Web Services Advanced Topics

Where things are now and where they are going …

Version 9

Web Services – Advanced Topics

- Enterprise Web Services
- Industry trends and organizations
- Security and Reliability
- Transactions
- Choreography and Orchestration
  - Composite web services
- Interoperability Requirements
  - Encoding discussion
- Futures
Enterprise Web Services

• Extending WS throughout the enterprise and among enterprises ⇒ greater emphasis on Qualities Of Service (QoS)
  • Security, Reliability, Transactions, Scalability
  • Systems Management
  • Interoperability

• It’s also important to allow it to be used over more transports (other than SOAP over HTTP)
  – IIOP (CORBA)
  – MQ Series messaging
  – TIBCO Rendezvous
  – Tuxedo
  – SOAP over FTP
  – JMS (J2EE)

Enterprise Standards are Evolving

Others    ➔    WS-Eventing, WS-Notification, WS-Resource Framework
Management ➔    Web Services Distributed Management (WSDM)
Orchestration ➔    BPEL4WS, WS-Choreography
Transactions ➔    WS-CAF, WS-T & WS-C
Security ➔    XML Encryption, XKMS, XRI, WS-Security, SAML
Reliable Messaging ➔    WS-Reliability, WS-ReliableMessaging
Look-Up & Discovery ➔    UDDI
Service Contract ➔    WSDL
Message Payload ➔    SOAP
Industry organizations

- Helping to develop the necessary web services related standards
  - W3C
    - SOAP (XMLP working group), WSDL, WS Architecture, WS Choreography
  - OMG
    - IDL to WSDL, WSDL to IDL (WSDL 1.1), WSDL to C++
  - OASIS
    - UDDI, WS-Security, WS-Reliability, WS-CAF, etc.
  - JCP (Java Community Process)
    - JAX-RPC, JWSDL, JAXM, JAXR, ...
  - Others
    - WS-I, etc.

Security

- Confidentiality – no one can see my data
- Integrity – my data gets there in one piece
- Authentication – a user or program is confirmed to be who they say they are
- Authorization – ensuring access to services and data only by those who are supposed to access them
- No overall architecture yet defined
  - May include firewalls, proxies, security servers, and identity management
- WS-Security is a well-agreed framework (OASIS)
SOAP Header example (WS-Security)

```xml
<envelope xmlns:s11="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="...">
  <Header>
    <Security xmlns:wsse="...">
      <CustomToken wsu:Id="MyID" xmlns:fabtok="http://fabr123/token"/>
      <Signature> ... </Signature>
      <SignedInfo> ... </SignedInfo>
    </Security>
  </Header>
... 
</envelope>
```

Reliability requirements

- Reliability of message delivery is critical to many applications
  - Guarantee of message delivery (both request and response)
    - At least once (duplicates are allowed)
    - At most once ("best effort", ok to drop some messages)
    - Once and only once – the toughest to guarantee!
  - Eliminate duplicates (if any)
  - Preserve message ordering (when important)

- Some of these features are provided as standard when WS are carried on a connection-oriented protocol
  - The “sessions” and correlation IDs of the connection-oriented protocol ensures features such as ordering
  - But some QoS must still be added (e.g., store-and-forward to allow clients to send messages even if a server isn’t running)
Example message flow (WS-ReliableMessaging)

Reliability

Web Services: Implications for TP

- Persistent sessions are missing from HTTP
  - Need to define a common context sharing mechanism
  - Transport “agnostic” means LCD
    - It’s difficult to write a new WS spec that assumes a more sophisticated protocol than HTTP
- Need a solution that is also suited for document-oriented transactions
  - Compensation, transactional “queues” needed
  - Reliable messaging needed
- Business process orchestration
  - Cancel/adjust transactions in long-running flight
  - Mixture of existing and new technologies
Current status of standardization

- First attempt – SOAP-TIP (Transaction Internet Protocol)
  - Didn’t work because it assumed persistent sessions (and these aren’t provided by HTTP)
- BTP not widely implemented
  - A bit complex and hard to map to existing systems
- WS-Coordination and WS-Transaction not open
  - Difficult to implement because they are under-specified
  - and it’s not very interoperable
- WS-Composite Application Framework (WS-CAF)
  - Current proposal at OASIS
  - Is able to co-ordinate between different transaction systems
    - Resolves the differences in the co-ordinator, not in the end-points

Web Services Composition

- Once you have a “critical mass” of Web services, you may want to:
  - link them together to create (composite) complex Web services
  - implement whole business processes based on those services
- Of course you can “hard-code” this, but the result will be
  - inflexible (hard to maintain)
  - proprietary, not explicit (business logic will be buried in code)
- Better: Standardized high-level mechanisms for defining business processes
  - Standardized declarative, non-programmatic way to compose sequences of web service calls to accomplish a business function
- Efforts ongoing to provide
  - WS-BPEL (seems to be winning at present) for run-time
  - WS-Choreography (more B2B oriented) between partners
**Orchestration Architecture**

This overall sequence is viewed as a WS from outside.

You can also do transformation of data within this overall WS.

Often, you have selection and loops as well just sequencing.

Web service interfaces

Other applications

Message queuing system – A2B/B2B
Integration broker

.NET Server
Or App Server
With Web Service APIs

Database tables and stored procedures

Adapters

ERP, CRM, Accounting, etc.

**Interoperability Requirements**

- Within the enterprise and among enterprises
  - Across hardware platforms, operating systems, programming languages, middleware and component technologies

- SoapBuilders and WS-I have been working to correlate the many WS standards into more interoperable configurations. E.g.,
  - Defining a consistent selection of standards
  - Subsetting (“profiling”) a standard

- Ensuring compatible message encoding is another key part of interoperability
  - As discussed in the new few slides
You have a choice of the encoding for complex data in msgs

- The recommendation is literal
  - This uses XMLSchema in the normal way
    - The message references a schema that defines the elements and types for formatting arrays, sequences, records/structures, and other complex types
    - The message uses these elements and types
  - The alternative is SOAP encoded
    - This defines a set of elements that can be used for complex data
      - Mostly, these elements were good at formatting data that typically arises in RPC calls
      - It was first introduced before XMLSchema was available
        - Newer versions (SOAP 1.2) use XMLSchema.
          - The difference then between it and literal is that SOAP encoded fixes the set of tags/types. This has advantages and disadvantages.
      - Many existing Web Services use SOAP encoded, but literal is recommended by WS-I for SOAP 1.1
        - WS-I hasn’t yet profiled SOAP 1.2, so it is silent on whether SOAP encoding is OK in SOAP 1.2

You also have a choice between two message styles

- The choice is rpc or document
  - document style is recommended by WS-I for SOAP 1.1 (albeit most existing Web services examples use rpc style)
  - Recall that each operation has an input and an optional output message
    - In rpc style, each message can have zero or more parts
      - <message name="getQuoteRequest">
        - <part name="symbol" type="s:string">
        - <part name="date" type="s:string">
      </message>
    - In document style, each message can have at most one part (and this needs to be an XML element, not a type).

WS-I has yet to profile SOAP 1.2
Combining these two choices:

<table>
<thead>
<tr>
<th></th>
<th>document</th>
<th>RPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>literal</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SOAP encoded</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This combination is used by many existing services.

Futures – Quickly !!

- Advances in Standards
  - Ratification of newer versions of standards (SOAP, WSDL, etc.)
  - Emergence of newer standards to support the evolution of WS into enterprise environments (WS-CAF, etc.)

- Advances in deployment support
  - Additional availability of transports/protocols for greater penetration within enterprises without requiring infrastructure changes

- Interoperability advances
  - More shake-out of interoperability issues
  - More adoption of recommendations from WS-I, OASIS, etc