <wsrm:RMAssertion />
    </wsp:All>
  </wsp:All>
  <wsp:All />
</wsp:ExactlyOne>
</wsp:Policy>
```

Note that a default value of “false” is always assumed unless specified otherwise.
Policy Inclusion

• `<wsp:PolicyReference>` allows assertions to be shared among policy expressions. It includes the content of one policy expression in another expression. In this example:
  • the `wsu:ID` attribute defines a reference to the `<wssx:Audit>` element
  • the `<wssx:Audit>` element effectively replaces the `<wsp:PolicyReference>` element in the policy statement.
Reusing a Portion of a Policy

```xml
<wsp:Policy xmlns:SecurityNS="..."
    xmlns: cus="...">
    <cus:Assert1>
        <wsp:ExactlyOne wsu:Id="options">
            <cus:Option1 />
            <cus:Option2 />
            <cus:Option3 />
        </wsp:ExactlyOne>
    </cus:Assert1>
    <cus:Assert2>
        <wsp:PolicyReference URI="#options"/>
    </cus:Assert2>
</wsp:Policy>
```

- The identification mechanism for `<wsp:PolicyReference>` can also be used with operator elements. In this example:
  - the `wsu:ID` attribute defines a reference to the `<wsp:ExactlyOne>` group
  - the `<wsp:ExactlyOne>` group effectively replaces the `<wsp:PolicyReference>` element in the policy statement.
WS-SecurityPolicy

- Business Process Execution Language
- Business Processes
- WS-Coordination
- WS-Transactions
- WS-Security
- WS-Security (Policy)
- WS-Security Policy
- WS-Security Policy (framework)
- WS-Policy
- WS-Policy (framework)
- WS-Policy Attachments
- WS-Policy Attachments
- Other protocols
- Other services
- UDDI
- UDDI
- WS-Security Policy
- WS-Security Policy (framework)
- WS-Security Policy Attachments
- WS-Security Policy Attachments
- WS-Security Policy (framework)
- WS-Security Policy Attachments
Security Policy Example

```xml
<wsp:Policy xmlns:wsp="...">
  <sp:SymmetricBinding />
  <wsp:Policy>
    <sp:ProtectionToken>
      <wsp:Policy>
        <sp:KerberosV5APREQToken
          sp:IncludeToken=".../IncludeToken/Once" />
      </wsp:Policy>
    </sp:ProtectionToken>
    <sp:SignBeforeEncrypting />
    <sp:EncryptSignature />
  </wsp:Policy>
</sp:SymmetricBinding>
<sp:SignedParts>
  <sp:Body/>
  <sp:Header
    Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
</sp:SignedParts>
<sp:EncryptedParts>
  <sp:Body/>
</sp:EncryptedParts>
</wsp:Policy>
```

Note that wsp:Policy can be nested to scope assertions and also that wsp:Policy includes an implicit wsp:All
WS-SecurityPolicy (Status)

- Spec (and schema) Submitted to OASIS October 2005

- Will be worked on by the new WS-SX TC
WS-PolicyAttachments

- Business Process Execution Language
- WS-Coordination
- WS-Transactions
- WSDL
- SOAP, SOAP Attachments
- XML, XML Infoset
- Transports
- WS-Policy
- WS-Security
- WS-Reliable Messaging
- UDDI

Description and Discovery
- WS-Policy
- WS-Policy (framework)
- WS-Policy Attachments
- WS-Policy Assertions

Quality of Service
- WS-Policy
- WS-Policy (framework)

Business Processes
- Other protocols
- Other services

Messaging and Encoding
- Messaging and Encoding

Transport
WS-PolicyAttachment Specification

- Defines means of associating a policy expression with one or more subjects or resources:
  - arbitrary XML element(s) (policy is defined as part of the definition of the subject)
  - arbitrary non-XML resource(s) (policy is externally bound)

- Describes the use of these mechanisms with WSDL and UDDI artifacts:
  - How to reference policies from WSDL definitions
    - Messages and PortTypes
  - How to associate policies with specific instances of WSDL services
    - Services and Ports
  - How to associate policies with UDDI entities
    - businessService and bindingTemplate
  - How to define a policy expression in a UDDI registry as a tModel

- Such *bindings* need to be able to be secured (so they can be trusted)
Attaching a Policy to a Deployed Endpoint

```xml
<wsp:PolicyAttachment>
  <wsp:AppliesTo>
    <wsa:EndpointReference xmlns:f="...">
      <wsa:Address>http://www.fabrikam123.com/acct</wsa:Address>
      ...
    </wsa:EndpointReference>
  </wsp:AppliesTo>

  <wsp:PolicyReference
    URI="http://www.fabrikam123.com/policies#DSIG" />

</wsp:PolicyAttachment>
```
WS-Policy Assertions
WS-Policy Assertions

- Defined a set of basic assertions for describing general processing semantics for Web services:
  - types of text encodings that are allowed, rejected, required, preferred
  - natural languages that are allowed, rejected, required, preferred
  - Web services specifications and version numbers to which the subject conforms
  - ensure that a message conforms to a given pre-condition

- **NOTE:** This spec is now considered “deprecated.”
  - The few basic assertions that were in the spec were not considered valuable enough to justify a dedicated spec just for them.
  - It is now left up to the domain specific policies to define the things they need.
Resources: Policy

- All specs are available on [http://ibm.com/developerworks](http://ibm.com/developerworks)
  - Search for WS-Policy to get the entire list

- Web Services Policy Framework

- Understanding WS-Policy processing

- Whitepaper: “Web Services Security: Moving up the stack”
Trust, Secure Conversation and Federation
Beyond Message Security
http://ibm.com/developerworks/webservices
WS-Trust

- A model for direct and brokered trust relationships
  - Third parties and intermediaries
  - Manage credentials across different trust domains

- Defines:
  - The “security token service”
    - A trusted authority for security tokens implemented as a Web service
  - SOAP messages sent to this service for security token issuance, validation and exchange (request/response).
WS-Trust Request/Response

Requesting a token:

```
<wst:RequestSecurityToken Context="...">
  <wst:TokenType>...</wst:TokenType>
  <wst:RequestType>...</wst:RequestType>
</wst:RequestSecurityToken>
```

Responding with a token:

```
<wst:RequestSecurityTokenResponse Context="...">
  <wst:TokenType>...</wst:TokenType>
  <wst:RequestedSecurityToken>...</wst:RequestedSecurityToken>
</wst:RequestSecurityTokenResponse>
```
WS-Secure Conversation

- Establish a secure, shared security context in which to exchange multiple messages
- Defines the mechanisms for
  - Establishing and sharing security contexts
  - Deriving session keys from security contexts

- Defines 3 ways of establishing a security context
  - Security context token created by a security token service
  - Security context token created by one of the communicating parties and propagated with a message
  - Security context token created through negotiation
SecurityContextToken Example

- **SecurityContext header**
- Identifies the security context using a URI
- Indicates the creation time of the security context
- Indicates the expiration time of the security context
- Holds the shared secrets of the security context
- References a shared secret of the security context

```xml
<wsse:SecurityContextToken wsu:Id="...">
  <wsu:Identifier>...</wsu:Identifier>
  <wsu:Created>...</wsu:Created>
  <wsu:Expires>...</wsu:Expires>
  <wsse:Keys>
    <xenc:EncryptedKey Id="...">...</xenc:EncryptedKey>
    <wsse:SecurityTokenReference>...</wsse:SecurityTokenReference>
    ...
  </wsse:Keys>
</wsse:SecurityContextToken>
```
Standards Update

- **WS-Trust & WS-SecureConversation**
  - Specs & schemas submitted to OASIS, October 17\textsuperscript{th} 2005
  - [http://schemas.xmlsoap.org/ws/2005/02/sc](http://schemas.xmlsoap.org/ws/2005/02/sc)

- **Will be worked on by the new WS-SX TC**
A *federation* is a collection of security realms (*e.g.* partner organizations) that have established trust to share security information about users belonging to the realms:

- identification, authentication
- attributes, authorization

The WS-Federation Specification:

- builds on the WS-Trust model
- can share this data using different or like mechanisms
- defines mechanisms for the brokering of trust and for security token exchange between trust domains
- does not require local identities at target services
- optionally allows hiding of identity info and other attributes
WS-Federation – Purpose

- Suppose:
  - A value network is composed of various organizations, systems, applications, and business processes.
  - Participants include customers, employees, partners, suppliers, and distributors.
  - There is no single entity for identity, authentication, authorization, *etc.*, because the cost of centralized identity management is high. Instead, there may be several such entities.
  - We need to manage security across multiple trust domains and among multiple business partners using multiple identity authorities.

- WS-Federation is a specification to solve this and other problems.
WS-Federation

- Other applications:
  - single-sign-on for users – don’t have to present new credentials *(e.g. a username and password)* when entering a new trust domain
  - multiple identity formats including pseudonyms
  - provision users between organizations
  - provision services between organizations
  - policy-driven trust management
Building on Other Security Technologies

- WS-Federation is not intended as a complete security solution.
- Instead, it builds on other Web services technologies:
  - WS-Policy specs can be used to indicate that a Web service requires a set of claims (security tokens and related message elements) in order to process an incoming request.
  - WS-Trust mechanisms can be used by the requester to acquire additional security tokens it may require.
  - WS-Security (WSS-SOAP Message Security) defines SOAP extensions used to provide security tokens.
  - WS-MetadataExchange defines a mechanism for exchanging policies, WSDL, and schemas for services within the federation.
Security Token Services

- A generic service that issues or exchanges security tokens using a common model and set of messages.
  - Follows the WS-Trust specification.
  - May be part of requester organization, provider organization, or a third party trusted by both of these.

- Common functions:
  - Verify credentials for entrance to a security realm
  - Evaluate the trust of supplied security tokens
  - Identity Provider – performs peer entity authentication and can make identity claims in issued security tokens
A Simple Direct Trust Federation Scenario

1. Security tokens from Requester’s organization are used to acquire security tokens from Provider’s organization which are required by the provider for the service request message.

The requester’s token is exchanged, stamped, or cross-certified by provider’s Security Token Service.
Another Direct Trust Federation Scenario

1. Security tokens from Requester’s organization are sent directly to provider’s service.
2. The service uses its Security Token Service to understand and validate the requester’s security token.
3. The validation response is sent as a security token which includes authentication and authorization data.
There may not be a direct trust relationship between requester and provider organizations. In that case, the two organizations may choose to use a trusted third party to establish and confirm trust for the transaction.

3. The provider asks the third party to verify the security token

4. The third party contacts the requester to verify the security token

Steps 1, 2, and 5 are as before.
Multi-party Federation

There might be several organizations involved in a business process, with multiple trust realms. Steps 4 and 5 are the same as 2 and 3, except they are for a different transaction from a different provider.
Delegation

A Web service provider may need to access another Web service on behalf of a requester. The delegator provides security tokens to allow or indicate proof of delegation. There are other possible variations on this scenario.

Diagram:

1. Requester’s organization
   - Security Token Service
   - Requester

2. Delegator’s organization
   - Security Token Service
   - Delegation
   - TRUST

3. Service Provider

4. TRUST

5. Provider 2
   - Security Token Service
   - Service Provider

A Web service provider may need to access another Web service on behalf of a requester. The delegator provides security tokens to allow or indicate proof of delegation. There are other possible variations on this scenario.
Attributes and Pseudonyms

- Privacy protection may require additional controls and mechanisms:
  - to provide access control on any private information
  - to prevent unwanted correlation
  - to automatically map identities
  - sharing of data between authorized parties

- WS-Federation defines mechanisms to support
  - **attribute services** to personalize the experience using restricted information (subject to authorization and privacy rules)
  - **pseudonym services** to facilitate single sign-on with automatic mapping of identities while keeping identity private
Resources: WS-Federation

http://ibm.com/developerworks/webservices

- Federation of Identities in a Web services world
  - Overview of goals and technologies

- Web Services Federation Language
  - The specification itself

- WS-Federation: Active Requestor Profile
- WS-Federation: Passive Requestor Profile
  - These specs define how the WS-Federation model is applied to active and passive requestors
Privacy refers to policies of how a service provider will use information supplied by a service requester.

WS-Privacy will be a model for how users state privacy preferences, and for how Web Services state and implement privacy practices.

Preferences are specified by way of policy assertions in the WS-Policy container.

More information: see Web Services Security Roadmap at

WS-Authorization Planned.

- Will describe how access policies for a Web service are specified and managed
- Will describe how claims may be specified within security tokens, and how these claims will be interpreted at the endpoint
  - More information: see Web Services Security Roadmap
WS-Security Implementation in WebSphere

**SOAP request header construction**
- Security token generation
- Digital signature generation
- Content encryption

**SOAP request header processing**
- Validate security tokens
- Set up security context
- Decrypt content
- Digital signature validation

**SOAP response header processing**
- Decrypt content
- Digital signature validation

**SOAP response header construction**
- Digital signature generation
- Content encryption
Deployment Descriptor for Security Requirements

- WS-Security requirements are specified as security constraints in the Deployment Descriptor:
  - Should the message be digitally signed or encrypted?
  - What is the trust mode for identity assertion?
  - What are the security tokens to be used as the caller identity?

- The Security Handlers act on the specified constraints to enforce WS-Security requirements

- This approach supports a separation of roles:
  - Developer of Web Service provider or requester app
  - Assembler or deployer of Web Service

- It also makes it easy to revise security requirements, since they are specified separately from the application code.
  - Microsoft .Net approach generates code to handle security; this is less flexible for dealing with changing security requirements
Time for a Break

END OF PART 1

We’ll continue this thread of discussion in the “Part 2” session. I hope to see you all there.