SOAP, WSDL, UDDI, JAX/RPC, WSIF, JSR109, WS-SECURITY, BPEL4WS, ... == Web Services

How Many Acronyms Does It Take to Make a Technology?

Denise Hatzidakis
Perficient, Inc.
denise.hatzidakis@perficient.com
So What Are We Going to Talk About?

- The Web Services Stack
- Key technologies
  - Foundation
  - Connection
  - Description and Discovery
  - Web Services Interoperability
  - Security
  - Reliability
  - Enterprise

We’re going to cover...

See Kelvin Lawrence’s “Web Services Advanced Topics: Beyond SOAP, WSDL and UDDI”
Chronology from 50K feet

Strategy, architecture, and first specs

Standardization & Interoperability

First Implementation

Product implementation

This is not the end of all standardization activities in the world!
Web Services Standards Evolution

Evolving to resolve pain ...

Phase I
“Connection”
- XML Schema
- SOAP
- WSDL
- UDDI

WS-I Basic Profile

Phase II
“Security and Reliability”
- XML Digital Signature
- XML Encryption
- WS-Security
- SAML
- WS-Policy
- WS-ReliableMessaging

Phase III
“Enterprise”
- Provisioning
- Transactions
- Workflow/BPEL Systems management
- …
What Is a Web Service?

- Specifically – Web Services are a stack of emerging standards that describe a service-oriented, component-based application architecture.
- Conceptually – Web Services represent a model in which discrete tasks within an e-business processes are widely distributed.

"...loosely coupled software components that interact with one another via standard technologies."

- Gartner Group

"...loosely coupled, reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard Internet protocols.

- Stencil Group

".. a broad based agreement for exposing programmatic behavior over a network and a set of core technologies that enable that capacity."

- Noel Bergman (CSS 2002)
What Is a Service Oriented Architecture?

- Service-oriented architectures are **distributed**.
  - Functional elements of the application are deployed on multiple systems and execute across local and even remote networks.
  - In particular, Web services make use of existing, ubiquitous transport protocols like HTTP. By piggybacking on the same, well understood transport as Web content, Web services leverage existing infrastructure and can comply with basic firewall policies.

- The systems are characterized by **loosely coupled** interfaces.
  - Traditional application design depends upon a tight interconnection of all subsidiary elements.
  - The complexity of these connections requires that developers thoroughly understand and have control over both ends of the connection; moreover, once established, it is exceedingly difficult to extract one element and replace it with another.
  - Loosely coupled systems, on the other hand, require a much simpler level of coordination and allow for more flexible reconfiguration.
What Is a Service Oriented Architecture? (Continued)

- Systems are conceived from a **process-centric** perspective.
  - By intent, services are designed with a task-orientation; they function as discrete steps in a larger workflow or business process.
  - A well designed service describes its inputs and outputs in a way that other software can determine what it does, how to invoke its functionality, and what result to expect in return.

- The connections are based upon **vendor-independent standards**.
  - The development of generally open and accepted standards is a key strength of the coalitions that have been building Web services infrastructure.
  - Most previous efforts at distributed computing (*e.g.*, CORBA, DCOM, RMI, and others) were very difficult to implement, because they were dependent upon a particular vendor's product offering, highly specified, or programmatically complex—usually all of the above.
Web Services Architecture

To support the use of Web Services in e-business, IBM, Microsoft, Sun, BEA and others are working to create a concrete Web Services 'stack' that defines how to construct Web Services based solutions.

- An instance of a service oriented architecture
- As defined by the W3C Web Services Architecture Working Group
  - Web Services Architecture Working draft
    http://www.w3.org/TR/ws-arch/
What Is the Web Services Stack?

The **Web Services Stack** is an emerging architectural model and set of standards for developing and deploying software applications.

- Conceptually, **service-oriented architectures** (SOAs) represent a model in which small, loosely coupled pieces of application functionality are published, consumed, and combined with other applications over a network.
- Specifically, **Web services** are a stack of emerging standards that define protocols and create a loosely coupled framework for programmatic communication among disparate systems.
What Is the Web Services Stack?

- A universal client/server architecture that **allows disparate systems to communicate** with each other without using proprietary libraries.

- The architecture simplifies the process typically associated with client/server applications by **effectively eliminating code dependencies between the client and server**.
Web Services are a set of protocols based on XML (Extensible Markup Language).

Base protocols that formed the initial specification for Web Services.

- **Simple Object Access Protocol (SOAP)** – defines the runtime message that contains the service request and response. SOAP is independent of any particular transport and implementation technology.

- **Web Services Description Language (WSDL)** – describes a Web Service and the SOAP Message. It provides a programmatic way to describe what a service does, paving the way for automation.

- **Universal Discovery, Description, Integration (UDDI)** – UDDI is a cross industry initiative to create a standard for service discovery together with a registry facility that facilitates the publishing and discovery processes.
Hot Areas of Web Services Standards...

- **Security**
  Program well underway in OASIS, uncontroversial. WS-Federation is well received.

- **Reliability**

- **Choreography and transactions**
  Specs published August, 2002, convergence of work from IBM and Microsoft, now in OASIS.

- **Management**
  Work is early stage, but good coordination among OASIS, W3C, GGF, and DMTF.

- **Interoperability**
  WS-I.org is the place to be and the work to watch.

- **User Experience**
  WSRP is an OASIS TC specification with many vendors announcing product.
Web Services Standards

Bodies

- **Java Community Process (JCP)**
  Its an open organization of international Java developers and licensees whose charter is to develop and revise Java technology specifications, reference implementations, and technology compatibility kits. Both Java technology and the JCP were originally created by Sun Microsystems, however, the JCP has evolved from the informal process that Sun used beginning in 1995, to a formalized process overseen by representatives from many organizations across the Java community.
  

- **Organization for the Advanced of OASIS Structured Information Standards (OASIS)**
  OASIS is a not-for-profit, global consortium that drives the development, convergence and adoption of e-business standards. Members themselves set the OASIS technical agenda, using a lightweight, open process expressly designed to promote industry consensus and unite disparate efforts. OASIS produces worldwide standards for security, Web services, XML conformance, business transactions, electronic publishing, topic maps and interoperability within and between marketplaces.
  

- **World Wide Web Consortium (W3C)**
  The World Wide Web Consortium (W3C) develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential. W3C is a forum for information, commerce, communication, and collective understanding.
  
  [http://www.w3.org/](http://www.w3.org/)
Web Services Standards

Bodies

- **Internet Engineering Task Force (IETF)**
  The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. It is open to any interested individual.
  

- **United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT)**
  United Nations Body governing technical development in the areas of trade development and e-Business

Web Services Standards Stack

- Web Services for Remote Portlets
  - Business Process Execution Language
  - WS-Policy
  - SOAP, SOAP Attachments
  - XML family of Specifications
  - Transports

- WS-Policy
- WS-Reliable Messaging
- WSDL
- WS-Coordination
- WS-Security family of specifications
- UDDI
- JMS, RMI/IIOP, ...
- XML family of Specifications
- Transports

- User Experience
- Enterprise
- Quality of Service
- Description and Discovery
- Messaging and Encoding
- Foundation

Denise Hatzidakis — SOAP, WSDL, UDDI, JAX/RPC, WSIF, JSR109, WS-Security, BPEL4WS, ... == Web Services
Web Services Stack – Foundation

- Web Services for Remote Portlets
  - Business Process Execution Language
  - WS-Security family of specifications
  - WSDL
  - SOAP, SOAP Attachments
  - XML family of Specifications
  - Transports

- User Experience
  - WS-Transactions
  - WS-Coordination
  - WS-Reliable Messaging
  - UDDI
  - JMS, RMI/IIOP, …

- Enterprise
  - Quality of Service
  - Description and Discovery
  - Messaging and Encoding
  - Foundation
The Web Services Stack – Foundation

**JSR 5 - XML Parsing Specification**
The Java API for XML Parsing (JAXP) allows developers to easily use XML parsers in their applications via the industry standard SAX and DOM APIs.

**Status** | Final release March 2000. See JSR 206 for the next maintenance release.

**JSR 63 - API for XML Processing 1.1**
This specification is a maintenance release of JAXP

**Status** | Final release September 2002. See JSR 206 for the next maintenance release

**JSR 206 - API for XML Processing (JAXP) 1.3**
JAXP 1.3 will be the next version of JAXP, an implementation independent portable API for processing XML with Java.

**Status** | Due for completion mid-2004 (aligned with J2SE 1.5)

**JSR 173 - Streaming API for XML**
The Streaming API for XML (StAX) parsing will specify a Java-based, pull-parsing API for XML. The streaming API gives parsing control to the programmer by exposing a iterator based API that allows the query of the next event (pull event).

**Status** | Community review scheduled for May 2003, final release date unknown.
### The Web Services Stack – Foundation

<table>
<thead>
<tr>
<th>JSR 102 - JDOM 1.0</th>
<th>JCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDOM 1.0 will define an API for easy and efficient reading, manipulation, and writing of XML documents and XML data. JDOM is already a popular open source product in beta release at <a href="http://jdom.org">http://jdom.org</a>.</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>Final draft was originally planned for August 2001, final release date unknown.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JSR 31 - XML Data Binding Specification (JAXB)</th>
<th>JCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>This specification defines an XML data-binding facility for Java. It compiles an XML schema into classes that handle the translation between XML documents that follow the schema. Ensures that constraints defined by schema are maintained.</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>Final release March 2003, date of next maintenance release unknown.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JSR 67 - Java APIs for XML Messaging 1.0 (JAXM)</th>
<th>JCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>This specification describes API’s for the exchange of XML documents such as invoices, purchase orders, and order confirmations. JAXM provides an API for packaging and transporting such transactions using on-the-wire protocols.</td>
<td></td>
</tr>
</tbody>
</table>
The Web Services Stack – Foundation

**XSLT/XPath 1.0 & 2.0**

XSLT transforms XML documents. XPath is a language for addressing parts of an XML document. Extensions to 1.0 provides a much more powerful language that operates on a much larger domain of data types.

**Status**

XSLT/XPath 1.0 is a W3C recommendation. XSLT/XPath 2.0 is in the working draft phase.

**XML Schema 1.0 & 1.1**

XML Schema definition language offers facilities for describing the structure and constraining the contents of XML 1.0 documents (including data types).

**Status**

XML Schema 1.0 is a W3C recommendation. XML Schema 1.1 is in the requirements gathering phase.
# The Web Services Stack – Implementation

<table>
<thead>
<tr>
<th>Specification</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSR 109 - Implementing Enterprise Web Services</td>
<td>This specification defines the programming model and architecture for implementing Web services in Java. It will build on the work of JSRs 67, 93 and 101, and also the JSRs for J2EE technologies, including J2EE itself, Servlets and JSPs.</td>
<td>Final release November 2002. See JSR 921 for the next maintenance release.</td>
</tr>
<tr>
<td>JSR 921 - Implementing Enterprise Web Services 1.1</td>
<td>This is a maintenance revision of the Implementing Enterprise Web Services specification.</td>
<td>Second draft review May 2003, final release date unknown.</td>
</tr>
<tr>
<td>JSR 153 - Enterprise JavaBeans 2.1</td>
<td>The Enterprise JavaBeans 2.1 specification extends the existing Enterprise JavaBeans 2.0 specification with new features, including support for JAXM message-driven beans and Web services usages within EJB.</td>
<td>Final draft was expected August 2002, actual completion date unknown but was due to track J2EE 1.4. Date of next maintenance release unknown.</td>
</tr>
</tbody>
</table>
### The Web Services Stack – Implementation

<table>
<thead>
<tr>
<th>ebXML</th>
<th>Harbinger, IBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>ebXML mission is to provide an open XML-based infrastructure enabling the global use of electronic business information in an interoperable, secure and consistent manner by all parties. EDI centric</td>
<td></td>
</tr>
</tbody>
</table>

| Status | OASIS UN/CEFACT - Some parts open Standards |
The Web Services Stack – Messaging

- Web Services for Remote Portlets
  - User Experience
    - Enterprise
    - Quality of Service
    - Description and Discovery
    - Messaging and Encoding
    - Foundation

- User Experience
  - WS-Transactions
  - WS-Correlation

- WS-Reliable Messaging
  - WS-Security family of specifications
  - WS-Policy
  - UDDI
  - WSDL
  - SOAP, SOAP Attachments
  - JMS, RMI/IIOP, ...
  - XML family of Specifications
  - Transports

Denise Hatzidakis — SOAP, WSDL, UDDI, JAX/RPC, WSIF, JSR109, WS-Security, BPEL4WS, ... == Web Services
## The Web Services Stack – Messaging

### SOAP 1.1 & SOAP 1.2

**Lightweight protocol intended for exchanging structured information using XML.**

**Status**

Both SOAP 1.1 and 1.2 are W3C recommendations.

### SOAP 1.2 w/Attachments & :

**Binary InfoSet & SOAP 1.2 Email Binding**

Provides the basis for the creation of SOAP bindings that transmit such attachments along with a SOAP envelope, and provides for reference of those attachments from the envelope. SOAP 1.2 transport bindings to email.

**Status**

W3C Note December 2000
# The Web Services Stack – Messaging

## Direct Internet Message Encapsulation (DIME)

<table>
<thead>
<tr>
<th>Status</th>
<th>Internet Draft</th>
</tr>
</thead>
</table>

Lightweight binary message format used to encapsulate one or more application defined payloads of arbitrary type and size. (Binary MIME)

## WS-Attachments

<table>
<thead>
<tr>
<th>Status</th>
<th>IETF - Internet Draft</th>
</tr>
</thead>
</table>

Defines an abstract model for SOAP attachments and based on this model defines a mechanism for encapsulating a SOAP message and zero or more attachments in a DIME message.

## JSR 101 - Java APIs for XML based RPC

|---|---|

This specification (known as JAX-RPC) defines Java APIs to support emerging industry XML based RPC standards.
The Web Services Stack – Description and Discovery

- Web Services for Remote Portlets
  - Web Services Execution Language
  - WS-Policy
  - WS-Security family of specifications
  - WSDL
  - SOAP, SOAP Attachments
  - XML family of Specifications
  - Transports

- User Experience
  - WS-Transactions
  - WS-Coordination
  - WS-Reliable Messaging

- Enterprise
  - WS-Coordination

- Quality of Service
  - WS-Timing

- Description and Discovery
  - UDDI

- Messaging and Encoding
  - JMS, RMI/IIOP, ...

- Foundation
  - User Experience

Denise Hatzidakis — SOAP, WSDL, UDDI, JAX/RPC, WSIF, JSR109, WS-Security, BPEL4WS, ... == Web Services
# The Web Services Stack – Description

## WSDL 1.1 & 1.2
XML based language for describing Web Services

**Status**

WSDL 1.2 published as a W3C Note 15 March 2001. WSDL 1.2 is at the working draft phase.

## JSR 110 - Java APIs for WSDL
This JSR provides a standard set of APIs for representing and manipulating services described by WSDL (Web Services Description Language) documents

**Status**

Final release March 2003, date of next maintenance release unknown.

## JSR 181 - Web Services Metadata for the Java Platform
This JSR defines an annotated Java format that that uses JSR 175, Java Language Metadata, to enable easy definition of Java Web services in a J2EE container.

**Status**

TBD based on the schedule for JSR 175 for Java Language Metadata.
## The Web Services Stack – Discovery

<table>
<thead>
<tr>
<th>UDDI v1.0, v2.0, v3.0</th>
<th>Ariba, IBM, Microsoft</th>
</tr>
</thead>
<tbody>
<tr>
<td>A &quot;meta service&quot; for locating Web services by enabling robust queries against rich metadata.</td>
<td></td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>UDDI v1.0, v2.0, v3.0 are OASIS specifications.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WS-Inspection (WSIL)</th>
<th>IBM/MS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provides an XML format for assisting in the inspection of a site for available services and a set of rules for how inspection related information should be made available for consumption.</td>
<td></td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Not Submitted; Specification Published November 2001.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JSR 93 - Java API for XML Registries 1.0 (JAXR)</th>
<th>JCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>API for a set of distributed Registry Services enabling business-to-business integration, using the protocols being defined by ebXML.org, Oasis, ISO 11179. JAXR may be viewed as analogous to Java Naming and Directory Interface (JNDI).</td>
<td></td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Final release June 2002, date of next maintenance release unknown.</td>
</tr>
</tbody>
</table>
The Web Services Stack – Policy
# The Web Services Stack – Policy

## WS-Policy

Provides a general model and syntax to describe and communicate the policies of a Web service. Defines a base set of constructs that can be used and extended to describe a broad range of requirements, preferences, and capabilities.

| Status | Not Submitted yet - Draft Published 18 December 2002 |

## WS-Policy Assertions

Specifies a set of common message policy assertions that can be specified within a policy (e.g. encoding, language).

| Status | Draft 18 December 2002 |

## WS-Policy Attachments

Define how to associate policy expressions with WSDL type definitions and UDDI entities as well as how to associate implementation-specific policy with all or part of a WSDL portType when exposed from a specific implementation.

| Status | Draft 18 December 2002 |
The Web Services Stack – Quality of Service

Web Services Interoperability

- Business Process Execution Language
- WS-Security family of specifications
- WSDL
- SOAP, SOAP Attachments
- XML family of Specifications
- Transports

User Experience

- WS-Transactions
- WS-Coordination
- WS-Reliable Messaging
- UDDI
- JMS, RMI/IIOP, ...

Enterprise

- Description and Discovery
- Messaging and Encoding
- Foundation

Web Services for Remote Portlets

- User Experience
- Enterprise
- Quality of Service
- Description and Discovery
- Messaging and Encoding
- Foundation

Denise Hatzidakis — SOAP, WSDL, UDDI, JAX/RPC, WSIF, JSR109, WS-Security, BPEL4WS, ... == Web Services
The Web Services Stack – Security

WS-Security family of specifications

Quality of Service

WS-Secure Conversation
WS-Federation
WS-Authorization

WS-Security Policy
WS-Trust
WS-Privacy

WS-Security (framework)

Kerberos profile
X509 profile
XrML profile

XCBF profile
Username profile
SAML profile
## The Web Services Stack – Security

<table>
<thead>
<tr>
<th>JSR 104 - XML Trust Service APIs</th>
<th>JCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>This JSR is to define a standard set of APIs and a protocol for a &quot;Trust Service&quot;. A key objective of the protocol design is to minimize the complexity of applications using XML Signature.</td>
<td>Final release date unknown.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JSR 105 - XML Digital Signature APIs</th>
<th>JCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>This JSR is to define a standard set of APIs for XML digital signature services. The XML Digital Signature specification is defined by the W3C. This proposal is to define and incorporate the high level implementation independent Java APIs.</td>
<td>Community review scheduled for April 2003, final release date unknown.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JSR 106 - XML Digital Encryption APIs</th>
<th>JCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>APIs for XML digital encryption services. XML encryption can be used to perform fine-grained, element-based encryption of fragments within an XML Document as well as encrypt arbitrary binary data and include this within an XML document.</td>
<td>Final release date unknown.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JSR 183 - Web Services Message Security APIs</th>
<th>JCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>This JSR is to define a standard set of APIs for Web services message security. The goal of this JSR is to enable applications to construct secure SOAP message exchanges.</td>
<td>Final release date unknown.</td>
</tr>
</tbody>
</table>
### The Web Services Stack – Security

<table>
<thead>
<tr>
<th>Standard</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>XML Signature 1.0</strong></td>
<td>XML digital signature processing rules and syntax. XML Signatures provide integrity, message authentication, and/or signer authentication services for data of any type.</td>
<td>W3C Recommendation, February 2002</td>
</tr>
<tr>
<td><strong>XML Encryption 1.0</strong></td>
<td>Specifies a process for encrypting data and representing the result in XML</td>
<td>W3C Recommendation, 10 December 2002</td>
</tr>
<tr>
<td><strong>XML KMS</strong></td>
<td>Specifies protocols for distributing and registering public keys, suitable for use in conjunction with the proposed standard for XML Signature [XML-SIG] and XML Encryption [XML-Enc].</td>
<td>W3C Working Draft, 18 April 2003</td>
</tr>
<tr>
<td><strong>SAML 1.0 &amp; 1.1</strong></td>
<td>SAML is an XML framework for exchanging authentication and authorization information.</td>
<td>BEA, HP, IBM, RSA, SAP, Sun, Verisign</td>
</tr>
<tr>
<td></td>
<td>SAML 1.1 has been submitted to become an OASIS specification.</td>
<td></td>
</tr>
</tbody>
</table>
# The Web Services Stack – Security

## SOAP-SEC 1.0

<table>
<thead>
<tr>
<th>Status</th>
<th>W3C NOTE 06 February 2001</th>
</tr>
</thead>
</table>

**IBM**

Specifies the syntax and processing rules of a SOAP header entry to carry digital signature information within a SOAP 1.1 Envelope.

## WS-Security (SOAP Security)

<table>
<thead>
<tr>
<th>Status</th>
<th>OASIS working draft.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact</td>
<td>Kelvin Lawrence</td>
</tr>
</tbody>
</table>

**IBM, Microsoft, Verisign**

Describes enhancements to the SOAP messaging to provide quality of protection through message integrity, and single message authentication.

## Tokens: XrML, Username, Kerberos, SAML, X509

<table>
<thead>
<tr>
<th>Status</th>
<th>OASIS working draft.</th>
</tr>
</thead>
</table>

**Various inc., BEA, HP, IBM, RSA, SAP, Sun, Verisign**

These documents describe how to use XrML, Username, Kerberos, SAML and X509 as WS-Security token types.

## WS-SecurityPolicy

<table>
<thead>
<tr>
<th>Status</th>
<th>Draft, 18 December 2002</th>
</tr>
</thead>
</table>

**IBM**

This document is an addendum to [WS-Security] and indicates the policy assertions for [WS-Policy] which apply to [WS-Security].
## The Web Services Stack – Security

### WS-Trust
- **IBM, Microsoft, Verisign**
- This specification defines extensions that build on [WS-Security] to request and issue security tokens and to manage trust relationships.
- **Status** | Not Submitted, Published 18 December 2002

### WS-SecureConversation
- **IBM, MS, Verisign**
- Defines extensions to WS-Security to allow security context establishment and sharing, session key derivation. The context authentication model authenticates a series of messages.
- **Status** | Not Submitted, Published 18 December 2002

### WS-Authorization
- **IBM**
- Management of authorization data and authorization policies.
- **Status** | Private DRAFT specification.
## The Web Services Stack – Security

### WS-Privacy

How to specify privacy preferences for subjects as well as specify organizational privacy practices

**Status** | Private DRAFT specification.

### WS-Federation, WS-Federation Active, WS-Federation Passive

This specification defines mechanisms to allow different security realms to federate by allowing and brokering trust of identities, attributes, authentication between participating Web services.

**Status** | Not Submitted, Published June 2003
The Web Services Stack – Reliability

- Web Services for Remote Portlets
  - WS-Coordination
  - WS-Transactions
  - WS-Reliable Messaging

- WS-Policy family of specifications
- WS-Security family of specifications

- WSDL
- UDDI
- SOAP, SOAP Attachments
  - JMS, RMI/IIOP, …

- XML family of Specifications
- Transports

- User Experience
- Enterprise
- Reliability
- Security
- Description and Discovery
- Messaging
- Foundation
The Web Services Stack – Reliability

WS-ReliableMessaging


Planned

WS-Reliable Messaging

Quality of Service

Denise Hatzidakis — SOAP, WSDL, UDDI, JAX/RPC, WSIF, JSR109, WS-Security, BPEL4WS, ... == Web Services
# The Web Services Stack – Reliability

<table>
<thead>
<tr>
<th><strong>WS-Reliability</strong></th>
<th>Oracle/Sun</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification for open, reliable Web services messaging including guaranteed delivery, duplicate message elimination and message ordering, enabling reliable communication between Web services.</td>
<td>OASIS – Technical Committee Formed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>WS-Reliable Messaging</strong></th>
<th>Oracle/Sun</th>
</tr>
</thead>
<tbody>
<tr>
<td>The purpose of this TC is to create a generic and open model for ensuring reliable message delivery for Web services. Reliable message delivery is the ability to guarantee message delivery to software applications - Web services or Web service client applications - with a chosen level of quality of service (QoS). Formerly WS-Reliability</td>
<td>OASIS – Technical Committee Formed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>WS-Reliablemessaging</strong></th>
<th>BEA/IBM/MS/Tibco</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describes a protocol that allows messages to be delivered reliably between distributed applications in the presence of software component, system, or network failures.</td>
<td>Not Submitted</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>WS-Addressing</strong></th>
<th>BEA, IBM, Microsoft, TIBCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defines XML elements to identify Web service endpoints and to secure end-to-end endpoint identification in messages. Enables messaging systems to support transmission through networks that include processing nodes (e.g. filters).</td>
<td>13 March 2003</td>
</tr>
</tbody>
</table>
The Web Services Stack - Enterprise

Web Services Interoperability:
- Web Services for Remote Portlets
  - Business Process Execution Language
  - WS-Security family of specifications
  - WSDL
  - SOAP, SOAP Attachments
  - XML family of Specifications
  - Transports

Enterprise:
- WS-Transactions
- WS-Coordination

Quality of Service:
- WS-Reliable Messaging

Description and Discovery:
- UDDI
- WS-Policy

Messaging and Encoding:
- JMS, RMI/IIOP, ...

Foundation:
- User Experience
# The Web Services Stack – Enterprise

<table>
<thead>
<tr>
<th><strong>BPEL4WS 1.1</strong></th>
<th>IBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification language for automatic business processes; executable processes; public view onto internal processes</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>05 May 2003</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>BPEL4WS 1.2</strong></th>
<th>IBM, BEA, Microsoft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification language for automatic business processes; executable processes; public view onto internal processes</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>OASIS TC discussions. Purpose is to standardize BPEL4WS 1.1.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>WS-Choreography</strong></th>
<th>EDS, HP, Oracle, Sun, Tibco</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working Group created to address the ability to compose and describe the relationships between Web services.</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>W3C Working Group Formed</td>
</tr>
</tbody>
</table>

### The Web Services Stack – Enterprise

#### WS-Composite Application Framework (WS-CAF)

<table>
<thead>
<tr>
<th>Status</th>
<th>Not Submitted</th>
</tr>
</thead>
</table>

Proposes standard, interoperable mechanisms for managing shared context and ensuring business processes achieve predictable results and recovery from failure. WS-CTX, WS-CF and WS-TXM.

**BEA/IBM/MS**

#### WS-Coordination

<table>
<thead>
<tr>
<th>Status</th>
<th>Not Submitted, Published August 2002</th>
</tr>
</thead>
</table>

An extensible framework for providing protocols that coordinate the actions of distributed applications. Used to support a number of applications, including those that need to reach agreement on the outcome of distributed transactions.

**BEA/IBM/MS**

#### WS-Transaction

<table>
<thead>
<tr>
<th>Status</th>
<th>Not submitted, Published 9 August 2002</th>
</tr>
</thead>
</table>

Defines two coordination types: Atomic Transaction (AT) and Business Activity (BA). You can use either or both of these coordination types when building applications that require consistent agreement on the outcome of distributed activities.

**IBM**
The Web Services Stack – Enterprise

Web Services Distributed Management (WSDM)

Define Web services management, including using Web services architecture and technology to manage distributed resources. This TC will also develop the model of a Web service as a manageable resource.

Status: OASIS Technical Committee formed

WS-Manageability

Web services manageability is defined as a set of capabilities for discovering the existence, availability, health, performance, and usage, as well as the control and configuration of a Web service within the Web services architecture.

Status: OASIS; Specification Published
# The Web Services Stack – Enterprise

<table>
<thead>
<tr>
<th>JSR 156 - XML Transactioning API for Java (JAXTX)</th>
<th>JCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAXTX provides an API for packaging and transporting ACID transactions (as in JTA) and extended transactions (e.g. the BTP from OASIS) using the protocols being defined by OASIS, W3C.</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>Final release date unknown.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JSR 157 - ebXML CPP/A APIs for Java</th>
<th>JCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>This JSR is to provide a standard set of APIs for representing and manipulating Collaboration Profile and Agreement information described by ebXML CPP/A (Collaboration Protocol Profile/Agreement) documents.</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>Final release date unknown.</td>
</tr>
</tbody>
</table>
## The Web Services Stack – Enterprise

### WS-Composition
Compose or aggregate service into higher order services. Define “relationships” between the members that form the composition

<table>
<thead>
<tr>
<th>Status</th>
<th>TBD</th>
</tr>
</thead>
</table>

### WS-Relationships
Relationships between Web services. To be abstracted from CORBA relationship services and UML associations

- Relationships of different semantics (Associations, aggregations, compositions, ...). Navigation capabilities

<table>
<thead>
<tr>
<th>Status</th>
<th>TBD</th>
</tr>
</thead>
</table>

### WS-Metadata Exchange
Exchange of WSDL, policies and XSD. Base for bootstrapping mechanics. “Get an EPR and discover all about the service”

<table>
<thead>
<tr>
<th>Status</th>
<th>TBD</th>
</tr>
</thead>
</table>
The Web Services Stack – User Experience

- Web Services for Remote Portlets
  - Business Process Execution Language
  - WS-Security family of specifications
  - WSDL
  - SOAP, SOAP Attachments
  - XML family of Specifications
  - Transports
  - WS-Policy
  - JMS, RMI/IIOP, ...
  - UDDI
  - WS-Reliable Messaging
  - WS-Transactions
  - WS-Coordination

- User Experience
- Enterprise
- Quality of Service
- Description and Discovery
- Messaging and Encoding
- Foundation
# The Web Services Stack – User Experience

**JSR 168 - Portlet Specification**

To enable interoperability between Portlets and Portals, this specification will define a set of APIs for Portal computing addressing the areas of aggregation, personalization, presentation and security.

| Status | Community review scheduled for June 2003, final release date unknown. |

**Web Services For Remote Portlets (WSRP) 1.0**

Web services standard that will allow for the "plug-n-play" of portals, other intermediary Web applications that aggregate content, and applications from disparate sources.

| Status | OASIS - Approved |

**JSR 213 - Micro WSCI Framework for J2ME.**

Micro WSCI is a framework that will reside on the J2ME device, and will act as a lightweight coordinator for choreographed Web services.

| Status | Final estimated release date November 2004. |

**JSR 172 - J2ME Web Services Specification**

The purpose of this specification is to define an optional package that provides standard access from J2ME to Web services.

| Status | Public review scheduled for March 2003, final release date unknown (initially scheduled for 3Q 2003). |
The Web Services Stack – Web Services Interoperability

- Web Services Interoperability
  - Business Process Execution Language
  - WS-Security family of specifications
  - WSDL
  - SOAP, SOAP Attachments
  - XML family of Specifications
  - Transports

- User Experience
  - WS-Transactions
  - WS-Coordination

- Enterprise
  - WS-Reliable Messaging

- Quality of Service
  - WS-Policy

- Description and Discovery
  - UDDI

- Messaging and Encoding
  - JMS, RMI/IIOP, ...

- Foundation
  - Web Services for Remote Portlets
WS-I Standards

**WS-I Basic Profile 1.0**
A specification defining interoperability conformance of a Web service instance providing constraints and clarification using a set of named and versioned Web services specifications.

**Status**

**WS-I Basic Profile 1.1**
A follow-on revision of Basic Profile 1.0 incorporating support for attachments.

**Status**
Expected December 2003.

**WS-I Sample Applications 1.0**
A simple Supply Chain Management sample application which demonstrates an application implementation built from Web services usage scenarios and use cases, and that conform to the Basic Profile 1.0.

**Status**
Expected October 2003.

**WS-I Test Tool 1.0**
Testing tools used to monitor and analyze interactions with a Web service to determine whether or not the messages exchanged conform to WS-I Profile 1.0 guidelines.
Standards Adoption

2003 2004 2005

User Experience
Enterprise
Quality of Service
Description and Discovery
Messaging
Foundation

WSRP
BPEL4WS
WS-Transaction
WS-Coordination
Choreography
WS-CAF
WS-ReliableMessaging
WS-Reliability
WS-Federation
WS-Addressing
WS-SecureConversation
WS-Trust
WS-Security
WS-Policy
UDDI
WSDL
WS-Attachments (DIME)
SOAP
XML Specifications
Transports

Specification
Experimentation
Early Adoption
Mainstream
Uncertain
Web Service Interoperability

Issues

- **Technology Users**
  - Will their deployed Web services interoperate with those of their partners?
  - Will their choice of technologies be supported in their partners and customers tools and platforms?
  - Will their choice of tools and platform be interoperable with those of their partners?

- **Technology Vendors**
  - Which specifications will gain broad support?
  - Will their implementations be interoperable with those of their competitors?
  - Will they be interpreting the specifications in the same way as their competitors?

- **Technology Standards**
  - W3C, OASIS, IETF, etc...
WS-I Organization

- Founded Feb 2000
- Industry initiative for Web services
  - Open to any organization committed to Web services
  - Promote and accelerate adoption, deployment
- Focused on promoting Web service interoperability
  - Across platforms, applications, and programming languages
  - Promote a common, clear definition for Web services
WS-I Organization

Mission Statement

"The Web Services Interoperability Organization is an open industry effort chartered to promote Web Services interoperability across platforms, applications, and programming languages. The organization brings together a diverse community of Web services leaders to respond to customer needs by providing guidance, recommended practices, and supporting resources for developing interoperable Web services."
Effect on the Marketplace

- Tool vendors will advertise that their products support development and deployment of WS-I conformant services
- Middleware vendors will advertise that their products support conformant Web service hosting
- Customers will look for WS-I conformance on:
  - Products
  - Deployed instances
  - Vertical standard interface descriptions
The WS-I Community

- 160+ members

- Software vendors of all sizes
  - IBM, Microsoft, BEA, Oracle, HP, Sun, ...

- Enterprise customers
  - AT&T, Daimler-Chrysler, NTT, Fidelity, United, ...

- Others interested in Web services
  - EDS, Accenture, ...

All members are invited to actively participate
WS-I Members
WS-I Working Groups

- WS-I has defined the charters for three initial working groups:
  - The Basic Web Services Profile Working Group
    - Will identify a core set of specifications (including XML Schema, SOAP, WSDL and UDDI) that provide the foundation for Web services, and will establish conventions and recommendations for coordinating their use.
  - The Sample Applications Working Group
    - Will provide sample applications of basic Web services to accelerate deployments. These sample applications illustrate best practices for implementation and will be developed in multiple programming languages using multiple development tools. Sample applications serve as working examples for companies planning to implement Web services.
  - The Test Materials and Tools Development Working Group
    - Will develop a suite of self-administered tests to verify conformance with the Basic Web Services Profile. These tools and materials can be used to ensure that Web services interoperate across platforms, applications and programming languages.
WS-I Deliverables

- **Use Cases and Usage Scenarios**
  - Use Case - business usage of Web services, Usage Scenario - technical usage of Web services
  - Formalized way to communicate community requirements
  - Specific emphasis on “real-world” use cases and scenarios

- **Profiles**
  - Named sets of specifications at given version levels
  - Constraints, clarifications and conventions about how they are used together

- **Sample Applications**
  - Demonstrated use of Profiles as defined in Use Cases and Scenarios

- **Test suites and supporting materials**
  - Conformance testing tools
  - Test assertions for the profile
WS-I Process

Scenarios and Sample Applications

Use Cases → Usage Scenarios → Sample Applications

Web Services Basic Profile

Profiles

Sample Applications

Testing Tools and Materials

Testing Tools

Other Test Materials

Profiles

Sample Applications

Usage Scenarios

Use Cases

Scenarios and Sample Applications

Profiles

Testing Tools and Materials

Testing Tools

Other Test Materials

Profiles
How Does All This Work?

- Look to SOAP builders and implementation experience
- Follow W3C XMLP, WSD WG decisions
- Specification trumps schema
- The “right” technical alternative
- Compromise and Negotiation
WS-I Value Add

- Choose practical paths through existing and emerging WS standards
  - Turn “MAY”s and “SHOULD”s into “MUST”s or “MUST NOT”s
  - Choose between overlapping standards
  - Provide guidance or best practices on standards implementation
  - Point out shortcomings in existing standards
  - Provide concrete examples, objective conformance criteria and conformance test tools

- Benefit to the Web services community
  - Simplification improves chances for interoperability
  - Reduced churn and noise allows people to focus
  - Single forum for discussion reduces duplicate and fragmented efforts
What’s a Profile?

- Named set of Web services specifications
  - e.g. SOAP, WSDL, UDDI
- Base specifications are normative unless...
- Profile adds constraints and guidance as to their interoperable usage based upon implementation experience
- Organized around base specifications
Philosophy of the Profile

- No guarantee of interoperability
- Application semantics
- Testability
- Strength of requirements
- Restriction vs. relaxation
- Choose between multiple mechanisms
- Future compatibility
- Compatibility with deployed services
- Focus on interoperability
- Conformance targets
- Lower-layer interoperability
Profile Composition

- **Foundation Profile** *(e.g. WS Basic Profile 1.0)*
  - Can be used standalone to deploy a Web service

- **Extension Profile**
  - *(formerly known as ... Feature Profiles)*
  - Requires foundation profile
  - May be used with one or more foundation
    - *(e.g. Basic Security (WS-Security))*

- **Objective** == minimum # of Profiles

- **Fold in feature profiles over time as extensions become widely adopted/used**
WS-Basic Profile 1.0

- SOAP 1.1
- WSDL 1.1
- UDDI 2.0
- XML Schema
- XML 1.0 (Second Edition)
- HTTP 1.1
- SSLv3 or TLS 1.0
- Other supporting/referenced specs/standards
Basic Profile 1.0 Issue Distribution

Issue Distribution

- **SOAP**: 24%
- **WSDL**: 44%
- **HTTP**: 12%
- **Schema**: 6%
- **UDDI**: 5%
- **Others**: 9%

Denise Hatzidakis — SOAP, WSDL, UDDI, JAX/RPC, WSIF, JSR109, WS-Security, BPEL/WS, ... == Web Services
Basic Profile 1.0 Technical Highlights

- **SOAP1.1**
  - Use of SOAP encoding disallowed
  - “Trailers” (element content after soap-env:Body) disallowed
  - Most spec ambiguity issues resolved in alignment with SOAP1.2
  - Use of SOAPAction, soap-env:actor clarified

- **WSDL1.1**
  - Limited to use of rpc/literal and document/literal
  - SOAP/HTTP binding required
    - Other bindings out of scope but may be present in WSDL document
    - Schema errors fixed
    - Spec treated as normative
  - Exclude use of wsdl:import for XSD files
  - Numerous spec clarifications
  - Markup for conformance claims provided: `<wsi:Claim conformsTo="..."/>`
Technical Highlights

- **UDDI2.0**
  - Require WSDL1.1 as description language
  - Established category to identify WS-I conformant entities

- **Security**
  - May use SSLv3 or TLS 1.0 over HTTP
  - HTTP1.1 Basic Auth
  - Identify risks and threats and countermeasures within Basic Profile

- **XML Schema**
  - Any valid XSD constructs may be used (all, choice, sequence, *etc*)
  - Recommend use of xsi:nil xs:nillable to designate NULL values

- **HTTP1.1**
  - Clarify use of HTTP response status codes
    - soap:Fault == 500, redirect == 307
  - Cookies permitted, but must not be required
Typical Form

- Reference section of underlying specification
  - WSDL1.1 Section 3.0
- Statement prose
  - Say it in English
- Requirement(s)
  - <RequirementProse xml:lang="en-specese"/>
- Rationale
  - Where appropriate, explain why decision was taken
- Example
  - Where appropriate include incorrect and correct examples
Example Requirement

The following specifications (or sections thereof) are referred to in this section of the Profile;

WSDL 1.1, Section 2.1

WSDL 1.1 defines an XML-based structure for describing Web services. The Profile mandates the use of that structure, and places the following constraints on its use:

5.1.1 WSDL Schema Definitions

The normative schemas for WSDL appearing in Appendix 4 of the WSDL 1.1 specification have inconsistencies with the normative text of the specification. The Profile references new schema documents that have incorporated fixes for known errors.

R2028 A DESCRIPTION using the WSDL namespace (prefixed "wsdl" in this Profile) MUST be valid according to the XML Schema found at "http://schemas.xmlsoap.org/wsdl/2003-02-11.xsd".

R2029 A DESCRIPTION using the WSDL SOAP binding namespace (prefixed "soapbind" in this Profile) MUST be valid according to the XML Schema found at "http://schemas.xmlsoap.org/wsdl/soap/2003-02-11.xsd".

Although the Profile requires WSDL descriptions to be Schema valid, it does not require consumers to validate WSDL documents. It is the responsibility of a WSDL document's author to assure that it is Schema valid.
Example Requirement

- A SOAP Fault is a SOAP message that has a single child element of the `soap:Body` element, that element being a `soap:Fault` element. The Profile restricts the content of the `soap:Fault` element to those elements explicitly described in SOAP 1.1.

R1000 When a MESSAGE contains a `soap:Fault` element, that element MUST NOT have element children other than `faultcode`, `faultstring`, `faultactor` and `detail`. 
Example Requirement

For example,

**INCORRECT:**

```xml
<soap:Fault xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/' >
  <faultcode>soap:Client</faultcode>
  <faultstring>Invalid message format</faultstring>
  <faultactor>http://example.org/someactor</faultactor>
  <detail>There were <b>lots</b> of elements in the message that I did not understand</detail>
    <m:Exception xmlns:m='http://example.org/faults/exceptions' >
      <m:ExceptionType>Severe</m:ExceptionType>
    </m:Exception>
  </detail>
</soap:Fault>
```
Example Requirement

CORRECT:

```xml
<soap:Fault xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/' >
  <faultcode>soap:Client</faultcode>
  <faultstring>Invalid message format</faultstring>
  <faultactor>http://example.org/someactor</faultactor>
  <detail>
    <m:msg xmlns:m='http://example.org/faults/exceptions'>
      There were <b>lots</b> of elements in the message that I did not understand
    </m:msg>
    <m:Exception xmlns:m='http://example.org/faults/exceptions'>
      <m:ExceptionType>Severe</m:ExceptionType>
    </m:Exception>
  </detail>
</soap:Fault>
```
Conformance of Services, Consumers and Registries

- **INSTANCE** - software that implements a `wsdl:port` or a `uddi:bindingTemplate`.
- **CONSUMER** - software that invokes an INSTANCE
- **REGISTRY** - a UDDI registry, capable of managing REGDATA.
- **SENDER** - software that generates a message according to the protocol(s) associated with it
- **RECEIVER** - software that consumes a message according to the protocol(s) associated with it (e.g., SOAP processors)

Note that conformance does not apply to a service as a whole; **only ports are considered when determining conformance of instances**. Therefore, the Profile places no constraints on `wsdl:service` definitions. In particular, they can contain multiple `wsdl:port` elements, each of which may or may not be conformant.
Basic Profile 1.1

- Basic Profile 1.0 foundation or starting point
  - WG hasn’t yet determined whether attachments is feature or if basic profile is just versioned to incorporate SwA and WSDL MIME ext
- Adds support for SOAP Messages with Attachments
  - W3C Note September 2000
  - Uses MIME multipart/related packaging
- WSDL MIME extension
- Closed most all of the issues
- Nothing earth shattering or controversial

Working group status – No public draft available
WS-I BSPWG
(Basic Security Profile Working Group)

- BSPWG Charter Defined (April 1, 2003)
  - The newly chartered BSPWG will develop an interoperability profile involving transport security, SOAP messaging security and other security considerations implicated by the WS-I Basic Profile.
  - The Basic Security Profile is intended to be an extension to the WS-I Basic Profile 1.0.
  - It will reference existing specifications used to provide security and provide clarifications and guidance designed to promote interoperability of those specifications.
WS-I Sample Applications

- Supply chain application
  - Retailers, Suppliers, Manufacturers, Consumers, Warehouses, ...

---

Diagram showing the processes and actors involved in a supply chain application.
WS-I Sample App

WS-I Sample Application Showcase

Enter a User ID and Password to continue.

Sign In ID: [Field]
Sign In Password: [Field]

Sign In

Customer Data
- Configure
- Place Order
- Order Status
- Order Tracking

Configurator

Retailer

Warehouse
- WarehouseCallback

Manufacturer

LoggingFacility

Request/Response
- One Way
- Basic Callback
WS-I Testing Tools

- Pre-release Testing Tools *(April 16, 2003)*
  - Availability of two testing tools for interoperability assessment with the WS-I Basic Profile 1.0.
  - Pre-release version of the Web Service Communication Monitor.
    - Captures messages exchanged between Web services and the software that invokes them and stores the messages for later analysis. The pre-release version captures HTTP-based SOAP messages.
  - Pre-release version of the Web Service Profile Analyzer.
    - Evaluates messages captured by Monitor, and also validates the description and registration artifacts of the Web service. This includes the WSDL document(s) that describes the Web service, and the XML schema files that describe the data types used in the WSDL service definition and the UDDI registration entries.
  - Implementations in both C# and Java.

Currently version 0.93 ... new release in a couple weeks.
Testing Tools

- **Objectives**
  - Conformance testing for Web services
  - Not focused on testing conformance of platforms or tools

- **Two testing tools**
  - Message Monitor and Analyzer

- **Deliverables**
  - Testing tools architecture document
  - Machine-readable version of WS-I profile definition
  - Implementation of testing tools with User’s Guide

- **Status**
  - V0.93 available soon on [www.ws-i.org](http://www.ws-i.org) Web site
  - V1.0 scheduled to be available 9/2003
When Will WS-I Deliver?

**Basic Profile 1.0**
- Approved unanimously July 22, 2003
  - 11-member WS-I board of directors
  - Approximately 150 member organizations-at-large

**Basic Profile 1.1**
- Working Group Draft
- Not publicly available at this point
Web Services Stack – Key Components

- **Web Services** are a set of protocols based on XML (Extensible Markup Language).
- **Base protocols** that formed the initial specification for Web Services.
  - **Simple Object Access Protocol (SOAP)** – defines the runtime message that contains the service request and response. SOAP is independent of any particular transport and implementation technology.
  - **Web Services Description Language (WSDL)** – describes a Web Service and the SOAP Message. It provides a programmatic way to describe what a service does, paving the way for automation.
  - **Universal Discovery, Description, Integration (UDDI)** – UDDI is a cross industry initiative to create a standard for service discovery together with a registry facility that facilitates the publishing and discovery processes.
SOAP Messaging Model

- SOAP messages are fundamentally one way transmissions from a sender to a receiver
- Remote Procedure Call (RPC) can be simulated using two one-way messages
  - one for the RPC request
  - one for the RPC response
  - other communication patterns can be simulated as well but SOAP defines a convention for RPC
- The SOAP specification doesn't enforce anything about the "real" contents of the message as long as it is valid XML.
  - Use of 'SOAP encoding' is anticipated but not required
  - There are strict rules for how the messages elements map to a method call
  - Does not require HTTP
- The SOAP model for message processing allows one or more intermediate nodes (intermediaries) to process a message before the message reaches its final destination
What are the components of a SOAP message?
Components of a SOAP Message

The SOAP message for a stock quote request as it appears on the WIRE

```xml
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <SOAP-ENV:Header>
    <dummy:header xmlns:dummy="http://nowhere.com"/>
 </SOAP-ENV:Header>
 <SOAP-ENV:Body>
    <ns1:getQuote xmlns:ns1="urn:StockQuoteService"
                  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <symbol xsi:type="xsd:string">IBM</symbol>
    </ns1:getQuote>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
SOAP Envelope

- The SOAP Envelope is the root element of the XML document representing a SOAP message. Everything is inside the Envelope.
- The Envelope can contain an optional Header element.
- The Envelope contains a required Body element.
- Example Envelope:

```xml
<SOAP-ENV:Envelope ...namespace stuff...>
  <SOAP-ENV:Header>
    ... Optional
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    ... Required
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
SOAP Header

- The header is optional
- Elements in the head can use two attributes from the SOAP Envelope namespace
  - SOAP-ENV:mustUnderstand
  - SOAP-ENV:actor
- Sample Usage
  - Authentication
  - Digital signature support
  - Encryption support
  - Transaction management
- Example Header:

```xml
<SOAP-ENV:Header>
  <dummy:header xmlns:dummy="http://nowhere.com"/>
</SOAP-ENV:Header>
```
SOAP Body

- body element is always required
  - may contain zero or more elements
  - body element contains
    - application specific XML vocabulary
    - must be well-formed XML
    - (should be valid against WSDL and Schema - outside of the SOAP spec)

- The actual 'service' code is minimally aware of the need to be XML or SOAP aware
  - depends upon SOAP engine implementation
  - may be an extension to existing application code

- When processed as an RPC request, the body element must have a single element in it which maps to a method name
  - The elements/parts within the method element must map exactly to the name, type, and order of the method parameters

- SOAP engine is responsible for executing the service code based on the SOAP Body and translating the return from the service into the SOAP Response message
  - conventions for the Response message
  - SOAP-ENV:encodingStyle is an optional attribute
SOAP Body – Details

- SOAP/WSDL define two styles for what the body means:
  - RPC Style
    - The body represents a procedure call / method invocation
  - Document Style
    - The body represents data OR
    - The body represents a procedure call / method invocation

- How can the body represent a procedure call / method invocation?
  - Explicitly – put the name of the method / procedure call in the body as a wrapper element (RPC Style)
    
    ```xml
    <methodname>
      <arg1>value</arg1>
      <arg2>value</arg2>
      ...
      <argn>value</argn>
    </methodname>
    
    Implicitly – the receiver of the message knows to interpret the data as an invocation so just put the arguments in the body (Document Style)
    
    ```xml
    <arg1>value</arg1>
    <arg2>value</arg2>
    ...
    <argn>value</argn>`
SOAP Body –

WS-I Recommendation

- The children of the soap:Body element in a MESSAGE MUST be namespace qualified.
- A PROCESSOR MUST NOT mandate the use of the xsi:type attribute in messages except as required in order to indicate a derived type
  - Apache SOAP does require xsi:type, Axis does not
SOAP Body – RPC

- Below is the body of the StockQuote RPC request
  - The name of the SOAP structure (first element in the body) is the same as the method being invoked
  - There is an XML element (an accessor) named and typed the same as the corresponding parameter in the method call. They should appear in the same order as in the method signature

```xml
<SOAP-ENV:Body>
  <ns1:getQuote xmlns:ns1="urn:StockQuoteService"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <symbol xsi:type="xsd:string">IBM</symbol>
  </ns1:getQuote>
</SOAP-ENV:Body>
```
The body of the StockQuote RPC response

- The response struct can be named anything, but by convention it is named methodNameResponse
- An accessor named return that represents the return value of the method call

```xml
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV=
        "http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SOAP-ENV:Body>
        <ns1:getQuoteResponse xmlns:ns1="urn:StockQuoteService"
            SOAP-ENV:encodingStyle=
                "http://schemas.xmlsoap.org/soap/encoding/">
            <return xsi:type="xsd:float">100.36</return>
        </ns1:getQuoteResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
Soap – Fault element

- SOAP Fault element is used to carry error and/or status information within a SOAP message
- If present, the SOAP Fault element MUST appear as a body entry
  - instead of a xxxMethodName or xxxResult
- Fault element defines the following four subelements:
  - faultcode
    - SOAP-ENV:Version Mismatch
      - Processing party found an invalid namespace for the SOAP Envelope
    - SOAP-ENV:MustUnderstand
      - a header which contained a SOAP-ENV:mustUnderstand attribute could not be processed
    - SOAP-ENV:Client
      - The client sent an incorrectly formed or incomplete message
    - SOAP-ENV:Server
      - The server was unable to process the message. There was nothing wrong with the message.
  - faultstring
  - faultactor
  - detail
SOAP Fault

A Fault is used to return an error or status information

```xml
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Server</faultcode>
      <faultstring>Method 'getQuoted' is not supported.</faultstring>
      <faultactor>/StockQuote/servlet/rpcrouter</faultactor>
      <detail>
        ...
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
SOAP and Namespaces

- Namespaces are used heavily throughout SOAP
- There are two SOAP namespaces
  - One for the elements of a SOAP Envelope – required
  - One for elements related to the SOAP encoding – optional
- All content in a SOAP Header must be namespace qualified.
- Content in the SOAP Body (element) doesn't have to be namespace qualified, but it is a best practice to use a namespace
  - application specific namespace for the application specific elements
    - examples: stockQuoteService
  - multiple namespaces may be used in the body of the message
  - often used by the SOAP engine to identifying the Web service being called
  - WS-I restricts - requires a namespace on the Body element
    - Apache SOAP uses this namespace for dispatching

Best Practice: use application specific namespaces for the body.
Soap and Namespaces

```xml
<SOAP-ENV:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <SOAP-ENV:Body>
     <ns1:getQuoteResponse
        xmlns:ns1="urn:StockQuoteService"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
       <return xsi:type="xsd:float">100.36</return>
     </ns1:getQuoteResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
SOAP and XML Schema

- SOAP requires a rich set of XML tags to represent data passed in with SOAP Call requests and returned as responses
  - DTDs are not adequate to define complex datatypes and to model things like inheritance.
- XML Schema provide a general way to define data, validate data and place constraints on that data
  - Not developed specifically for SOAP but provides the functionality that is needed for SOAP
- SOAP can use XML Schema to encode data
  - Parameters to method calls
  - Return values
- XML Schema is not required by the SOAP specification but most SOAP implementations (e.g. Apache SOAP) use them
  - Best Practice: Use XML Schema
**Encoding and Mapping**

- SOAP needs a way to convert programming language types into XML for the SOAP Body when the body represents an RPC.
- This is called type encoding:
  - SOAP provides a pre-built encoding scheme:
    - Based on subset of XML Schema datatypes
    - Defined in the namespace "http://www.w3.org/2001/XMLSchema"
  - You can also choose no encoding – this is called “literal XML”:
    - Elements conform to an XML Schema

SOAP allows you to use any encoding scheme you want and provides means to plugin custom encoding code.

SOAP -encoding simple examples:

```xml
<symbol xsi:type="xsd:string">IBM</symbol>
<return xsi:type="xsd:float">550.99</return>
```

**WS-I Recommendation**

"For interoperability, literal XML is preferred."
What Does a SOAP Engine Do...

- Provides Java Classes to create and parse SOAP messages
- Server side infrastructure for deploying, managing and running SOAP enabled services
- Client-side API for invoking SOAP services

![Diagram of SOAP communication process]
What Does a SOAP Engine Do?

- Receive a message from a Transport
- Check SOAP semantics
- Process the SOAP headers
- Deserialize the message
- Route the message to the service
- Serialize the response (if request/response)
- Process the response SOAP headers
- Send the response back out over the Transport
SOAP Engine Processing

Transport
- HTTP
- JMS
- SMTP

Handler Chain (Intermediaries)

Type Mapping
- Deserialization
- Serialization

Service Implementation (JavaBean/EJB)

Configuration
Deployment Descriptors
Client API

- Provide a proxy for the Web service
- Provide a way to build up a set of parameters for the service request
- Provide a way to invoke the Web service
- Provide a way to get the result of invoking the service
- Provide a way to notify the application of faults that occur
SOAP over HTTP

- HTTP is one transport for SOAP messages
  - also HTTPS which just adds SSL security encryption
- HTTP Request Header must include SOAPAction header field (URI)
  - Indicates intent of HTTP Request
  - Format not specified, can be empty ""
- Some Web Servers and SOAP engines use SOAPAction for routing
- The SOAP Envelope is carried in content of HTTP POST
- Response may contain a SOAP Envelope
- When an Error occurs:
  - Must send HTTP 500 with SOAP envelope containing fault (not 200)
- The use of HTTP as the protocol makes the interaction synchronous

WS-I Recommendation
- header MAY contain any quoted string including "".
- The SOAPAction header is purely a hint to processors.
- All vital information regarding the intent of a message is carried in the Envelope.
- specifies HTTP response codes for variety of situations
SOAP over HTTP – Request

POST /StockQuote/servlet.rpcrouter HTTP/1.0
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: 454
SOAPAction: "urn:StockQuoteService"

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:getQuote xmlns:ns1="urn:StockQuoteService"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xsi:type="xsd:string">IBM</symbol>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

The SOAPAction header is required by some SOAP engines to route the request to the matching Web service – non-conformance with WS-I
HTTP/1.1 200 OK
Server: WebSphere Application Server/4.0
Content-Type: text/xml; charset=utf-8
Set-Cookie: JSESSIONID=0000NMQZFAGA30SVMHZXQ125FRQ:-1;Path=/
Cache-Control: no-cache="set-cookie,set-cookie2"
Expires: Thu, 01 Dec 1994 16:00:00 GMT
Content-Length: 474
Content-Language: en
Connection: close

<?xml version='1.0' >
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
   <ns1:getQuoteResponse xmlns:ns1="urn:StockQuoteService"
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <return xsi:type="xsd:float">100.36</return>
</ns1:getQuoteResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
SOAP Attachments

- Sometimes it is desirable to transmit a SOAP message along with some attachments
  - Legal briefs
  - Engineering diagrams
  - Specifications
  - Contract documents

- SOAP Attachments are a proposal for sending attachments along with a SOAP message
  - in the W3C approval process as a Note

- SOAP Attachments rely on packaging a SOAP message as part of a MIME multipart structure
  - The SOAP message can refer to the attachments using URIs constructed during creation of the multipart

- SOAP Attachments are independent of the transport
SOAP Intermediaries

- We need a way to modularize services to promote reuse. One way is via intermediaries.
- Intermediaries can both receive and forward SOAP messages.
  - can also add headers
- Examples:
  - Authentication
  - Digital signature
  - Encryption
  - Internal message routing
  - Other processing based on envelope characteristics
- SOAP-ENV:actor attribute indicates a header meant for an intermediary
SOAP vs. XML over HTTP

- You may be wondering: Why don't I just use HTTP GET and POST to transmit XML data over HTTP? Why do I need SOAP?

- Interoperability
  - Your XML over HTTP is almost guaranteed not to be compatible with someone else's.
    - you would have to write custom XML RPC processor for each partner's XML definition of RPC
    - SOAP is a standard supported by many tools

- Economy of Scale
  - You will have to implement your own XML over HTTP libraries
  - SOAP is simple, standard, extensible, and dozens of implementations are available, many for free

- Development productivity
  - toolkits, wizards, and SOAP engines integrated into IDEs
  - standardization of skills

- Headers
  - ability to separate information about the message from the message body in a standard fashion
  - Leverage SOAP header processing services
SOAP Based Web Services – Future

- Extensions (new standards)
  - WS-Security, WS-Transactions, WS-Coordination
  - workflow - BPEL4WS standard

- Industry-specific SOAP messages will start a new round of vocabulary standards work
  - need a standard way to present the content/data of the messages
    - ex. purchase order
  - momentum behind Web Services should accelerate

- Interoperability
  - Ability for different SOAP engines to exchange messages seamlessly
  - Currently is major focus area by the key vendors
  - WS-I - interoperability consortium
SOAP – Specification Status

- SOAP 1.0: Userland, Microsoft, DevelopMentor
- SOAP 1.1 (April 26, 2000) – includes contributions from IBM and Lotus
- SOAP 1.1 was submitted to W3C, as a Note
  - W3C "Web Services Protocol" working group: Sept 13, 2000
    - work is done in public
    - [http://www.w3.org/2002/ws/](http://www.w3.org/2002/ws/)
    - drafts available: requirements, definitions
    - chairperson: David Fallside, IBM
- SOAP 1.2 - W3C candidate recommendation
  - [www.w3.org/2000/xp/](http://www.w3.org/2000/xp/)
- SOAP 1.2 E-mail Binding - W3C Note
- SOAP Attachments – W3C Note
  - [http://www.w3.org/TR/SOAP-attachments](http://www.w3.org/TR/SOAP-attachments)
WSDL Service Description

Web Services Description Language

- From the specification: "XML format for describing network services as a set of endpoints operating on messages ... The operations and messages are described abstractly, and then bound to a concrete network protocol and message format"

- Service description is the key to interoperability of services

Serves as a recipe for automating the details involved in applications communication
WSDL Definition

WSDL describes:

- Implementation independent description of the service
  Service interface definition
- Where the service is located
  Service implementation definition

WSDL provides:

- A simple standardized way for service providers to describe the basic format of requests to their system.
- A "contract" between the client and server

WSDL specification

http://www.w3.org/TR/2001/NOTE-wsdl-20010315
How Is WSDL Used?

- Standardized Service Interface Descriptions
  - Allows advertisement and dynamic discovery of services
  - Communicates to the service requesters all of the information they need to be able to invoke the service
  - Enables dynamic binding to service by service requesters
  - Can be stored in a UDDI registry

- Used by Tools
  - To generate client code to bind to a service
  - To generate SOAP deployment descriptors
Our Example

- **EmployeeManager class**
  - 2 methods we will expose
    - public Vector getEmployeeList()
    - public Employee getEmployee(String lastName)

- **Employee Class**
  - return type used in the service

Employee
- employeeNo, lastName, firstName, phoneNumber
- department, salary, hireDate
Basic WSDL Syntax

- Describes
  - What a service can do
  - Where the service can be found
  - How to invoke the service

- Describes them as
  - Collection of endpoints that are able to exchange messages
WSDL Document Overview

- **Definition**
  - The root of the WSDL document
  - Contains the definition of one or more services
  - Usually contains attributes

- **Service**
  - Defines the service

- **Messages and PortTypes**
  - Describes the actions available for the service

- **Bindings**
  - How to communicate with the service
WSDL Elements

A WSDL document defines Web Service via:

**Messages**
- Defines a single interaction with the service

**Types**
- Defines data types used in a message

**Operations**
- Description of an action

**Port Types**
- Describes the set of operations supported by the service.

**Bindings**
- A concrete protocol and data format for a particular port type.

**Port**
- Describes the network address where the service is being hosted.

**Service**
- Ties together all the elements of the service.
## WSDL Elements

### Service Interface

Abstract, reusable service definition

Represents a type of service that can be implemented

### Service Implementation

Implementation of one or more service interfaces

Contains the endpoint reference
<definitions> element

- The root element of the WSDL document
- Usually defines a single service

```xml
<wSDL:definitions
    targetNamespace="http://employee.webservices.wsad.ibm.com"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:apachesoap="http://xml.apache.org/xml-soap"
    xmlns:impl="http://employee.webservices.wsad.ibm.com"
    xmlns:intf="http://employee.webservices.wsad.ibm.com"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

Denise Hatzidakis — SOAP, WSDL, UDDI, JAX/RPC, WSIF, JSR109, WS-Security, BPEL4WS, … == Web Services
<definitions> element attributes

- targetNamespace
  - Chosen to be unique for this service
  - targetNamespace="http://employee.webservices.wsad.ibm.com"

- xmlns
  - Default namespace – the WSDL defined name space
  - xmlns="http://schemas.xmlsoap.org/wsdl/"

- xmlns:apachesoap
  - Apache SOAP Type namespace
  - xmlns:apachesoap="http://xml.apache.org/xml-soap"

- xmlns:impl
  - Implementation namespace
  - xmlns:impl="http://employee.webservices.wsad.ibm.com"

- xmlns:intf
  - Interface namespace
  - xmlns:intf="http://employee.webservices.wsad.ibm.com"

- xmlns:wsdl
  - WSDL namespace
  - xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"

- xmlns:wsdlsoap
  - SOAP namespace
  - xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"

- xmlns:xsd
  - Schema namespace
  - xmlns:xsd="http://www.w3.org/2001/XMLSchema"
<types> element attributes

- Contains data type definitions other than those defined by the base schema
- May be imported into the WSDL document

```xml
<wsdl:types>
  <schema elementFormDefault="qualified"
    targetNamespace="http://employee.webservices.wsad.ibm.com"
    xmlns="http://www.w3.org/2001/XMLSchema">
    <complexType name="Employee">
      <sequence>
        <element name="department" nillable="true" type="string"/>
        <element name="phoneNumber" nillable="true" type="string"/>
        <element name="lastName" nillable="true" type="string"/>
        <element name="hireDate" nillable="true" type="dateTime"/>
        <element name="firstName" nillable="true" type="string"/>
        <element name="salary" type="double"/>
        <element name="employeeNo" nillable="true" type="string"/>
      </sequence>
    </complexType>
    <element name="getEmployeeResponse">
      <complexType>
        <sequence>
          <element name="getEmployeeReturn" nillable="true" type="intf:Employee"/>
        </sequence>
      </complexType>
    </element>
  </schema>
</wsdl:types>
```
**<message> and <part> Elements**

- **<message>**
  - A single piece of information moving between the requester and provider
  - A single interaction between requester and provider
  - By convention
    - methodName `Request` and methodName `Response`

- **<part>**
  - Describes a piece of data associated with the message
  - Optional

```xml
<message name="getEmployeeRequest">
  <part name="lastName" element="impl:getEmployee"></part>
</message>

<message name="getEmployeeResponse">
  <part name="result" element="impl:getEmployeeResponse"></part>
</message>
```

Request and Response messages for public Employee `getEmployee(String lastName)`
<operation> and <portType>

- Defines what operations the Web Service provides

- <operation>
  - An action
  - Like a Java Method
  - Three messages
    - input message
    - output message
    - fault message

- <portType>
  - A collection of operations
  - Like a Java Class

```xml
<message name="getEmployeeRequest">
  <part name="lastName" element="impl:getEmployee"></part>
</message>
<message name="getEmployeeResponse">
  <part name="result" element="impl:getEmployeeResponse"></part>
</message>
<portType name="EmployeeManager">
  <operation name="getEmployee">
    <input message="impl:getEmployeeRequest" name="getEmployeeRequest"></input>
    <output message="impl:getEmployeeResponse" name="getEmployeeResponse"></output>
  </operation>
</portType>
```
<binding> Elements

- **<binding>**
  - **HOW** the operation is invoked
  - Ties to the <portType> element to the protocol defined for the binding
  - Defines the transport **protocol** and **style** (rpc or document)
  - If a service supports more than one protocol then there should be multiple bindings for the port type.
  - For each operation of the <portType> each <message> is detailed
  - For SOAP, the <soap:body> element defines **use** (encoding or literal) and **namespace**
  - If using encoding, <soap:body> specifies **encodingStyle**

```xml
<portType name="EmployeeManager">
  <operation name="getEmployee">
    <input message="impl:getEmployeeRequest" name="getEmployeeRequest"> </input>
    <output message="impl:getEmployeeResponse" name="getEmployeeResponse"></output>
  </operation>
</portType>

<binding name="EmployeeManagerSOAPBinding" type="impl:EmployeeManager">
  <wsdlsoap:binding
    transport=http://schemas.xmlsoap.org/soap/http
    style="document"/>
  <wsdl:operation name="getEmployee"><wsdlsoap:operation />
    <input name="getEmployeeRequest"> <wsdlsoap:body use="literal"/>
    </input>
    <output name="getEmployeeResponse"> <wsdlsoap:body use="literal"/>
    </output>
  </wsdl:operation>
</binding>
```
<service> Element Attributes

- **<service>**
  WHERE the service is located
  A collection of <ports>s
- **<port>**
  defines the availability of a particular binding at a specific endpoint
  binding attribute must correspond to a <binding> element
  <soap:address> defines the actual location of the service

```xml
<service name="EmployeeManagerService">
  <port name="EmployeeManagerServicePort" binding="impl:EmployeeManagerSOAPBinding"
    location="http://localhost:9080/WSWSDLWeb/services/EmployeeManager" />
</service>
```

```xml
<binding name="EmployeeManagerSOAPBinding" type="impl:EmployeeManager">
  <service name="EmployeeManagerService">
    <port name="EmployeeManagerServicePort" binding="impl:EmployeeManagerSOAPBinding">
      <wsdlsoap:address location="http://localhost:9080/WSWSDLWeb/services/EmployeeManager" />
    </port>
  </service>
</binding>
```
The WSAD WSDL Editor
What Is UDDI?

Universal Description, Discovery and Integration

- **Purpose**
  - To facilitate service discovery both at design time and at runtime
  - A platform independent framework for describing services, discovering businesses, and integrating business services via the Internet
  - A directory for storing information about Web Services
  - UDDI provides support for many types of service descriptions
  - Intended to enable the potential of establishing a potential relationship between a service provider and a service requester

- **Uses**
  - SOAP, XML and lower level Internet protocols
Uddi.org

- Partnership among industry & business leaders
  - Founders (IBM, Microsoft, Ariba) developed basic version 1 specification
- Eventually grew to over 320 UDDI community members in uddi.org
  - Members jointly developed Versions 2 & 3
  - Included companies such as Accenture, Compaq, Fujitsu, HP, Intel, NTT, Oracle, SAP, Sun & many others
The Evolution of UDDI

- UDDI Version 2 – (June 2001)
  - Business Relationships
    - Modeling of Complex Organizations - business units, departments, divisions, and subsidiaries
    - Uses such as membership, certification, etc.
  - Additional categorization and identifier schemes
  - Support for externally 'checked' taxonomies
  - Richer search options: more expressive query parameters, more powerful & complex filters
    - Better results with less work
  - Improved Internationalization
    - Names and descriptions in multiple languages
  - Peer based replication for improved scaling
The Evolution of UDDI

- UDDI Version 3 – (June 2002)
  - Multi-registry topologies
  - New subscription API
  - New security features
  - Improved WSDL support
  - Core information model improvements
The Evolution of UDDI

Transition to a Standard...

- UDDI was transferred to OASIS 8/2002
- New Technical Committee (TC) formed for UDDI
  - Approved UDDI V2 as an OASIS UDDI TC Specification
  - Approved UDDI V2 to enter the OASIS Standard Process
  - Approved UDDI V3 as an OASIS UDDI TC Specification

Upcoming Activity for the OASIS UDDI TC

- Completion of V2 OASIS Standard Process
- Advance V3 as an OASIS Standard
- Continued development of Best Practices & Technical Notes
- Maintain the UDDI Specifications
- Development of UDDI V4
  - Spec. compatibility
  - Taxonomy handling
  - more...

NOTE: UDDI Operators Council is still an independent group
# UDDI Registry

- There are three roles defined for UDDI data
  - Described in terms of telephone directories
  - Conceptual only

<table>
<thead>
<tr>
<th>White Pages</th>
<th>Contains contact information for service providers so that a requester can contact the provider directly (negotiation, technical support, etc...)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yellow Pages</td>
<td>Contains service information; service categories</td>
</tr>
<tr>
<td>Green Pages</td>
<td>Technical information about the individual service</td>
</tr>
</tbody>
</table>
UDDI Provides

- UDDI provides three functions common to all "name services"
  - Publish
    - A provider of a Web Service Registers itself
  - Find
    - An application finds a provider of a Service
  - Bind
    - An application connects to the Service
UDDI Message Dynamic

Client

UDDI
SOAP Request

UDDI
SOAP Response

UDDI Registry Node

HTTP
Server

SOAP
Server

Process UDDI
API Request

Registry Data
UDDI Structure

- UDDI Structure is defined in XML
- Defines 4 basic elements
  - Business Entities
  - Business Services
  - Binding Templates
  - tModels

Diagram:
- Business Entity
  - Business Service
    - Binding Template
    - Binding Template
  - Business Service
    - Binding Template
    - Binding Template
  - Interface (binding.wsdl)
  - Implementation (service.wsdl)
  - tModel
  - tModel
  - tModel
businessEntity

- The white and yellow pages of the registry
- Contains business information about the service provider
  - business name, business contacts, descriptions, identifiers, categories, etc...

```xml
<?xml version="1.0" encoding="utf-8" ?>
<businessDetail generic="1.0" xmlns="urn:uddi-org:api" operator="www.ibm.com/services/uddi"
  truncated="false">
  <businessEntity authorizedName="100000A8B6"
    operator="www.ibm.com/services/uddi"
    businessKey="2CEF9630-9118-11D6-B746-000C0E00ACDD">
    <discoveryURLs>...</discoveryURLs>
    <name>WSADWS</name>
    <description xml:lang="en">WSAD Web Services Bootcamp</description>
    <businessServices>...</businessServices>
    <identifierBag>...</identifierBag>
    <categoryBag>...</categoryBag>
  </businessEntity>
</businessDetail>
```
businessService

- Green pages (part 1)
- Provides non technical service information
- Used to group a set of related Web Services

- Catagorization is available at this level
- Maps to a WSDL service

```xml
<businessServices>
  <businessService serviceKey="51A0BEA0-9118-11D6-B746-000C0E00ACDD"
                     businessKey="2CEF9630-9118-11D6-B746-000C0E00ACDD">
    <name>TemperatureConverterService</name>
    <description xml:lang="en">Temperature Converter Service</description>
    <bindingTemplates>...
    <categoryBag>
      <keyedReference tModelKey="UUID:C0B9FE13-179F-413D-8A5B-5004DB8E5BB2"
                      keyName="Professional, Scientific, and Technical Services"
                      keyValue="54">
      </keyedReference>
    </categoryBag>
  </businessService>
</businessServices>
```
bindingTemplate

- Green pages (part 2)
- Contains service access information
- Points to a service implementation description
  - Wraps a WSDL port

```xml
<businessService serviceKey="51A0BEA0-9118-11D6-B746-000C0E00ACDD"
    businessKey="2CEF9630-9118-11D6-B746-000C0E00ACDD">
    <name>TemperatureConverterService</name>
    <description xml:lang="en">Temperature Converter Service</description>
    <bindingTemplates>
        <bindingTemplate bindingKey="51A838B0-9118-11D6-B746-000C0E00ACDD"
            serviceKey="51A0BEA0-9118-11D6-B746-000C0E00ACDD">
            <description xml:lang="en" />
            <accessPoint URLType="http">http://localhost:8080/WSSWeb/servlet/rpcrouter</accessPoint>
            <tModelInstanceDetails>....</tModelInstanceDetails>
        </bindingTemplate>
    </bindingTemplates>
    <categoryBag>...</categoryBag>
</businessService>
```
**tModel**

- **Technology Model**
- The technical fingerprint (also known as the service type)
- Points to a service interface description
- **Example**
  - Use a tModel to define a new WSDL port type
  - Specify a business service implements that port type by associating the tModel with one of the services binding templates

```xml
<bindingTemplate bindingKey="51A838B0-9118-11D6-B746-000C0E00ACDD"
    serviceKey="51A0BEA0-9118-11D6-B746-000C0E00ACDD">

  ...

  <tModelInstanceDetails>
    <tModelInstanceInfo tModelKey="UUID:CF2D5680-9104-11D6-B746-000C0E00ACDD">
      <instanceDetails>
        <overviewDoc>
          <overviewURL>
            http://localhost:8080/WSSWeb/wsdl/TempConverter-service.wsdl#TemperatureConverterPort
          </overviewURL>
        </overviewDoc>
      </instanceDetails>
    </tModelInstanceInfo>
  </tModelInstanceDetails>

</bindingTemplate>
```
**WSDL and UDDI**

**WSDL**

- **Service Implementation**
  - import

- **Service**
  - port

- **Service Interface**
  - types
  - message
  - portType
  - binding

**UDDI**

- **Business Entity**
  - **Business Service**
    - Binding Template
    - Binding Template
    - tModel
UDDI4J

<table>
<thead>
<tr>
<th>Business Entity</th>
<th>Service</th>
<th>Binding</th>
<th>tModel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save/Update</td>
<td>save_business</td>
<td>save_service</td>
<td>save_binding</td>
</tr>
<tr>
<td>Delete</td>
<td>delete_business</td>
<td>delete_service</td>
<td>delete_binding</td>
</tr>
<tr>
<td>Find</td>
<td>find_business</td>
<td>find_service</td>
<td>find_binding</td>
</tr>
<tr>
<td>GetDetail</td>
<td>get_businessDetail</td>
<td>get_serviceDetail</td>
<td>get_bindingDetail</td>
</tr>
</tbody>
</table>
Finding a Service with UDDI4J

- Access the registry
- Collect the tModels
  - Based on the service name
  - "http://www.temperatureconverter.com/definitions/TemperatureConverterRemoteInterface"
- Find the Business Entity
  - Based on the business Name
    - WSADWS
- Find the Business Services of the Business Entity
- Find the Bindings of the Service that implements the tModel
  - Match the Service keys with the Binding Template keys
- Collect the Access Points from the Bindings
- Set the Access Point in the Proxy generated for the Service
- Invoke the Service based on the Access Point
Finding a Service with UDDI4J

Provider

WSADWS

Business Entity (BusinessInfo)

BusinessService (ServiceInfo)

Service

"http://www.temperatureconverter.com/definitions/TemperatureConverterRemoteInterface"

Service Interface (TModelInfo)

TModelBag

UUIDs (TModelKey)

Binding Detail (BindingTemplate)

Entry Point (AccessPoint)

http://localhost:8080/WSWeb/servlet/rpcrouter
UDDI V3 .. What’s Coming...

- UDDI V3 creates a more robust registry for the growth of Web Services, addressing:
  - Affiliated Registry Environments
  - Policy & Security
  - Modeling Enhancements
  - Tooling Needs
  - Extensions

- Now as OASIS Specifications, they will grow to address infrastructure needs of Web Services and improve compatibility with other standards
Monitor

- **Interceptor**
  - Message Capture
  - Transport-specific
  - Captures transport level messages and errors

- **Logger**
  - Logging and formatting
  - Records run-time output of Interceptor
  - Formats log entries (XML) and exports as a file
Using the Monitor

- Alter the Requestor
- Move the Service
- Alter the WSDL port or UDDI accessPoint

![Diagram showing the interaction between requesters and services]
Analyzer

Test Assertion Document

Analyzer Config File

WSDL Document

Analyzer

UDDI Entry

Message Log

Conformance Report

Analyzer

XSLT

Summary
- Result: passed

Artifacts
- discovery
- description
- message

Artifact: discovery

Assertion Result Summary:

<table>
<thead>
<tr>
<th>Assertion ID</th>
<th>Passed</th>
<th>Failed</th>
<th>Warning</th>
<th>Not Applicable</th>
<th>Not Testable</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSDL002</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>WSDL003</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Denise Hatzidakis — SOAP, WSDL, UDDI, JAX/RPC, WSIF, JSR109, WS-Security, BPEL4WS, ... == Web Services
Analyzer

- Analysis of each artifact can be done separately
  - Discovery: Analysis of UDDI Entries
  - Description: Test WSDL content conformance
  - Message: Execute HTTP and SOAP related test assertions

- Conformance Report
  - Test assertion results, failure details, summary

- De-coupled from Monitoring
  - Conformance tests can be at run-time or later
5.6.3 Consistency of style Attribute

The style, "document" or "rpc", of an interaction is specified at the wsdl:operation level, permitting wsdl:bindings whose wsdl:operations have different styles. This has led to interoperability problems.

R2705 A wsdl:binding in a DESCRIPTION MUST use either be a rpc-literal binding or a document-literal binding.
Basic Profile Test Assertions – Example

```xml
<testAssertion id="WSI2017" entryType="binding" type="required" enabled="true">
  <context>For a candidate wsdl:binding</context>
  <assertionDescription>The "style" attribute of each operation in the contained soap:binding has the same value of "document" or "rpc", for all operations of the wsdl:binding.</assertionDescription>
  <failureMessage>
    The "style" attribute of an operation in soap:binding, does not have the same value of "document" or "rpc", as other operations of the binding.
  </failureMessage>
  <failureDetailDescription>defective soap:binding element.</failureDetailDescription>
  <additionalEntryTypeList>
    <messageInput>none</messageInput>
    <wsdlInput>none</wsdlInput>
    <uddiInput>none</uddiInput>
  </additionalEntryTypeList>
  <prereqList/>
  <referenceList>
    <reference profileID="BP1">R2705</reference>
  </referenceList>
  <comments/>
</testAssertion>
```
Analyzer Configuration

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wsi-analyzerConfig:configuration name="Sample Basic Profile Analyzer Configuration"
    xmlns:wsi-analyzerConfig="http://www.ws-i.org/testing/2003/03/analyzerConfig/"
    xmlns:wsi-common="http://www.ws-i.org/testing/2003/03/common/">
    <wsi-common:description xml:lang="en">
        This file contains a sample of the configuration file for
        the Basic Profile Analyzer, which can be used with the
        other sample files.
    </wsi-common:description>
    <wsi-analyzerConfig:verbose>false</wsi-analyzerConfig:verbose>
    <wsi-analyzerConfig:assertionResults type="all" messageEntry="true" failureMessage="true"/>
    <wsi-analyzerConfig:reportFile replace="true" location="flyByNight/report.xml">
        <wsi-common:addStyleSheet href="../../common/xsl/report.xsl" type="text/xsl"/>
    </wsi-analyzerConfig:reportFile>
    <wsi-analyzerConfig:testAssertionsFile>
        ../common/profiles/BasicProfileTestAssertions.xml
    </wsi-analyzerConfig:testAssertionsFile>
    <wsi-analyzerConfig:wsdlReference>
        <wsi-analyzerConfig:wsdlElement type="port"
            parentElementName="Service1"
            namespace="http://FlyByNight.com/service/">
            Service1Soap
        </wsi-analyzerConfig:wsdlElement>
        <wsi-analyzerConfig:wsdlURI>
            flyByNight/FlyByNightService.wsdl
        </wsi-analyzerConfig:wsdlURI>
    </wsi-analyzerConfig:wsdlReference>
</wsi-analyzerConfig:configuration>
```
Analyzer Report

<artifact type="description">
  <entry type="definitions" referenceID="file:/E:/WSI/wsi-test-tools/java/flyByNight/FlyByNightService.wsdl">
    <assertionResult id="WSI2201" result="passed">
    </assertionResult>
    <assertionResult id="WSI2700" result="passed">
    </assertionResult>
    <assertionResult id="WSI2701" result="passed">
    </assertionResult>
    <assertionResult id="WSI2703" result="passed">
    </assertionResult>
    <assertionResult id="WSI2018" result="passed">
    </assertionResult>
    <assertionResult id="WSI2099" result="passed">
    </assertionResult>
    <assertionResult id="WSI2100" result="passed">
    </assertionResult>
    <assertionResult id="WSI2101" result="passed">
    </assertionResult>
    <assertionResult id="WSI2103" result="passed">
    </assertionResult>
    <assertionResult id="WSI2104" result="passed">
    </assertionResult>
    <assertionResult id="WSI2105" result="passed">
    </assertionResult>
    <assertionResult id="WSI2416" result="passed">
    </assertionResult>
    <assertionResult id="WSI2013" result="passed">
    </assertionResult>
  </entry>
</artifact>
Getting Started with the WS-I Testing Tools

- Download V0.93 Tools Package
- Unzip the tools package and review the documentation
  - README.txt \wsi-test-tools
  - UserGuide.pdf \wsi-test-tools\common\docs
  - ReleaseNotes.txt \wsi-test-toos\java
  - MonitorSpecification.pdf \wsi-test-tools\common\docs
  - AnalyzerSpecification.pdf \wsi-test-tools\common\docs
- Run the tools from the java directory using the sample files
  - bin\monitor –config samples\monitorConfig.xml
  - bin\analyzer –config samples\analyzerConfig.xml

See Kelvin Lawrence’s “Web Services Advanced Topics: Beyond SOAP, WSDL and UDDI”