
IBM Software Services for WebSphere
Introduction to Web Services
Web Services Advanced Training
Sept 2004
© 2004 IBM Corporation

Agenda:

Business and Architecture
- Business Context
- Web Service definition

Web Services Defined
- What is Web Service?
- Relationship to SOA
- Introduction to Standards

Web Services Standards
- Soap Protocol
- WDSL Defined
- UDDI
- Standards and Organizations
- WS-I

Additional Information

Key Lecture points

•There are SOAs built and will continue to be built without use of Web Services for example with Corba or messaging.

•Web Services is a great technology to use to build SOAs because of its standards base, platform and language independence, and wide vendor and industry adoption.

•The business drivers leading to implementing SOA are very similar to those leading to use of Web Services. However Web Services maybe adopted to solve tactical or architecture challenges that are not as broad as adoption of SOA – for example integrating two systems directly together vs. a full SOA integration.

- •The biggest drive for Web Services is interoperability

•IBM is THE leader for open standards and particularly for XML and Web Services

•Introduce the 3 core technologies and how the relate – emphase the relative importance and adoption

- •XML is absolutely core and provides the platform independence
- •The 3 core technologies have matured
- •IBM's view of Web Services is that WSDL is the KEY Technology
- •SOAP is very important as well and has wide adoption
- •UDDI adoption is primarily within companies and in extranet; public UDDI registries are not being widely used for business

•Interoperability basically works

- •For more complex cases can get challenging
- •Level of product support do remain an issue but much more so for newer standards than for SOAP and WSDL
- •WS-I is key

Business Context

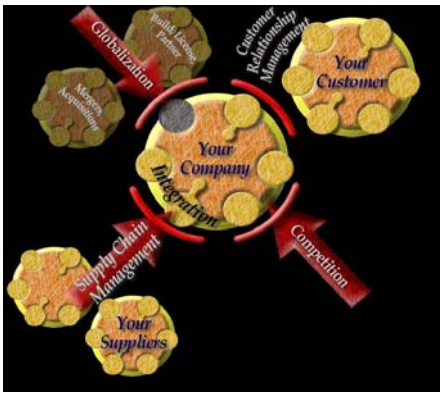
There is a tendency in the IT industry to use the newest cool technology. Technology for technology's sake.

This leads to poorly architected solutions, incorrect uses of technology and reliance on lightly tested code paths.

This section is intended to provide an understanding of why the industry is moving to web services, and

where they fit into a J2EE architecture.

What's Driving e-Businesses?



New Market threats & opportunities

An Explosion of Access Points

- **a trillion connected devices**
- **a billion connected people**
- **a million connected e-businesses**

An Explosion of Transactions

"Transaction Volumes will Grow by A Factor of 50 in the next 5 years" - Forrester 1/99

Rethinking IT

A world wide economic down turn is causing many companies to rethink their IT budgets.

Number of connections around the Internet is skyrocketing. Think about PDAs, cell phones, intelligent refrigerators all connecting to the Internet. People want more services from the Internet.

They want to not just browse - not just human to web interaction. App to app interaction. Example, want a custom app in your PDA to get information from multiple applications, data set.

A New Internetworking Model: Integration

- Until now, the Web has provided for
  - browsing of linked documents
  - manually-initiated purchases and transactions
  - downloading files
    - –all of this is manual, by way of a browser
- Web Services is a new model for using the Web
  - transactions initiated automatically by a program, not necessarily using a browser
  - can be described, published, discovered, and invoked dynamically in a distributed computing environment
  - new ways of using the web: intelligent agents, marketplaces, auctions
    - –**all built on XML!**
- The content-oriented web will be complemented by a service-oriented web
  - the process has started and is gaining momentum

In business-to-business, same situation. Need to do more than just cool websites to make money.

Business models are going towards hard core B2B - especially within marketplaces and industries. Get a specific set of companies talking together - not all of the Internet. Extranet.

This is not a new problem. We have had distributed technologies for connecting applications - RPC, DCOM, CORBA, RMI, messaging, etc. But they do not work over the open Internet, across different platforms and systems. Cannot just decide on EJBs for your partner interactions.

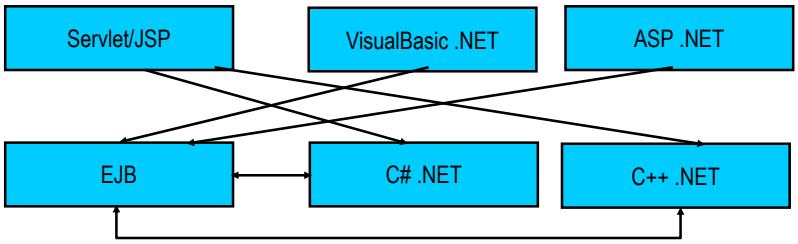
Need vendor and platform independence.

That's where WS play - non-vendor specific over the open Internet.

Packet structure of IIOP does things that do not work over open Internet such as embedding the IP addresses. Also even if you fix CORBA stuff, MS won't play.

Industry agreement across key vendors is key.

Interoperability



- Enabling interoperability of disparate technologies
  Microsoft .NET and Java/J2EE
- Connect disparate systems
  - business partners across the Internet
  - legacy systems across the intranet
- All based on widely accepted, vendor-neutral standards

Web Services Defined

What is a Web Service?

- Difficult to give a strict definition (but I will)
- Web Services is abstract like:
  - e-business
  - on demand
- Web Services makes uses of several standards such as:
  - XML
  - WSDL
  - SOAP
  - UDDI
- IBM's view is that a Web Services is described by the Web Service Description Language (WSDL)

You can't buy "on demand" or "e-business", because its not a thing. It's a class of things, or a strategy.

So too with Web Services. There are many implementations of Web Services, Apache SOAP, Axis, JAX-RPC,

dot Net.

Web Services are deliberately defined to be flexible. Which to some means complicated!

Even these standards are subject to replacement. The Web Services architecture could be

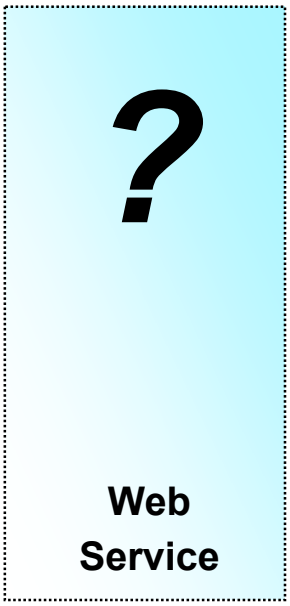
rewritten using some other encoding other than SOAP.

The intention is to evolve an interoperability standard. The target is moving.

What is a Web Service?

- **"A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards ."**

- from the Web Services Architecture, February 11, 2004



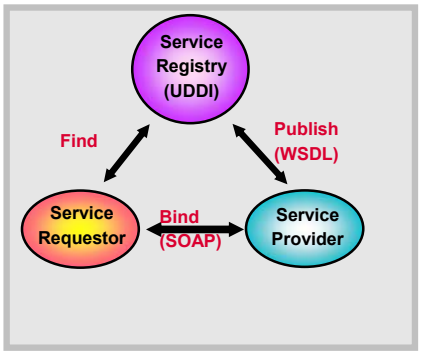
Source: http://www.w3.org/TR/ws-arch/

Web Services Relationship to SOA

- There are Service Oriented Architectures which are implemented without using Web services
  - For example: based on CORBA or WBI Message Broker
- Web services are a leading technology choice to use for implementing SOAs today
  - Standards based
  - Cross platform and cross language
  - Widely supported
  - Message oriented
  - Tooling support speeds implementation of SOA
- There are many web services implementations which are not SOA
  - Example: connecting two heterogeneous systems directly together
  - These uses of web services solve real problems and provide significant value
  - Maybe the starting point or first iteration to moving to SOA

Web Service Components

- Service Provider
  - Provides e-business services
  - publishes availability of these services through a registry
- Service Registry
  - Provides support for publishing and locating services
  - like telephone yellow pages
- Service Requestor
  - Locates required services via the Service Registry
  - binds to services via Service Provider



Web Services: Base Technologies

- SOAP -

  - An XML protocol to invoke a "function" on a server to perform a given operation
  - Request message is sent by service requestor and optionally a response message may be sent by service provider
  - May be an asynchronous message <---> notification

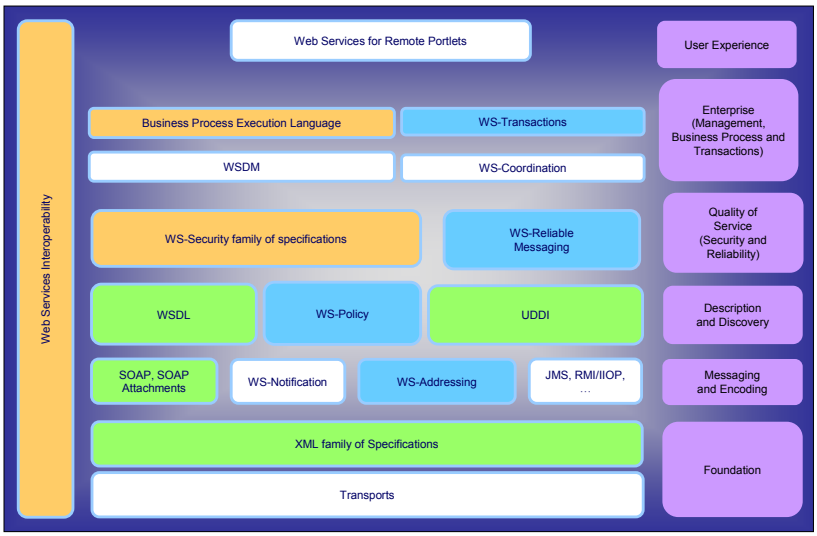
- WSDL - Web Services Description Language

  - An XML vocabulary to describe service interfaces

- UDDI - Universal Description, Discovery, Integration

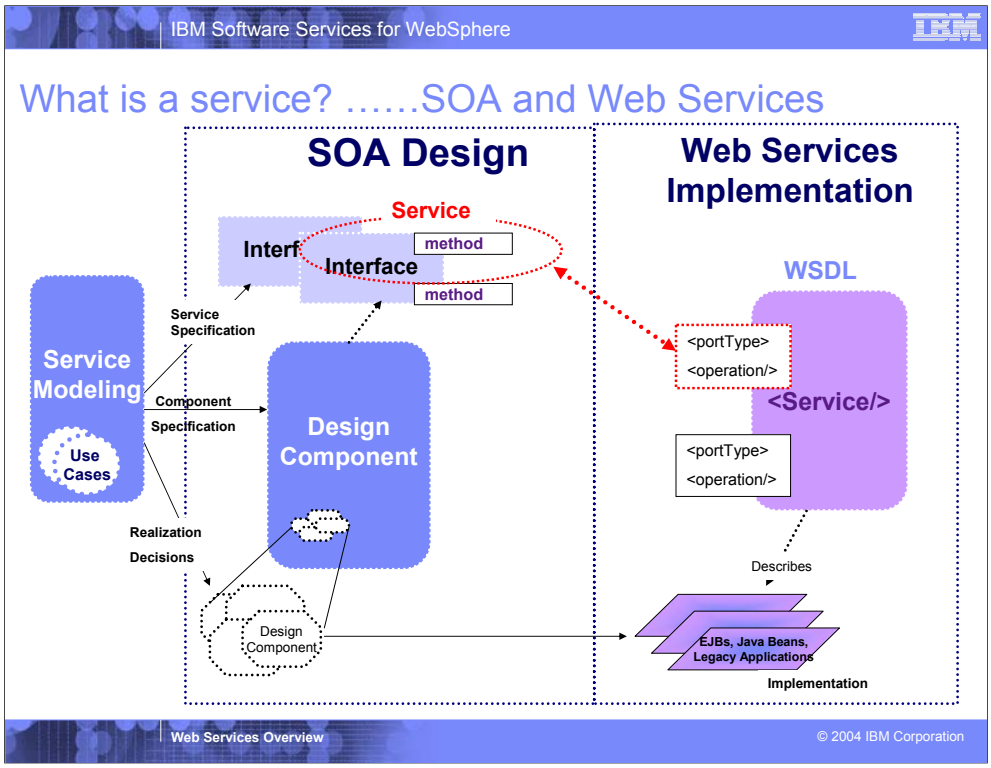
  - UDDI servers act as a directory of available services and service providers
  - A SOAP application used to update and query a registry for services

Web Services Protocol Stack



Advanced Web Services Standards and Specifications

- WS-Security
  - Message-level standard defined how to secure SOAP messages
- Business Process Execution Language (BPEL)
  - Describe business processes in XML which use web services
- WS-Addressing
  - How to represent an address of a web service and define a target of a message
- WS-Policy
  - Assertions of the capabilities and requirements of a WS requestor or provider
  - Assertions and may (or may not) have any wire format translation
- WS-ReliableMessaging
  - Allow applications to send and receive messages simply, reliably, and efficiently even in the face of application, platform, or network failure
- WS-Transaction
  - AtomicTransactions/ACID and Business Agreement which use Compensation transactions



The purpose of this slide is to bridge the gap between SOA and Web services implementations. This slide will be used in two places in the lecture throughout the week: (1) introduction and (2) web services lectures.

The right side of the slide is discussed during the Web services lecture and the left half of the slide is discussed during Day 1 during the introduction to SOA and SOMA.

Using a services modeling process which will be described in the Services Oriented Modeling and Architecture (SOMA) lecture we have three major types of outputs which help bridge our understanding of a service when we discuss SOA and Web services:

- service identification and specification
- component identification and specification
- service and component realization decisions

At design time, we identify and specify a service specification which provides the design interfaces which include the specifications of methods. The combination of the definition of the method and the interface at design time is what we refer to as a service from a SOA perspective.

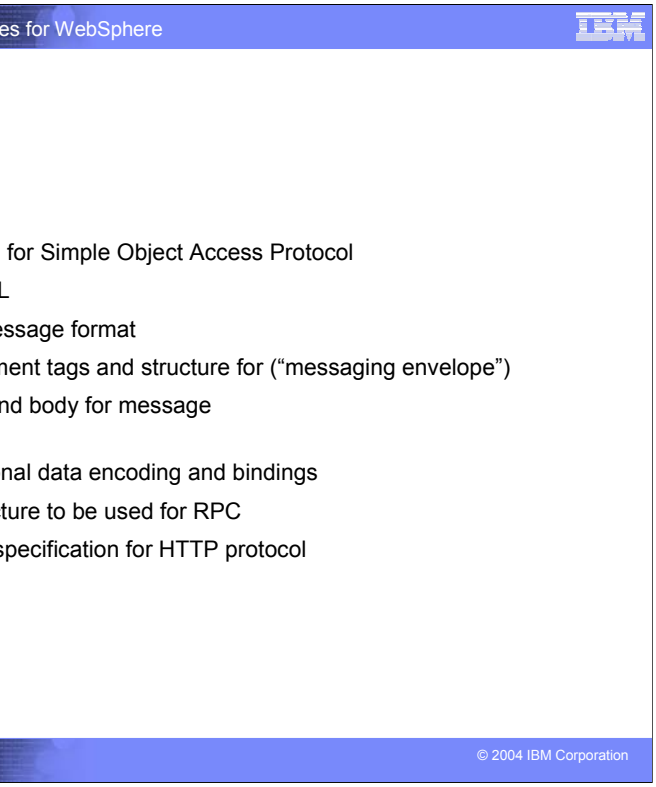
The notion of a use case provides a conceptual bridge between the design time and implementation notion of a service.

Use case is a key component of services modeling which identify services that are defined at design time by the design component's interfaces and methods. That in, turn are realized by the web services definition and implementation.

Marrying the web services implementation described by WSDL to the design time concept of a service, the combination of porttype and operation correspond to the design time notion of a service.

The design interface corresponds to the WSDL porttype and the design notion of a method corresponds to an operation on the corresponding porttype of the WSDL service description.

SOAP Protocol



When SOAP was first introduced it stood for Simple Object Access Protocol, when it was turned over the W3C they dropped the acronym meaning.

Design Principles

KISS

Vendor neutral

Any language, object model, platform...

Scriptable

Firewall friendly

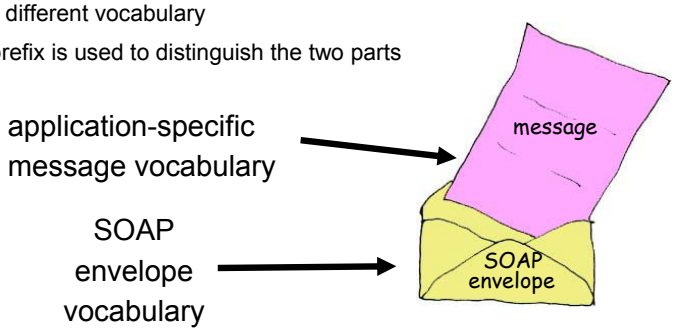
Infrastructure concerns delegated to processing intermediaries

Flexible layering....substitutable:

- Transport bindings
- Language bindings
- Data encodings

SOAP RPC Message Structure

- Request and Response messages
  - Request invokes a method on a remote object
  - Response returns result of running the method
- SOAP defines an "envelope"
  - "envelope" wraps the message itself
  - message is a different vocabulary
  - namespace prefix is used to distinguish the two parts



Messaging patterns

Document Oriented

Send/Receive any valid XML Document

Maybe respond to it (now or later); Can be used for asynchronous processing

Allows finer grain of control over what is being transmitted

Expose and exploit lower-level SOAP APIs to build the SOAP message and send it

RPC

Simple Request-Response protocol

Send a request and expect an immediate response

Lower-level SOAP methods build and send messages

Hide the messaging nature of SOAP

A SOAP Request Message

```
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://www.w3.org/2001/06/soap-envelope">

   <SOAP-ENV:Body>

      <m:GetLastTradePrice xmlns:m="Some-URI">
          <symbol>DIS</symbol>
      </m:GetLastTradePrice>

   </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

message

SOAP envelope

A SOAP Response Message

```
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://www.w3.org/2001/06/soap-envelope">

   <SOAP-ENV:Body>

      <m:GetLastTradePriceResponse
         xmlns:m="Some-URI">
         <Price>134</Price>
      </m:GetLastTradePrice>

   </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Result returned in Body

app-specific message

SOAP envelope

Why SOAP?

- Standard agreed to by all major industry players
- Use of existing protocol infrastructure (HTTP, SSL, SMTP, Messaging)
- Accessible software services on the web require:
  - Protocol -- how interactions are structured, who initiates requests and how responses are made. eg TCP/IP, HTTP, SMTP
  - Wire Format -- How information is organized when sent over a network.
- Need for simple, platform independent, standard for message exchange
- Many benefits to XML usage
  - Machine and human readable
  - Expressiveness, standardization, tools
- Extensibility
- Separation of application specific information (body) from quality of service and processing information (header)

By extensibility – use of XML namespaces, headers, etc

WSDL: Describing Services and Service Providers

WSDL = Web Services Description Language

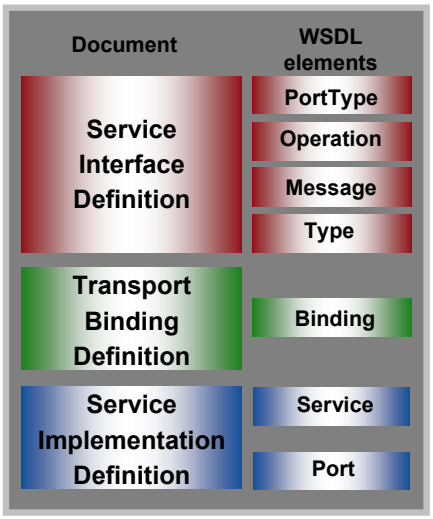
WSDL

- XML for describing Web services, similar in purpose to an IDL
  - Operations and messages performed/understood are described abstractly
  - Describes how this interface is bound to a transport
  - Endpoints for accessing the services
- Functional description of network accessible services
  - Service Interface
- Transport Binding Information – how messages are transported
  - Service Binding
- It contains operational information about the service
  - Service Implementation
- 'Contract' between the Service Requestor and Service Provider

Transport bindings are not limited to SOAP

WSDL Document

- **Service Interface**
  - Abstract, reusable service definition
  - Represents a type of service that can be implemented
  - Elements: types, message, portType
- Service Binding
  - how messages are placed on a given transport
  - Elements: types, message, portType
- **Service Implementation**
  - Implementation of one or more service interfaces
  - Contains the endpoint reference
  - Elements: port and service



At runtime, you can choose the implementation to use.

Part defined in the WDSL 1.1 Spec.

types, which provides data type definitions used to describe the messages exchanged.

message, which represents an abstract definition of the data being transmitted. A message consists of logical parts, each of which is associated with a definition within some type system.

portType, which is a set of abstract operations. Each operation refers to an input message and output messages.

binding, which specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType.

port, which specifies an address for a binding, thus defining a single communication endpoint.

service, which is used to aggregate a set of related ports.

WSDL Usage Scenarios

- As an IDL
  - tools can generate client access code and server template implementations for a service description
  - these tools can also create WSDL from existing code
- As a Binding mechanism
  - Extensible to support many types of protocols (not just HTTP)
- Standardized service interface descriptions
  - Allows advertisement and dynamic discovery of services
  - Enables dynamic binding to service
  - Complements UDDI registry
- **Best Practice** is to start by designing your WSDL for the interface map it to the server implementations

Binding is needed for type conversion.

WSDL was written to be extensible; the COBOL tags are from IBM for CICS. JCA and CICS are covered by the IBM extension.

XML, database, and OO are 3 different ways of looking at data.

Reality is most common is a meet-in-the-middle scenarios where you design the WSDL for the interface you want and then you have to provide a handcoded mapping into your backend systems.

How does the Requestor get the WSDL

- WSDL (or its URL) can be emailed to requestor
- WSDL for available services is stored at repository sites like xmethods.net or www.salcentral.com
- using WS-Inspection language on target site
- use UDDI "find" methods to look it up
  …in the UDDI Business Registry

UDDI: Universal Description, Discovery, and Integration

What is UDDI?

- Universal Description, Discovery, and Integration
- A project to speed interoperability and adoption for web services
  - Standards-based specifications for service description and discovery
  - Shared operation of a web-based business registry
  - Partnership among industry and business leaders - more than 70 companies have signed up so far
  - Was an independent organization - UDDI.org
- UDDI Standards work now turned over to OASIS
- UDDI has two pieces:
  - the UDDI Business Registry (hosts the data)
  - the API and data model (provides access to the data)

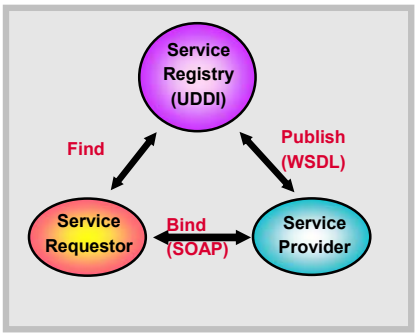
Google has published WSDL for google services; they have security routing, up to 1000 req/day.

IBM is a founding member.

UDDI Roles and Operations

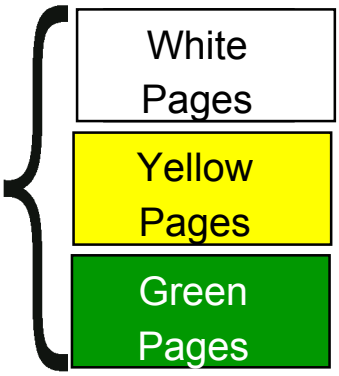
- Service Provider
  - Provide e-business services
  - Publishes availability of these services
- Service Registry
  - Provides support for publishing and finding services
- Service Requestor
  - Finds and Binds to services in the registry

UDDI defines publish and find API
Services provider define bind operations

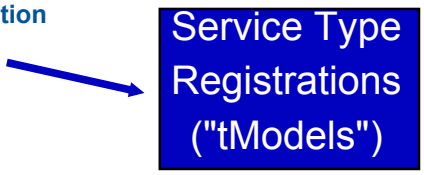


Registry Data

- **Businesses register public information about themselves**



Standards bodies, programmers, businesses register information about their service types



Standards and Organizations

Values of Open Standards for Clients

- Freedom of choice
  avoid vendor lock-in
- Consistency and productivity
  proven to work
- Skill availability
  widely taught and practiced
- Best practices
  expert driven
- Simplification
  don't reinvent the wheel
- Cost reduction
  as the result of all of the above

Sam Palmisano, IBM CEO, November 2002

"Our software business is completely committed to open industry standards, including the stuff that we're driving in open-infrastructure middleware with web services and WebSphere."

If we look at the following list, “Standards” presents the following benefits to clients.

-The freedom of choice (to choose among a different set of vendors implementing that standard).

-A common model where things are proven to work (such that risk is reduced and knowledge is gained by other applications using the same standard)

-With a common model, there is much more availability of resource that can be obtained (since the information isn’t spread across a set of disparate technologies)

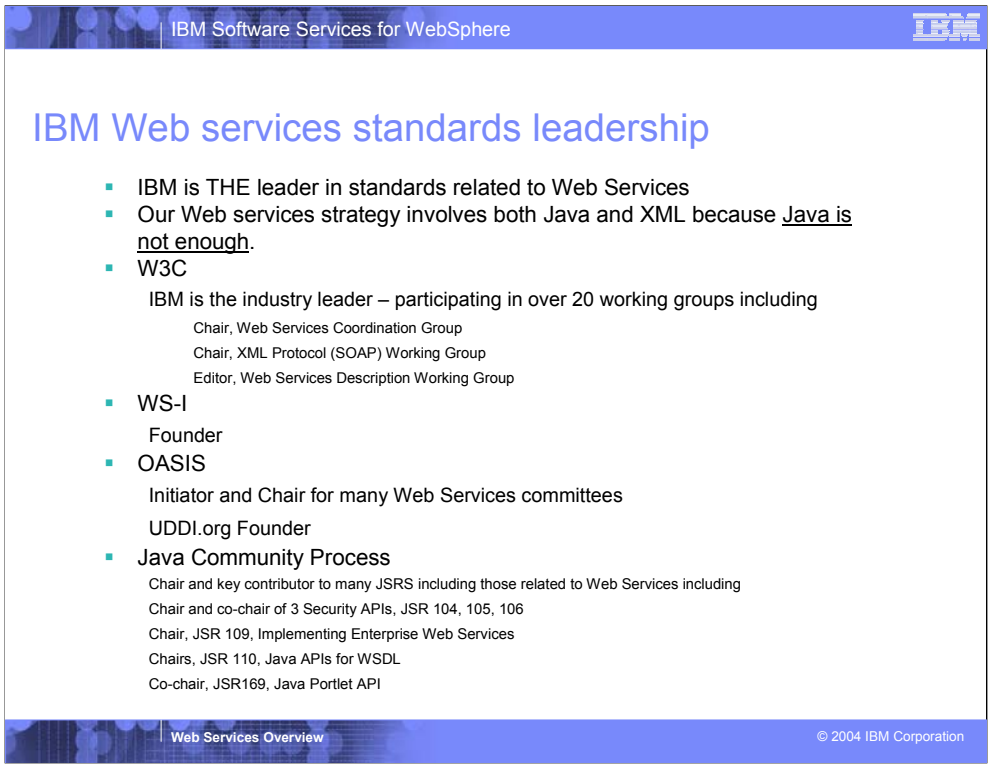
-As standards get defined, best practices are defined to use them (which are then used as templates for development)

-Simplification (as best practices, design patterns, code, etc… can all be reused)

This leads to long-term cost reduction to the end user (which benefits from these items).

IBM and WebSphere is committed to driving open-infrastructure middleware with WebSphere and web services.

See **Sam Palmisano** interview at http://www.fortune.com/fortune/print/0,15935,389941,00.html with **FORTUNE**, Sunday, November 10, 2002 By David Kirkpatrick.



As the preceding slides already indicated, IBM plays a significant role in the creation of many if not all of the relevant specifications in the web services space. We are actively participating in the standardization bodies and working groups that take the specs and turn them into widely accepted standards, like W3C or OASIS.

WS-I

Web Services Interoperability Organization (WS-I) evolution

- Challenge :
  - Each standard deals with a specific problem
  - Customer solutions utilize multiple standards
  - Infinite ways to interpret the standards
  - Customers have multi-vendor environments
  - Interoperability was very difficult (if not, impossible)
- Answer:
  - Web Services Interoperability Organization (WS-I)
    - Founded in Feb. 2002 by IBM and Microsoft
    - Expanded to 120 members world wide
  - Input
    - Standards Organizations
    - Industry requirements
  - Output
    - Profiles
    - Requirements back to Standards Organizations

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

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

WS-Basic Profile 1.0

- SOAP1.1
- WSDL1.1
- UDDI2.0
- XML Schema
- XML1.0 (Second Edition)
- HTTP1.1
- Other supporting/referenced specs/standards

Basic Profile – that was then, this is now

- That was then…
  - BP 1.0
- This is now…
  - BP 1.1
  - SSBP 1.0 - Simple SOAP Basic Profile
  - AP 1.0 - Attachments Profile – SOAP with Attachments (Sw/A)
  - As of August 3, 2004, member approved drafts – soon to be final
- What's the difference?
  - BP 1.1 + SSBP 1.0 == BP 1.0 (plus errata)
  - AP 1.0 is the attachments work that was never addressed in BP 1.0
  - A vendor can either support BP1.1 + SSBP1.0 or BP1.1 + AP1.0. (This gives Microsoft a way out of supporting Sw/A.)
- Work going on for WS-Security Profile
  - Basic Security Profile Version 1.0 – draft released May 12, 2004

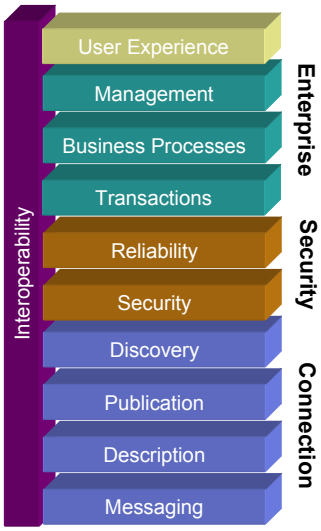
Example Requirement

- 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.*

IBM's View of Web Services

- A Web Services is described by WSDL
  - not limited to SOAP
  - multiple protocols
  - both synchronous and asynchronous
  - both RPC and document oriented
- Evolutionary
  - add web services to your existing designs
  - does not require a radical redesign
  - Web Services will not supplant other distributed technologies. They will supplement them.
- You have to start with good distributed design principles
  - Proper layer design and layer placement is critical to success
- Enabling technology for Service Oriented Architectures and On demand systems
- Standards, Standards, Standards
  - Interoperability is key
  - Use open source and commercial tools wisely and adhere to standards



IBM's view of Web Services is unique. We view a Web Services as a software component (service) that is described by WSDL. Issues like envelope protocol - SOAP, and transport mechanism – HTTP, JMS, and style – RPC, document, may vary, but the service is described with WSDL. This is the broadest view of web services in the industry.

We see enterprises adapting their existing systems for a Service Oriented Architecture based around WSDL described services.

Existing J2EE need to have a good core design as a starting point – good layering, and a MVC type structure. Standards for web services (JSR109 and JAX-RPC) have been added to J2EE 1.4 and are designed along the same principals as EJB, JCA, and other J2EE technologies with declarative aspects. We now have the standards for how to expose EJBs and Java Beans as web services in a standard, portable way.

Legacy systems such as CICS transaction may or may not make be good to expose as web services as is – depends on the granularity of the CICS transactions. If your legacy and existing systems are well structured then you are well positioned to take advantage of a Service Oriented Architecture, the foundation of e-business on-demand. If your legacy is not well structured, then you will need to re-factor or adapt the systems. We will talk about this approach later.

Resources

- IBM developerWorks web services domain
  http://www-106.ibm.com/developerworks/webservices/?loc=dwmain
- IBM alphaWorks
  http://www.alphaworks.ibm.com
- IBM web services homepage
  http://www-4.ibm.com/software/solutions/webservices/
- Web Services on WebSphere (WoW)
  http://www.ibm.com/websphere/wow
- World Wide Web Consortium (W3C)
  http://www.w3.org/
- Java Community Process (JCP)
  http://www.jcp.org
- Oasis
  http://www.oasis-open.org/home/index.php
- Apache
  http://xml.apache.org
- WS-I
  http://ws-i.org

References

- **WS-I Org**
  http://www.ws-i.org
- **Developer works articles**
  http://www-106.ibm.com/developerworks/webservices/library/ws-basicprof.html
- **New WS-I sanctioned WSDL schemas**
  http://schemas.xmlsoap.org/wsdl/2003-02-11.xsd
  http://schemas.xmlsoap.org/wsdl/soap/2003-02-11.xsd
  Same as what is currently posted at namespace URI
- WS-I Sample application on alpha works
  http://wsi.alphaworks.ibm.com/IBMShowcase/

Additional Information

Sept 2004

Web Service Implementations

WebSphere 5.1/5.0.2

- IBM WebSphere Web Service engine
- Business Process Execution Language for Web Services (WBI SF 5.1)
- Implements Web Services Standards
  - JAX-RPC: Java APIs for XML RPC (JSR 101)
  - JSR 109 – Web Services for J2EE
  - Ws-security - OASIS draft 13 specification and Username token profile draft 2
  - Supports WS-I.org Basic Profile 1.0
- Enhanced performance – SAX style parsing
- SOAP/JMS and SOAP/HTTP support for Web Service
- Web Services Gateway (SOAP over HTTP channel and provider)
  - Web Services Invocation Framework (WSIF) SOAP over HTTP provider
- UDDI support for custom taxonomies

JSR 101 - is the Java standard for web Services

WebSphere will not interoperate with ws-security version 1.0 until WAS version 6.0, Microsoft is shipping ws-security 1.0 compliance now – this is a big problem for interoperation.

What do we have today ?

WebSphere Application Server v5.1
- Contains IBM SOAP Engine (Maelstrom)
- Contains the Apache SOAP 2.3 runtime
- Contains the Apache SOAP 2.2 runtime for backwards compatibility

Network Deployment v5.1
- Private UDDI Registry
- Web Services Gateway

WebSphere Business Integration Server Foundation v5.1
- BPEL4WS - Business Process Execution Language for Web Services

WSAD v5.1.1
- Fully supports the WAS v5.0 and WAS v5.1/5.0.2 runtime
- Web Service Explorer
- WSDL editor
- Support for generating Web Services from WSDL, EJBs, JavaBeans, DB2
- Support for generating Clients from WSDL

WSAD-IE v5.1
- Development of BPEL processes for WBI SF 5.1
- Support for enterprise (legacy) services

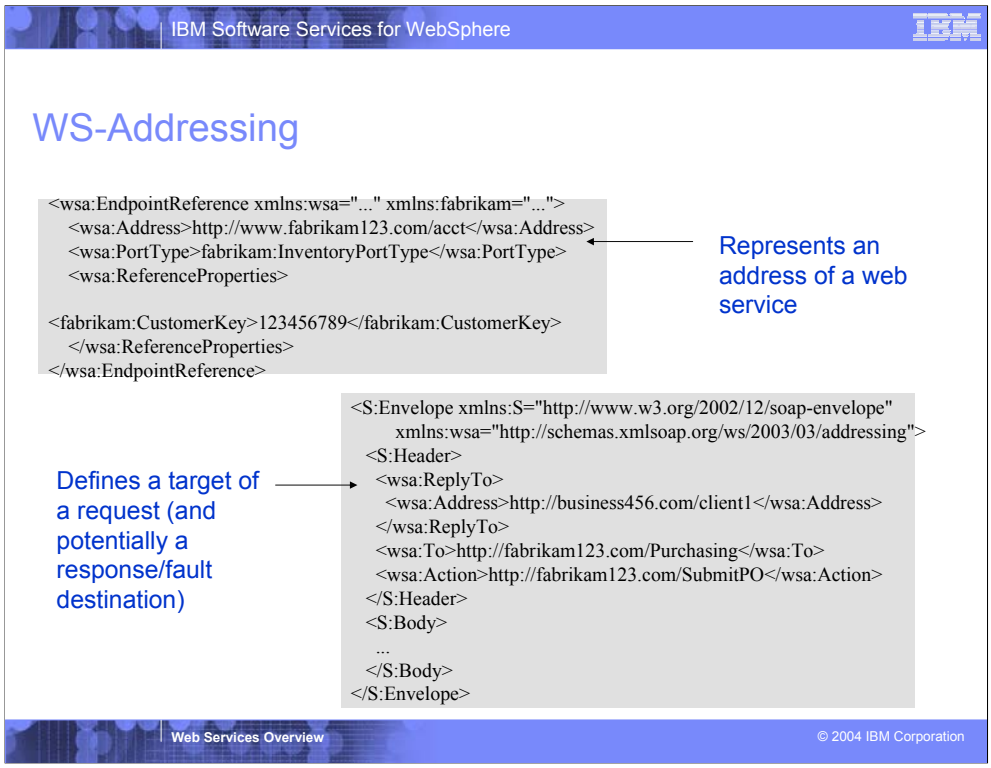
Does not support the WAS 5.0.2

The W3C focuses mostly on interoperable technologies that are more web related (including HTML, HTTP, XML, and SOAP)

OASIS is focues on the development/adoption of web services QOS standards (WS-Security, WS-Distributed Management, etc…) along with other application-specified standards like BPEL, UDDI, WSRP, etc…

The JCP holds a different focus. It's focus is on the standardization of Java technology (and developing a programming model which is consistent across Java vendors).

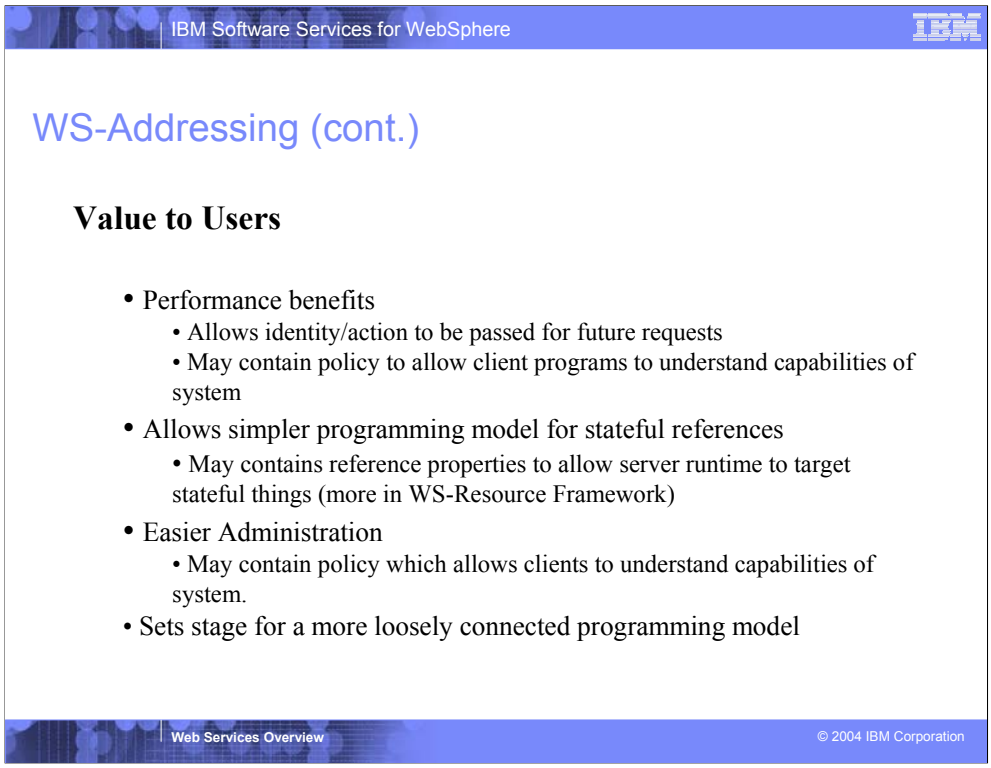
Lastly, the WS-I organization isn't meant to focus on the creation of any particular technology standard, but more on promoting interoperability across vendor runtimes (and focuses more on resolving ambiguities and contradictions between specs, as they try and achieve interoperability between different vendors).



Let's focus on the first specification, WS-Addressing. This is a building block specification (used by a variety of others). There are 2 main parts of the specification:

The first part focuses on defining what an endpoint address is (the address of a web service, and a potential of other pieces of information including portType, policy, and reference properties).

The second part focuses on a message information header (sent on web service requests to define the target and action of a web service) along with perhaps other additional information like a replyTo or faultTo header.



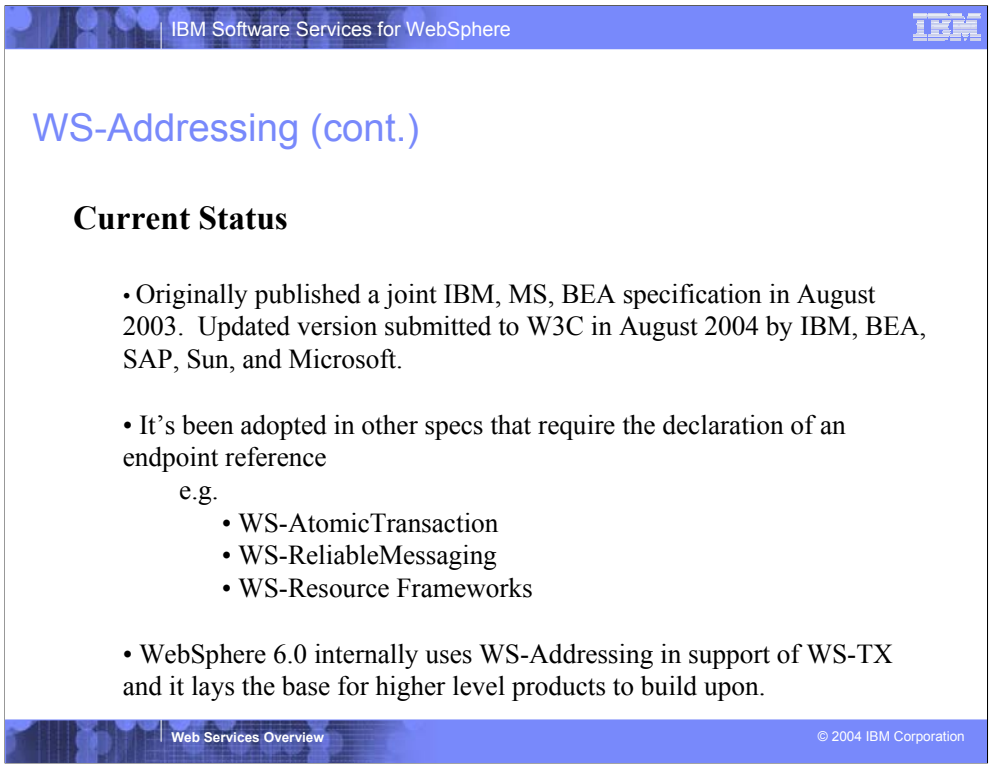
Now, focusing on the value for users, there are lots of benefits to WS-addressing.

Performance benefits. Since the request can now contain more information about the request itself (such as target and action identifier), it can now classify the work more easily (without having to look at the contents within the body to determine the action. In addition, if an EPR is returned which contains policies which display the capabilities of a system, it does not have to query that system to understand what it's policies are (saving on multiple remote requests which tend to slow down performance).

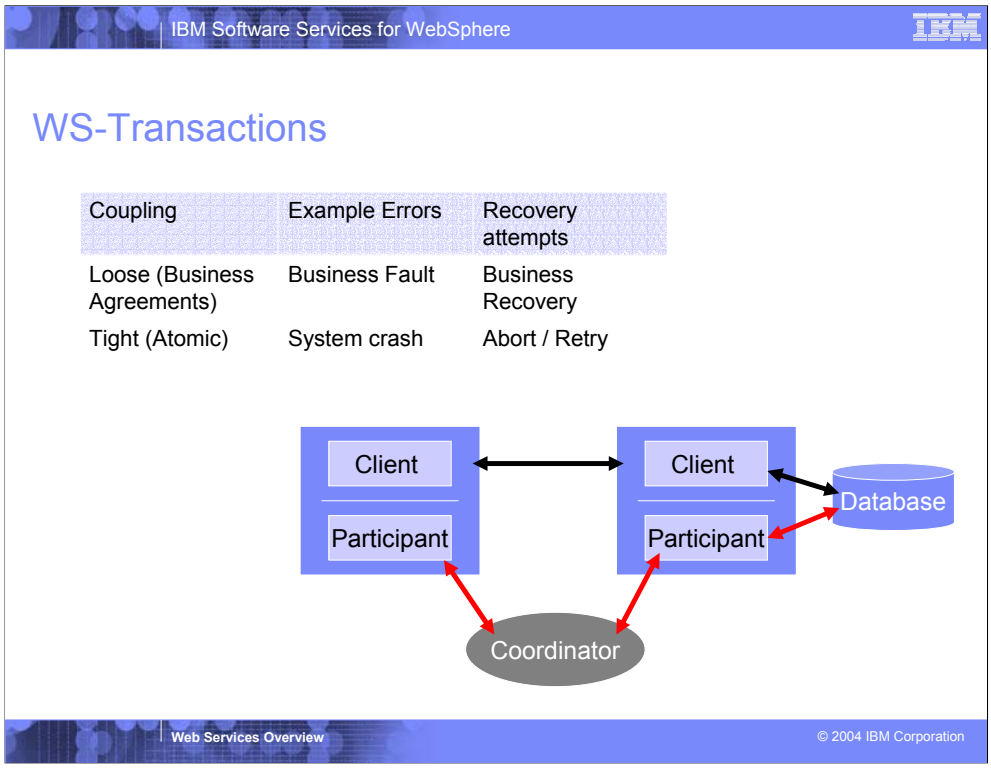
Similar programming model. Without this mechanism, clients have to resort to either exposing properties (key's) which hold a reference to a clients state on a server system to the application (similar to what BPEL does today), or hide the key in another mechanism like transport header, e.g., HTTP cookie, which can be used to store state related to the specific client issuing the request.

Easier Administration. If a server can declare it's capabilities (e.g., the ability to accept compressed messages), the client can automatically use those capabilities without requiring static configuration. This allows a much more functional and dynamic model than available today.

Loosely coupled programming model. Now that addresses can be inserted into the request for reply/faultTo information, the same socket connection is no longer required to send the response (allowing for a more loosely coupled interaction model).



The current status of the specification is that an updated version of the WS-Addressing specification has been contributed to W3C in August 2004. It has since also been adopted and used as a building block in other higher-level specifications that need the ability to pass around references to web services endpoints.



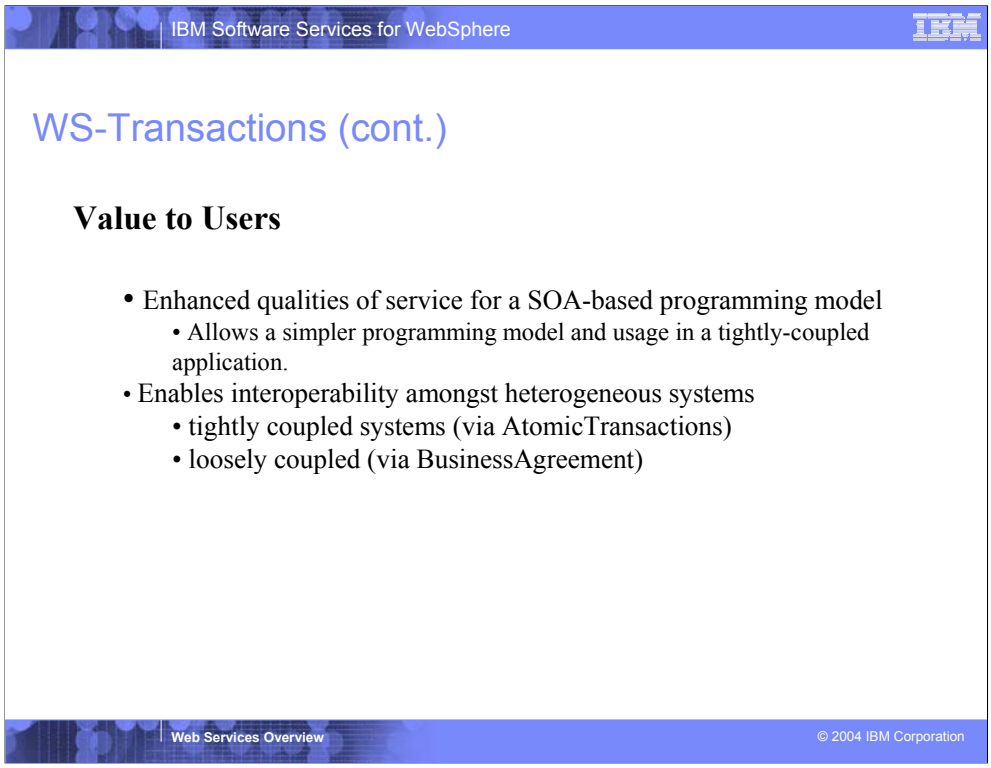
Now, I would like to move the focus to the next specification, WS-Transactions. There are two parts of WS-Transactions:

-AtomicTransactions - for tightly coupled systems where ACID properties are required

-Business Agreement – for loosely coupled systems where Compensation transactions can be used (where ACID properties and long-lived locks are not required).

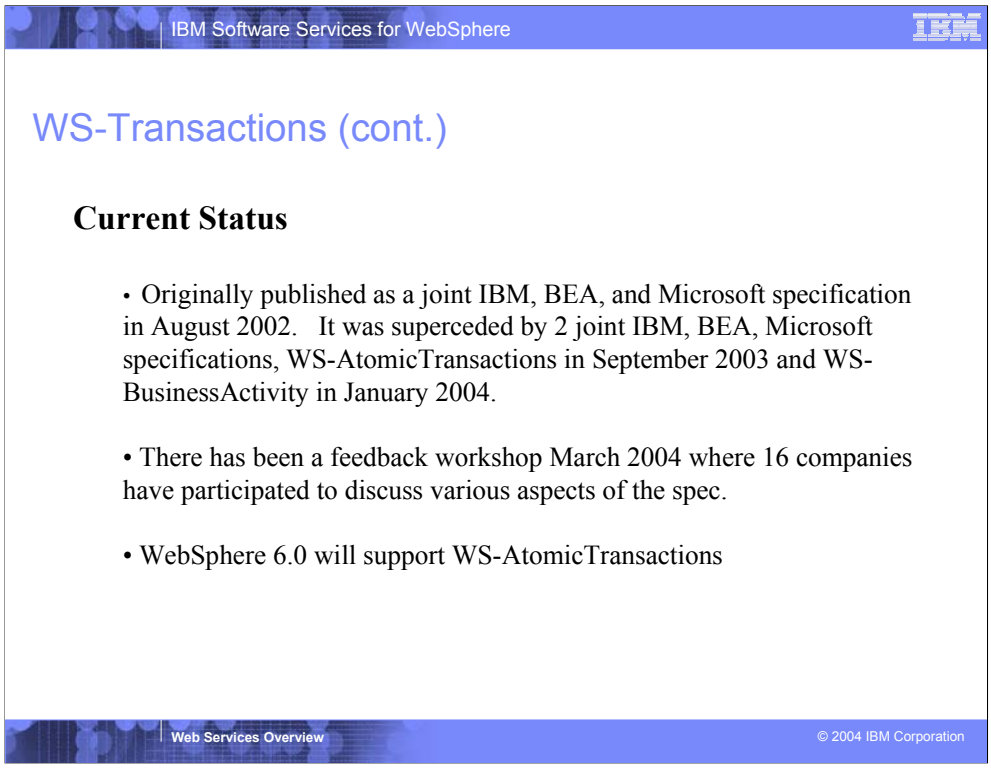
In the Atomic transactions case, participants in the transaction are registered with a coordinator, and during and when the transaction is about to end, protocol negotiation is then achieved to prepare/commit the transaction amongst the participants.

In the BusinessActivity case, where users don't want to lock resources, services are requested to do work, and if they have the ability to undo work, they are responsible for registering with the coordinator that they have the ability to undo the work. That way, if compensation needs to be done, it can be done driven by the coordinator. Therefore, the responsibility for recovery is put into the hands of the service provider (and their ability to compensate for work done).



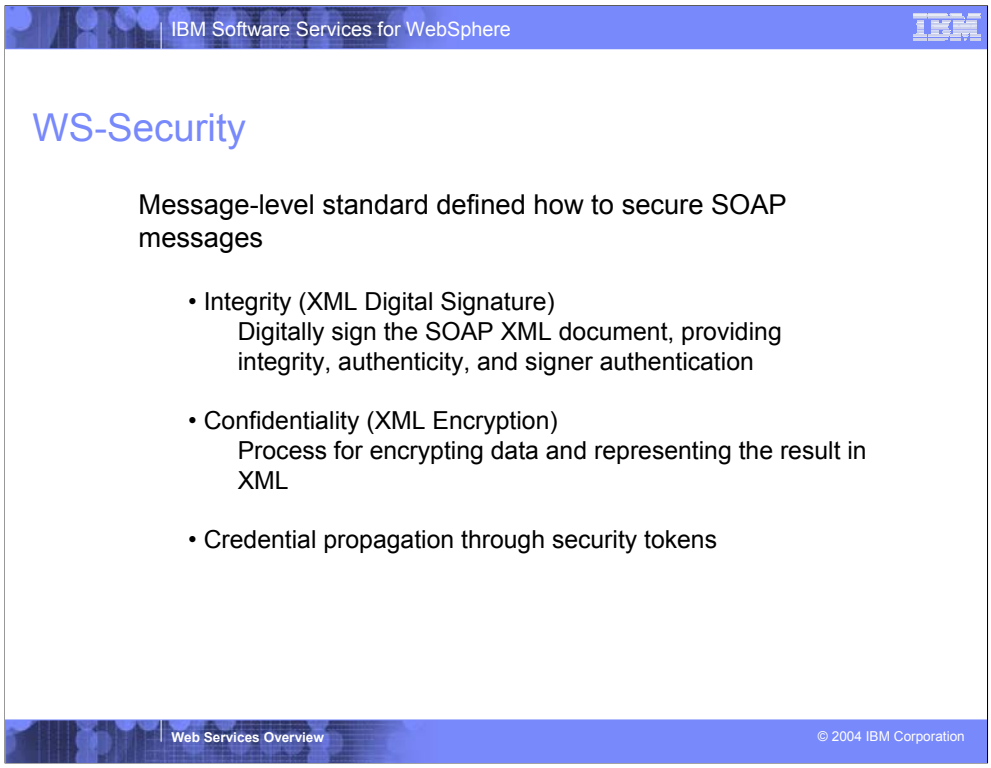
As web services start to become popular, we are starting to see a large push to move to a services oriented architecture (some due to a simplification and standardization of a programming model). However, this hasn't reduced the need for tight coupling in intra-enterprise applications. Therefore, WS-TX gives a higher QOS over the programming model.

Web services is also about interoperability (not just loose coupling). Therefore, The WS-AtomicTransactions and WS-BusinessActivity specs give the user both a tightly-coupled and loosely-coupled model for dealing w/resources that need to be coordinated.



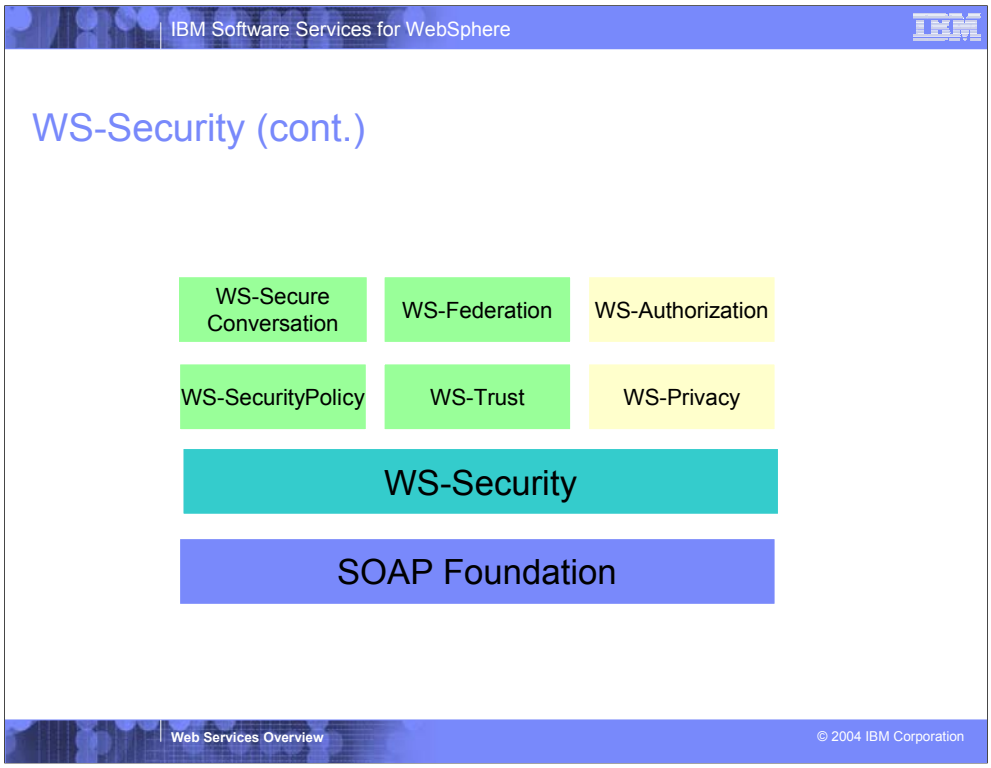
The current status of the spec is that 2 specs superceding the original have since been published in Sept. 2003/Jan 2004 for AtomicTransactions and BusinessActivity. There has been a feedback workshop in March 2004 where 16 companies have participated to discuss aspects of the spec.

WebSphere 6.0 will support AtomicTransactions.



That said…

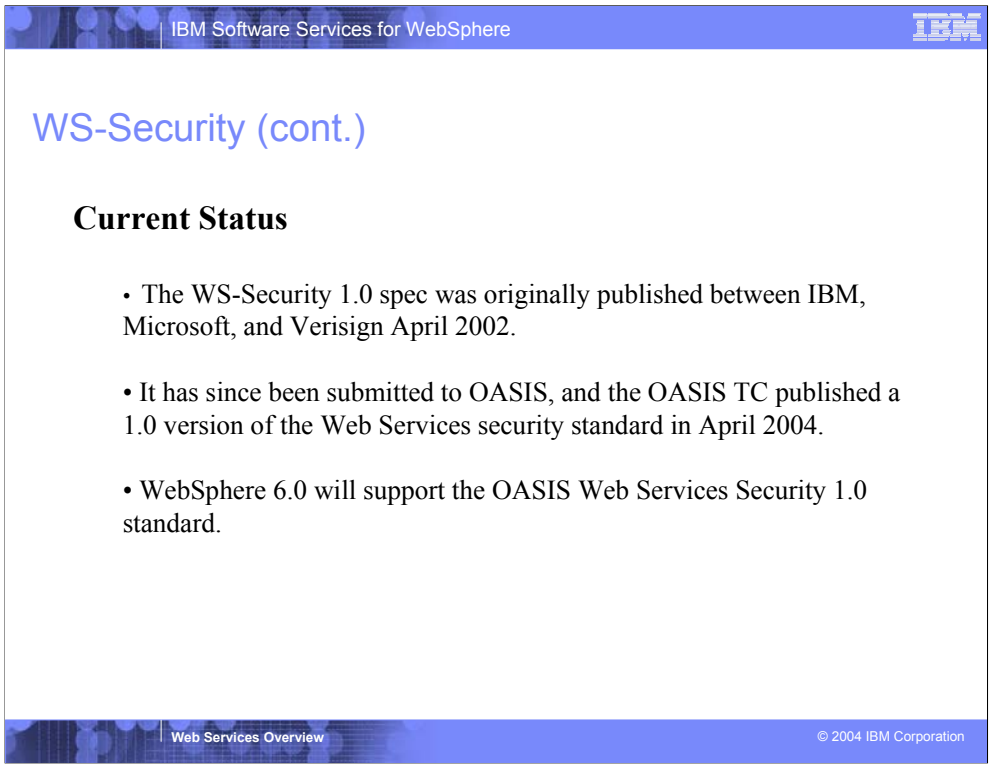
WS-Security provides for integrity, confidentiality, and the principal for web services requests.



It's a fundemental spec to discuss how to secure a web service request. It's used as a building block to support higher level specs. For example, WS-Trust needs to communicate with a secure token service to create a secure token. Therefore, it must secure those requests (as well as the application requests which will eventually use this token).

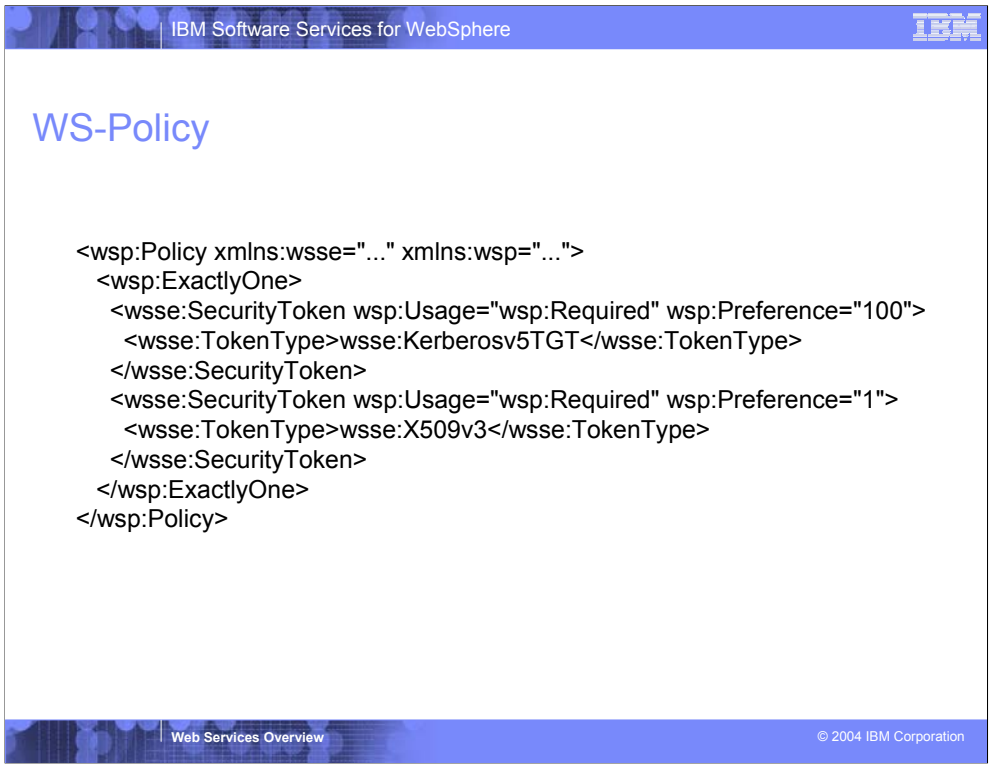
There are still other web services specs which could also help simplify configuration for a user (like WS-SecurityPolicy) – which sits upon WS-Policy and defines a mechanism for identifying requirements of a server.

The value statement is fairly basic, it provides secure communication of web services. However, since the security is targetted against a message (vs. a transport), the security information is available to flow with the message through intermediaries keeping it safe and secure.



WS-Security was used as the basis of the OASIS TC WSS spec, and OASIS has published their 1.0 standard in April 2004.

WebSphere 6.0 will support this WSS 1.0 standard.



WS-Policy provides an XML-based grammer which allows the description of the capabilities and requirements of a receiver or a sender of web services requests/responses. The requirements/capabilities come in a form of assertions and may (or may not) have any wire format translation.

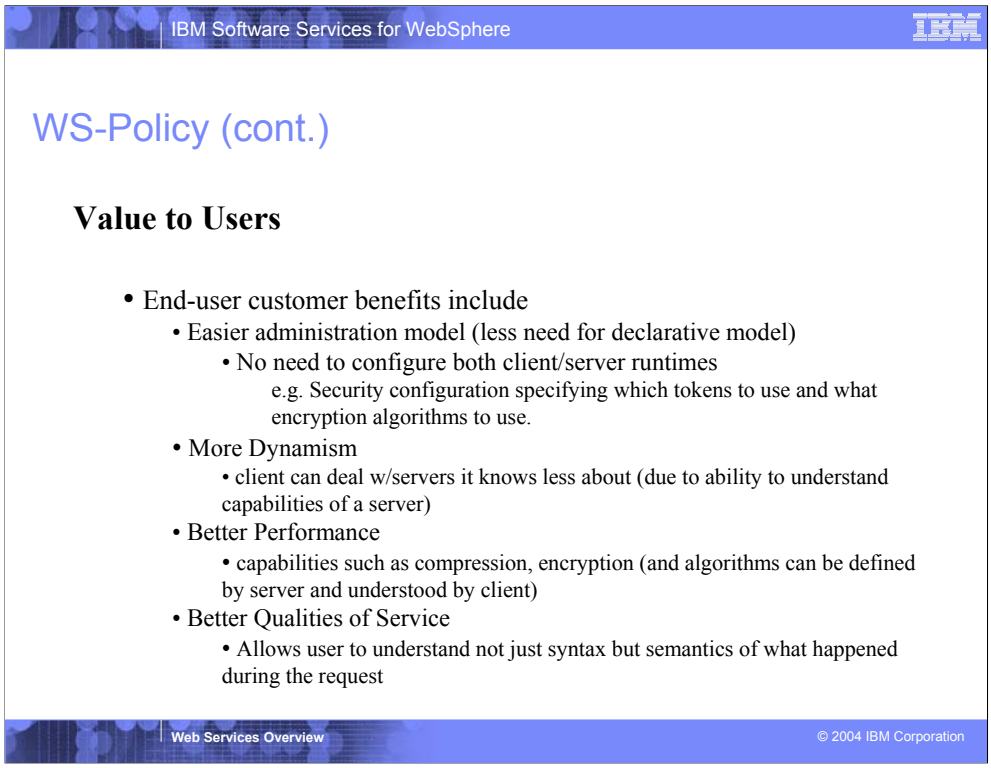
The base spec doesn't define how these policies are obtained or exchanged (but we've already learned one way – through EPR references, it allows policies to be exchanged).

Other specs like WS-PolicyAttachments or WS-MetaDataExchange talk about how to attach policies to WSDL documents or how to request policy information from a service

- Looking at items that we have talked about already, you can already see where some benefit can be obtained. Take for example Security. If a server describes that they can handle (or require) only one or a smaller set of tokens as policy data, or perhaps asserts to use a specific encryption algorithm in order to talk to it, this is information which doesn't possibly have to be defined on a client side (on a per reference basis). Therefore, configuration of the security models becomes much easier to deal with.

- In fact, it permits a more dynamic behavior model (in a realistic business environment setting where you can look up services, and adjust your runtime behavior accordingly) vs. having to statically configure your runtime.

- Analogies to this in the CORBA world would be something like tagged components.

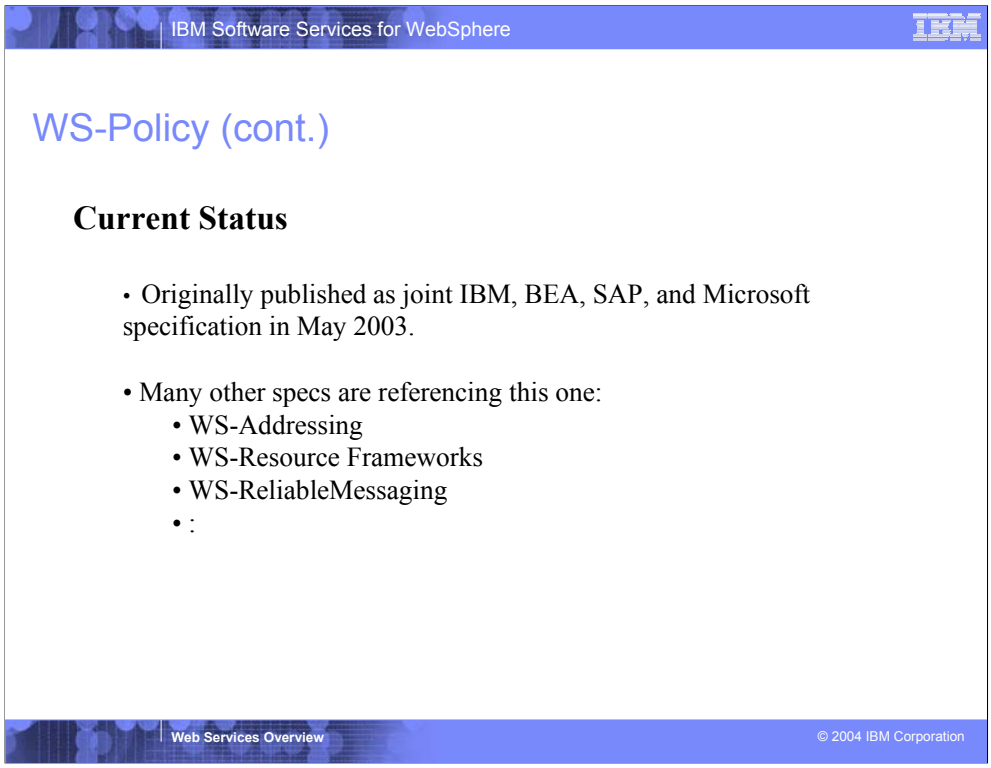


With respect to the value to the end user, this can be seen in a variety of ways: Easier administration model. In the absence of declaring capabilities, both sides of a communication stack much be configured such that it understands the requirements to interact with the other side. This may be defining security tokens accepted, encryption algorithms that can be used, etc…

This will lead to a more dynamic environment than is capable today (allowing the hosting of many more on-demand type scenarios).

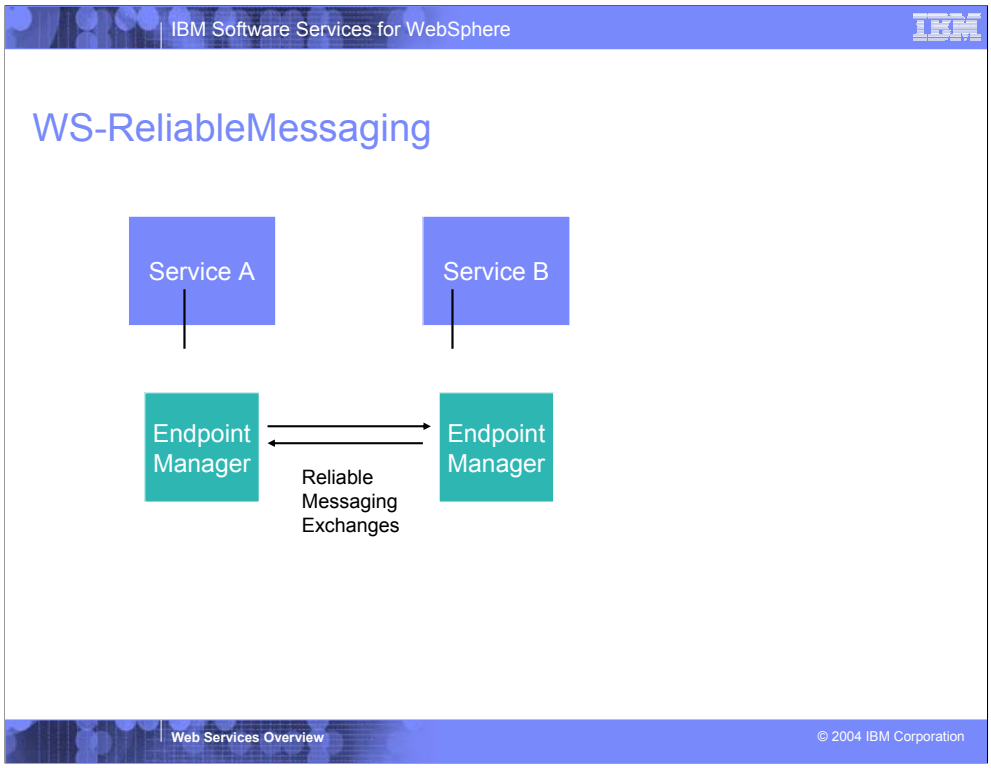
From a performance perspective, have the ability to get policy (perhaps along with an EPR reference or attached via WSDL will lead to enhanced performance in that it doesn't have to go out and query the system to see what it's capable of handling).

Lastly, it will lead to better qualities of service (and understanding how that Web service can be used). For example, a user would want to know whether or not that the service he is participating in will participate in a global transaction or not. This may affect the outcome of their application program – so they need to know.



The current status is that the WS-Policy spec was published by IBM, BEA, SAP, MS in May 2003.

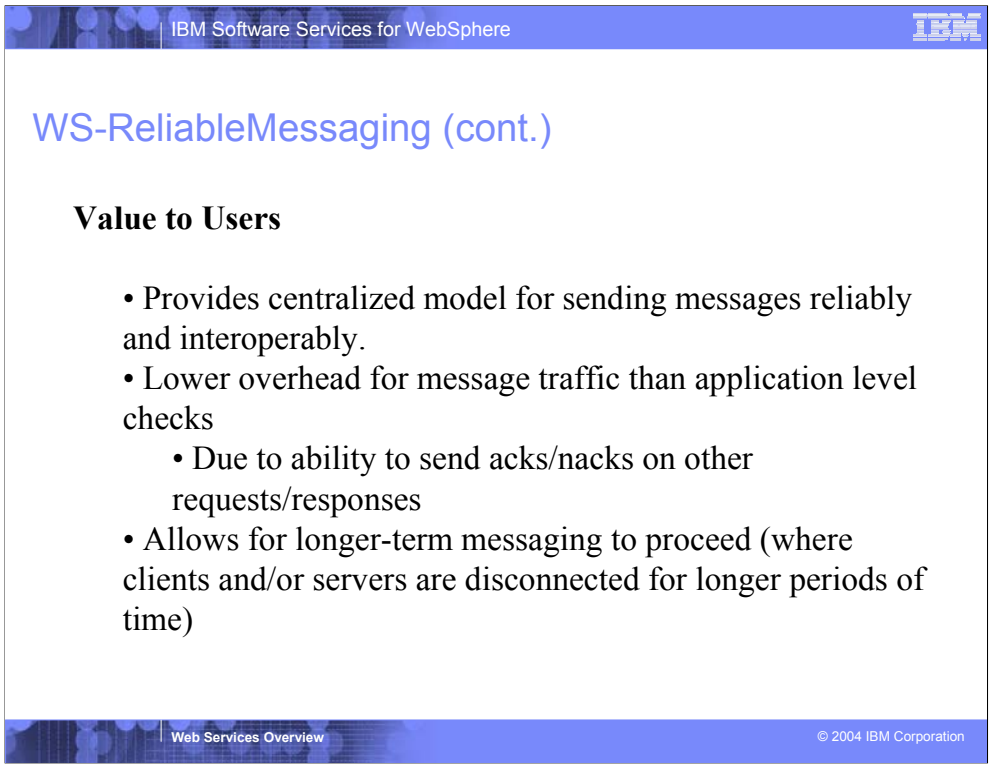
It is used as a basic specification and is referenced in other higher level specs.



Messages exchanged between the distributor and supplier travel over multiple nodes, some on the public Internet. This means that some messages may be lost in transit. Additionally, it is possible that either the supplier's or the distributor's system may fail while a message is in transit, leaving the still-running system in a state of confusion as to whether a given message has been processed or not. The goal of reliable messaging in Web services is to allow applications to send and receive messages simply, reliably, and efficiently even in the face of application, platform, or network failure. The WS-ReliableMessaging specification, recently published by BEA, IBM, Microsoft, and Tibco defines a protocol and a set of mechanisms that allow developers of Web services to ensure that messages are delivered reliably between two endpoints, and supports a broad spectrum of delivery assurance and robustness characteristics.

The most important prospect for the WS-ReliableMessaging protocol will be to provide the standard protocol for interoperability between different vendors' message oriented middleware environments. In this scenario, we envisage that WS-ReliableMessaging would be augmented with a set of standardized WSDL portTypes and bindings that would be specific to endpoint managers. These standardized portTypes and bindings would be implemented by vendors of messaging middleware products to enable messages from other middleware environments to be reliably, and interoperably exchanged with their own.

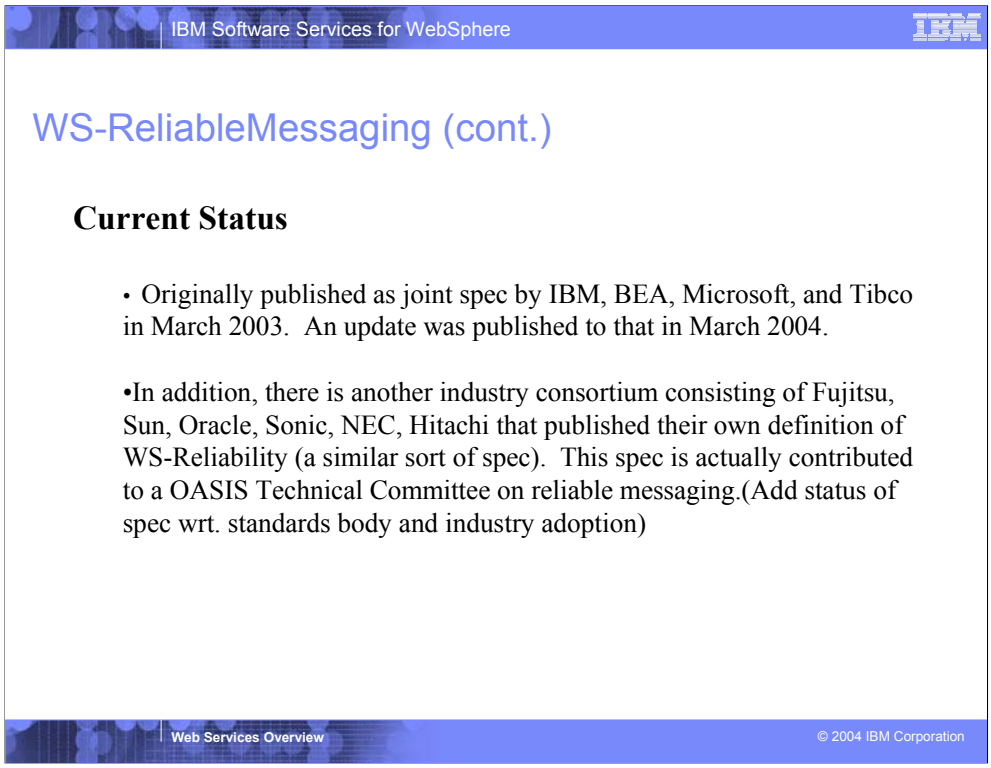
The basic model of WS-ReliableMessaging is fairly simple to understand. Under WS-ReliableMessaging, the source node sends a normal Web service message containing a WS-ReliableMessaging header. Upon receipt of the message, the destination node sends an acknowledgement message back to the source indicating that the message was successfully delivered.



There is plenty of opportunity for value with WS-ReliableMessaging. Apart from the enhanced quality of service that it gives you in an interoperable fashion, since the RM checks take place within the infrastructure, there's an opportunity for a lower overhead of message traffic than you would get trying to achieve the same at an application level.

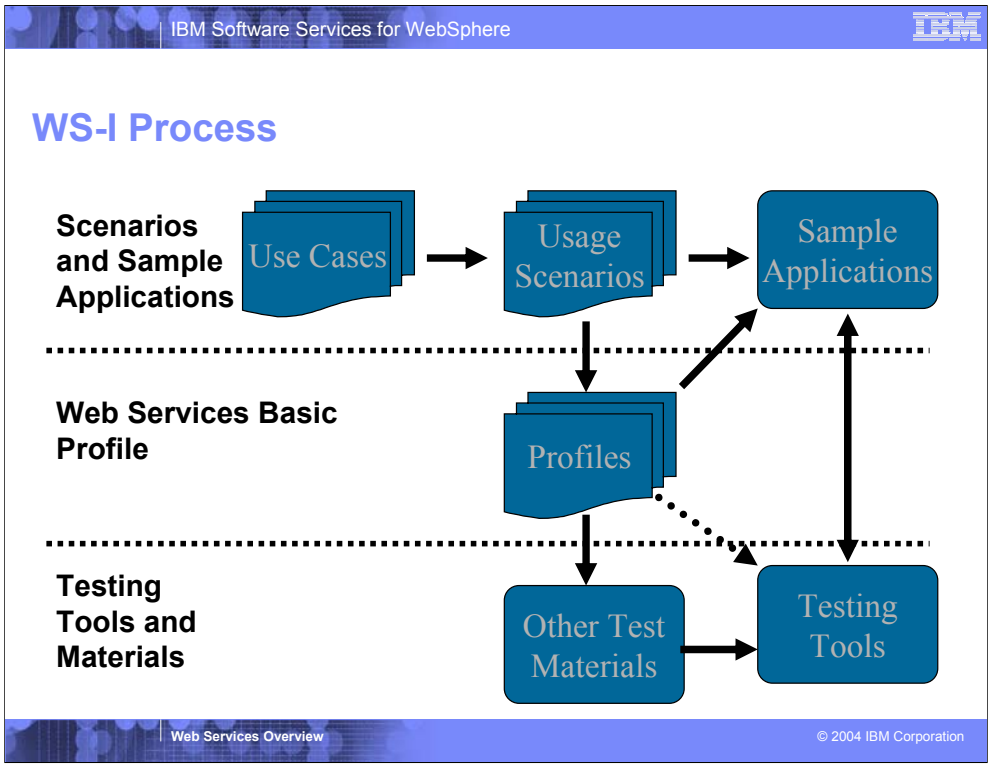
Similarly, you know have a much more beneficial proposal for longer-termed message request/response flows where clients/servers can be disconnected from one another.

Note however, the WS-ReliableMessaging specification alone does not address a number of important considerations that must be taken into account to realize levels of robustness, integrity, and performance that are often required by applications.



IBM, MEA, MS, and Tibco defined a spec March 2003 for WS-ReliableMessaging and published an update to that March 2004.

In addition, there is another industry consortium consisting of Fujitsu, Sun, Oracle, Sonic, NEC, Hitachi that published their own definition of WS-Reliability (a similar sort of spec). This spec is actually contributed to a OASIS Technical Committee on reliable messaging.(Add status of spec wrt. standards body and industry adoption)



The Test assertions from the profile identify the requirements for testing

Ie:

•Artifacts to be tested

•Assertions to be tested

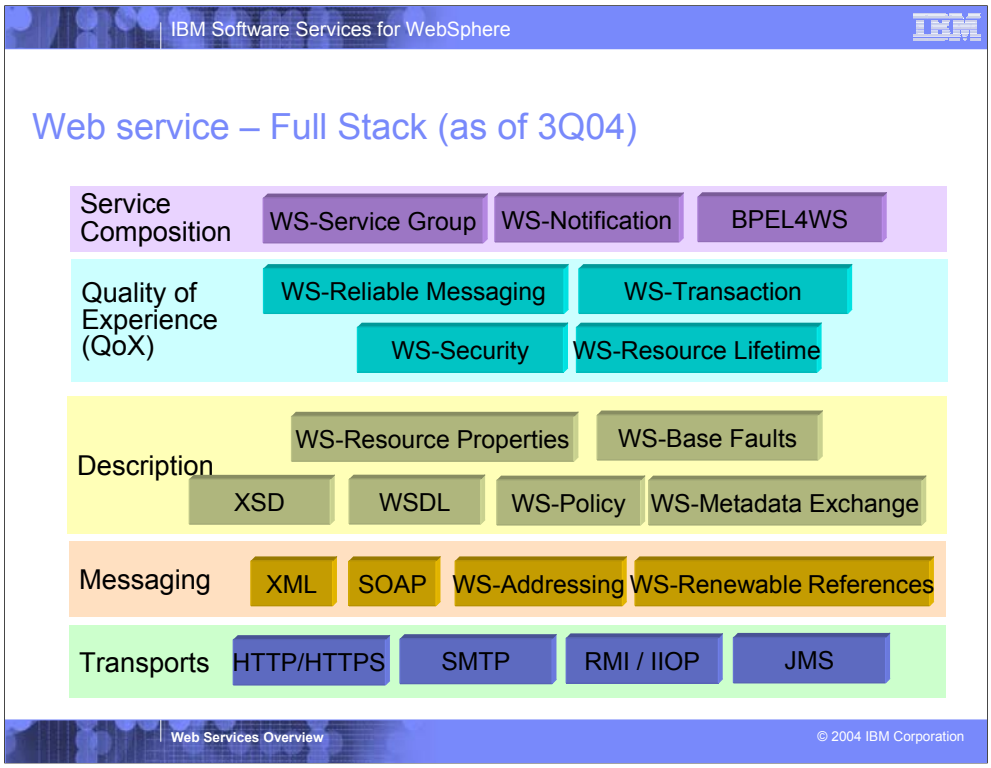
Basic Profile 1.0 Technical Highlights

- 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 (continued)

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


IBM Software Services for WebSphere
Backups
Sept 2004
© 2004 IBM Corporation



Therefore, looking at the web services specifications, you will see on some, a focus at various layers (transport, Messaging, description of the web services, qualities of service, and a higher level set of compositeability.

The W3C focuses mostly on interoperable technologies that are more web related (including HTML, HTTP, XML, and SOAP)

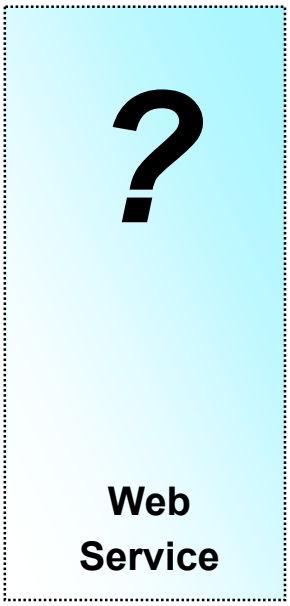
OASIS is focues on the development/adoption of web services QOS standards (WS-Security, WS-Distributed Management, etc…) along with other application-specified standards like BPEL, UDDI, WSRP, etc…

The JCP holds a different focus. It's focus is on the standardization of Java technology (and developing a programming model which is consistent across Java vendors).

Lastly, the WS-I organization isn't meant to focus on the creation of any particular technology standard, but more on promoting interoperability across vendor runtimes (and focuses more on resolving ambiguities and contradictions between specs, as they try and achieve interoperability between different vendors).

What is a Web Service?

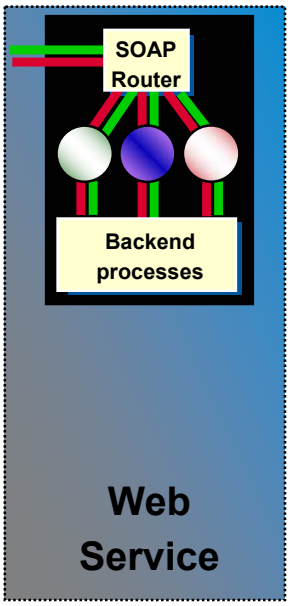
- **"Web services are software components described via WSDL which are capable of being accessed via standard network protocols such as SOAP over HTTP."**



What is a Web Service?

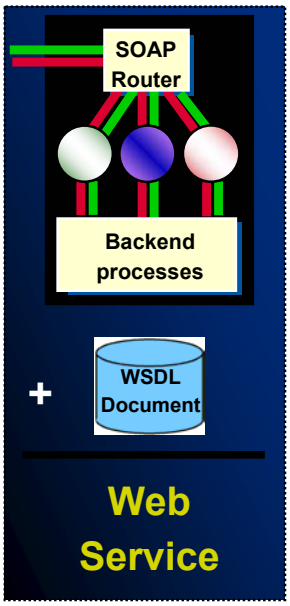
- **"Web services are software components described via WSDL which are capable of being accessed via standard network protocols such as SOAP over HTTP."**
- Today, SOAP over HTTP is the common protocol for Web services.
- For now, a SOAP interface connected to application processes can be thought of as a minimum...
- Other protocols besides SOAP can be used

...but by itself does not address rapid integration.



What is a Web Service

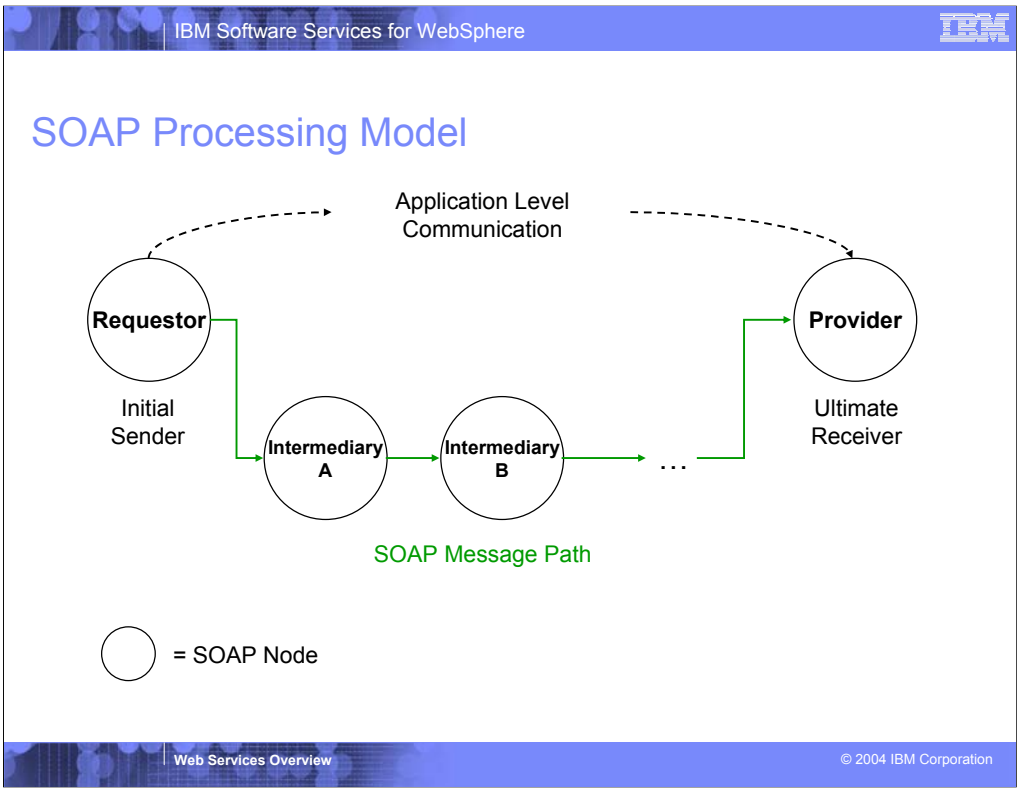
- **"Web services are software components described via WSDL which are capable of being accessed via standard network protocols such as SOAP over HTTP."**
- WSDL descriptions can be used to drive assembly tools, code generators, and other tools to speed integration.
- For now, SOAP+WSDL can be thought of as the base technologies for any Web service.
  - UDDI, other technologies can be considered optional, to add on as makes sense for the application



Two Main Message Exchange Patterns (MEPs)

- Document Oriented
  - Send/Receive any valid XML Document
  - Maybe respond to it (now or later); Can be used for asynchronous processing
  - Allows finer grain of control over what is being transmitted
  - Expose and exploit lower-level SOAP APIs to build the SOAP message and send it
- RPC
  - Simple Request-Response protocol
  - Send a request and expect an immediate response
  - Lower-level SOAP methods build and send messages
  - Hide the messaging nature of SOAP

Think of these as styles of interactions. As you will learn later, there is an explicit style used in the description called document or RPC. But when document is used as the style it does not mean the messages are being used in a messaging oriented fashion – most often a remote procedure is still being called.



At the application level, SOAP message exchange occurs between the Initial Sender of a message and the Ultimate Receiver.

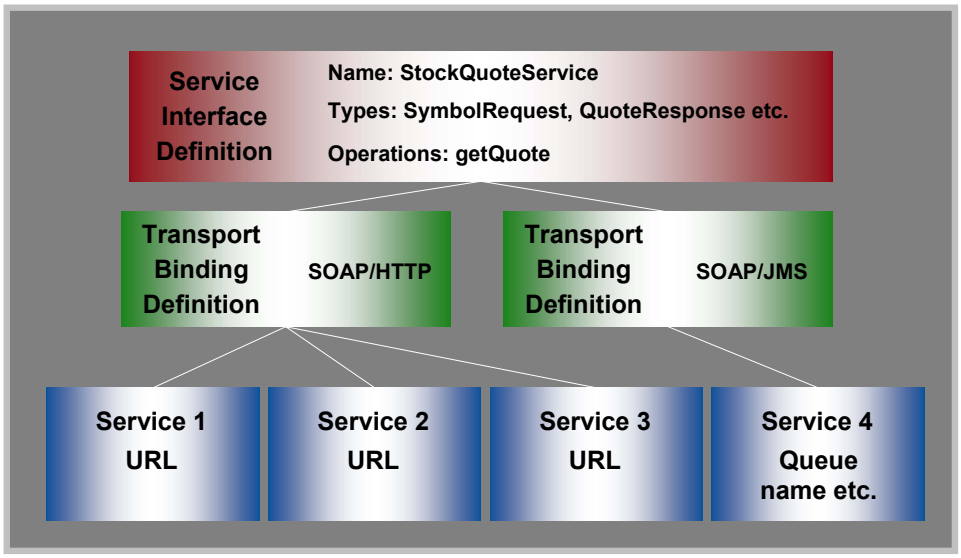
However, the SOAP Processing model allows for the traversal of 'intermediaries'; SOAP Nodes that handle QoS, infrastructure and other messaging related operations that are not application specific. Common intermediation operations include: message logging, message routing, message authentication/authorization, etc.

Intermediaries may be deployed at the Requestor/Provider locations or on SOAP Engines somewhere in between. They are often deployed for service management/QoS reasons. In general, intermediaries should not alter the meaning of the message Body or influence the business semantics of the message.

SOAP Standardization

- SOAP 1.1 is a W3C standard
- W3C released SOAP version 1.2
  - Recommendation, December 19, 2002
  - Some changes from SOAP 1.1, not substantial
- Industry-specific SOAP messages will start a new round of vocabulary standards work

How the Pieces Fit



The decomposition used in WSDL supports the reuse of description information. Here we see one interface definition bound to 2 different transports, one of which is implemented by 3 providers.

WSDL Service Interface – type definitions

```
<?xml version="1.0"?>
<definitions name="StockQuoteService-interface"
  ...
 <wsdl:types>
  <schema targetNamespace="http://www.getquote.com/StockQuoteService"
...
   <element name="symbol" nillable="true" type="xsd:string"/>
   <element name="getQuoteReturn" nillable="true" type="xsd:string"/>
  </schema>
 </wsdl:types>

<wsdl:message name="getQuoteRequest">
      <wsdl:part name="symbol" type="xsd:string"/>
 </wsdl:message>

 <wsdl:message name="getQuoteResponse">
     <wsdl:part name="getQuoteReturn" type="xsd:string"/>
   </wsdl:message>
...
```


Will not be writing from scratch. Will generate from tools.

But we do believe, developers will edit WSDL to get it exactly how they want it.

As the public interface to the service you want it to a complete, precise description of the interface.

WSDL Service Interface - operations

```
...
  <wsdl:portType name="StockQuoteService">
    <wsdl:operation name="getQuote" parameterOrder="symbol">
      <wsdl:input message="intf:getQuoteRequest" name="getQuoteRequest"/>
      <wsdl:output message="intf:getQuoteResponse" name="getQuoteResponse"/>
    </wsdl:operation>
  </wsdl:portType>

  ...
</definitions>
```

```
public interface StockQuoteService {
   String getQuote(String symbol);
}
```


A PortType represents the functional definition of a web service free from QoS and transport concerns; it is analogous to a Java interface.

WSDL Service Binding

```
<?xml version="1.0"?>
<definitions name="StockQuoteService-soapbinding"
  ...
  <import namespace="http://www.getquote.com/StockQuoteService-interface"
    location="http://localhost:80/services/sqs-interface.wsdl"/>

  <wsdl:binding name="StockQuoteServiceSoapBinding"
    type="intf:StockQuoteService">
    <wsdlsoap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="getQuote">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getQuoteRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getQuoteResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>

</definitions>
```


A binding definition identifies a transport and the manner in which messages are carried on that transport… marshalling rules etc. The binding shown here declares that messages will be sent as Document/Literal. The body is expected to be a well-formed XML document and will not undergo any marshalling/de-marshalling to convert the payload to/from the wire format.

WSDL Service Implementation

```
<?xml version="1.0"?>
<definitions name="StockQuoteService"
  targetNamespace="http://www.getquote.com/StockQuoteService"
  xmlns:binding="http://www.getquote.com/StockQuoteService-soapbinding"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

  <import namespace="http://www.getquote.com/StockQuoteService-soapbinding"
          location="http://localhost:80/services/sqs-soapbinding.wsdl"/>

  <service name="StockQuoteService">
    <documentation>Stock Quote Service</documentation>

    <port name="localhost" binding="binding:StockQuoteServiceBinding">
      <soap:address
        location="http://localhost:8080/soap/servlet/rpcrouter"/>
    </port>
    . . .
  </service>
</definitions>
```


This service definition declares that an implementation of the StockQuoteServiceBinding binding has been deployed with the endpoint http://localhost:8080/soap/servlet/rpcrouter.

WSDL Standards

- WSDL V1.1 Specification
  - WSDL is the convergence of separate Web Services IDL definition efforts of IBM and Microsoft
  - WSDL was submitted to the W3C in February 2001 with a total of 25 co-submitters.
  - http://www.w3c.org/TR/wsdl
- WSDL V 2.0 Specification (was 1.2)
  - Latest version is Working Draft, published 3/26/2004
  - http://www.w3.org/TR/wsdl20/