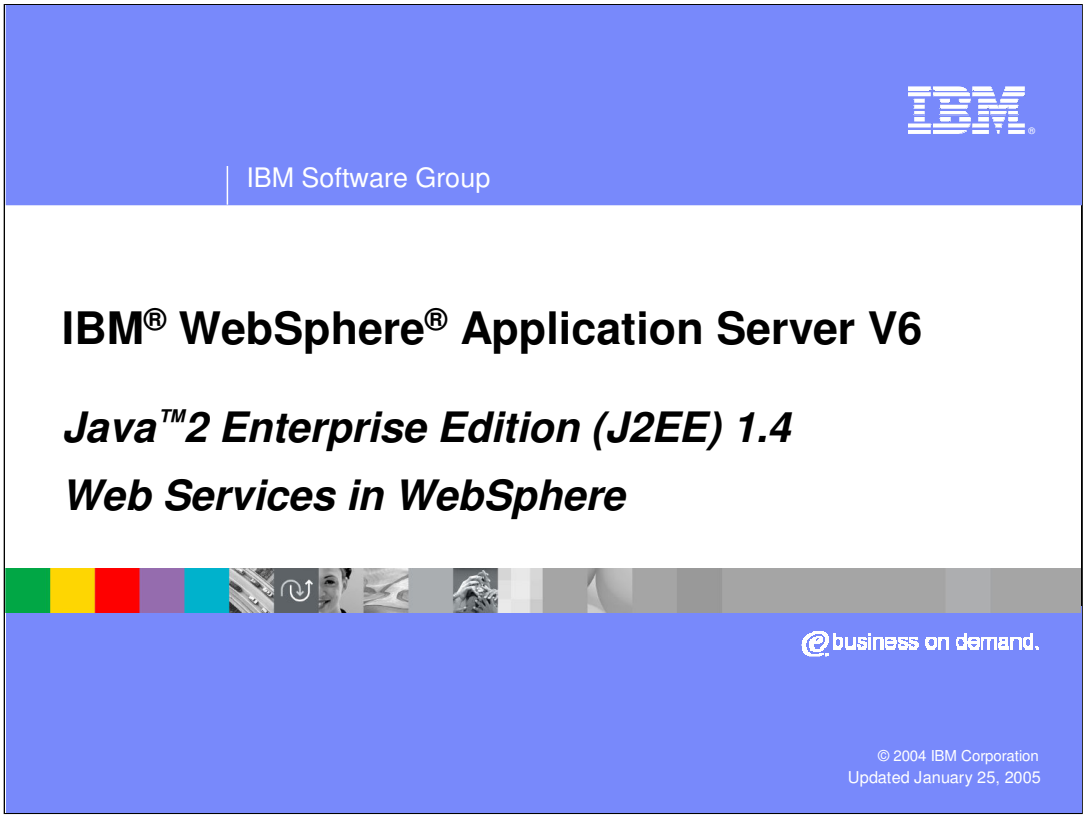


This presentation will focus on the specifics of how Web Services are supported within WebSphere Application Server V6.

Goals

- Provide an overview of
  - Web Services support in WebSphere Application Server V6
  - Migrating existing Web Services to V6
- Prerequisite:
  - Basic understanding of Web Services concept and terminology (like SOAP, WSDL, etc.)
  - Basic understanding of J2EE 1.4 Web Services standards
- What is covered in other presentations
  - New V6 Enhancements
  - WS-Security in WebSphere Application Server
  - Web Services support in IBM® Rational® Application Developer
  - Web Services Gateway Details
  - UDDI registry
  - JAXR

The goal of this lecture is to explain the support for J2EE Web Services offered by WebSphere Application Server V6. Though, not all of the information contained in this lecture is new to V6. Various Web Services Features have been supported as far back as V4.x. There is also another lecture that covers Web Services enhancements in WebSphere V6. These enhancements are outside the J2EE specifications and will not be covered in this lecture.

Other lectures will also go into greater detail on Web Services support in IBM Rational Application Developer, The Web Services Gateway, and UDDI registry.

Agenda

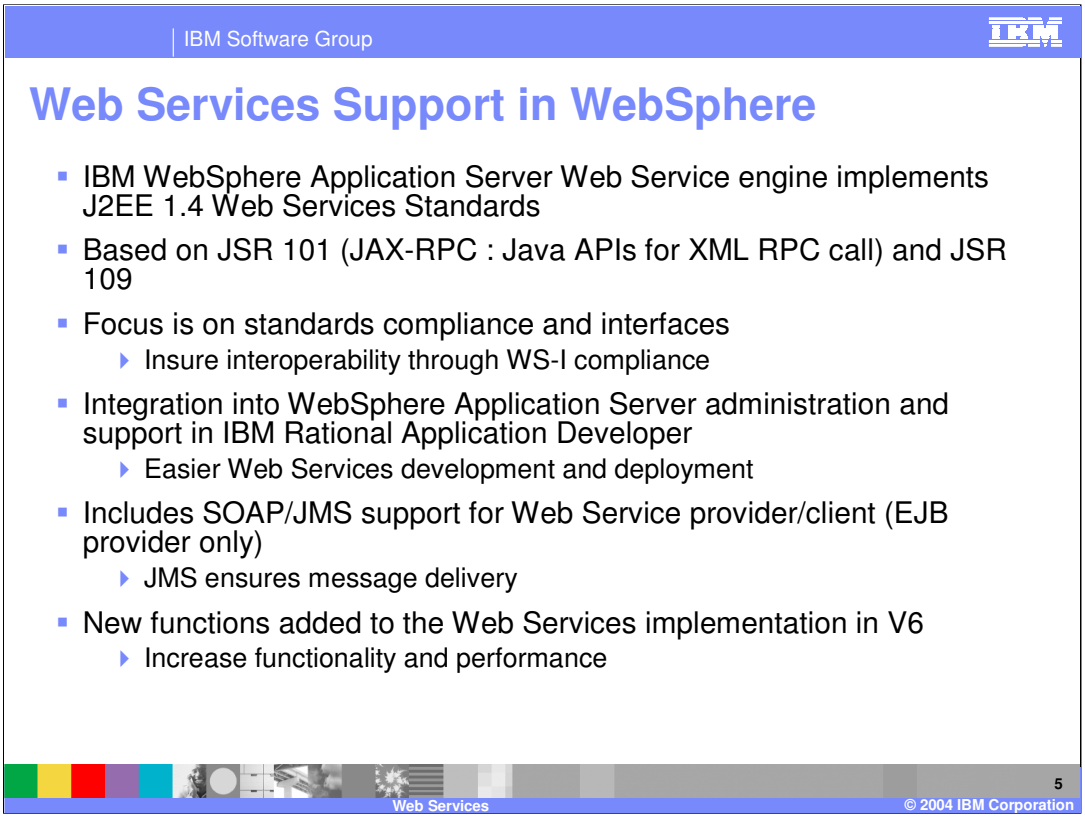
- Web Services Support in WebSphere Application Server
  - JMS Support
  - Web Services Gateway
- Migration of Web Service applications from V5

This lecture will start by discussing the Web Services support in WebSphere Application Server V6, most of these features are not new to WebSphere version 6. It will then go into detail on the support for Java Messaging service or JMS. It will then briefly discuss migrating to V6, detailing some of the changes that have occurred in J2EE.

Section

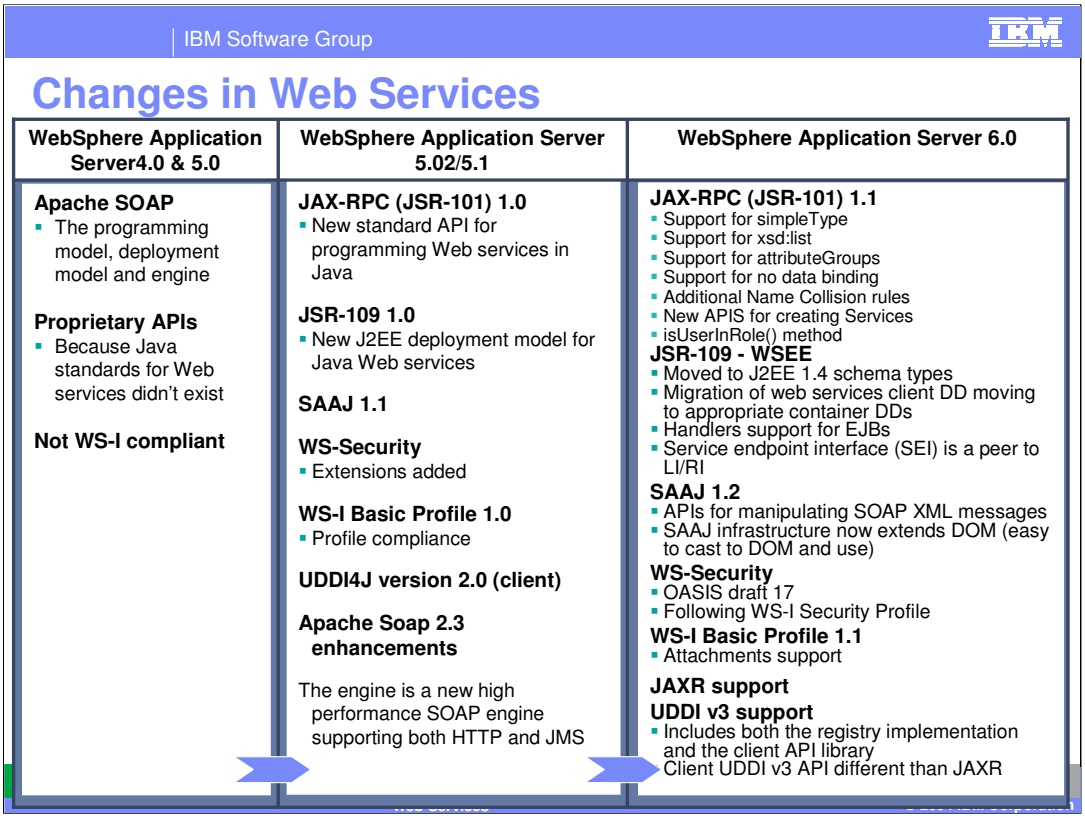
Web Services in WebSphere Application Server

Now on to the Web Services support in WebSphere.

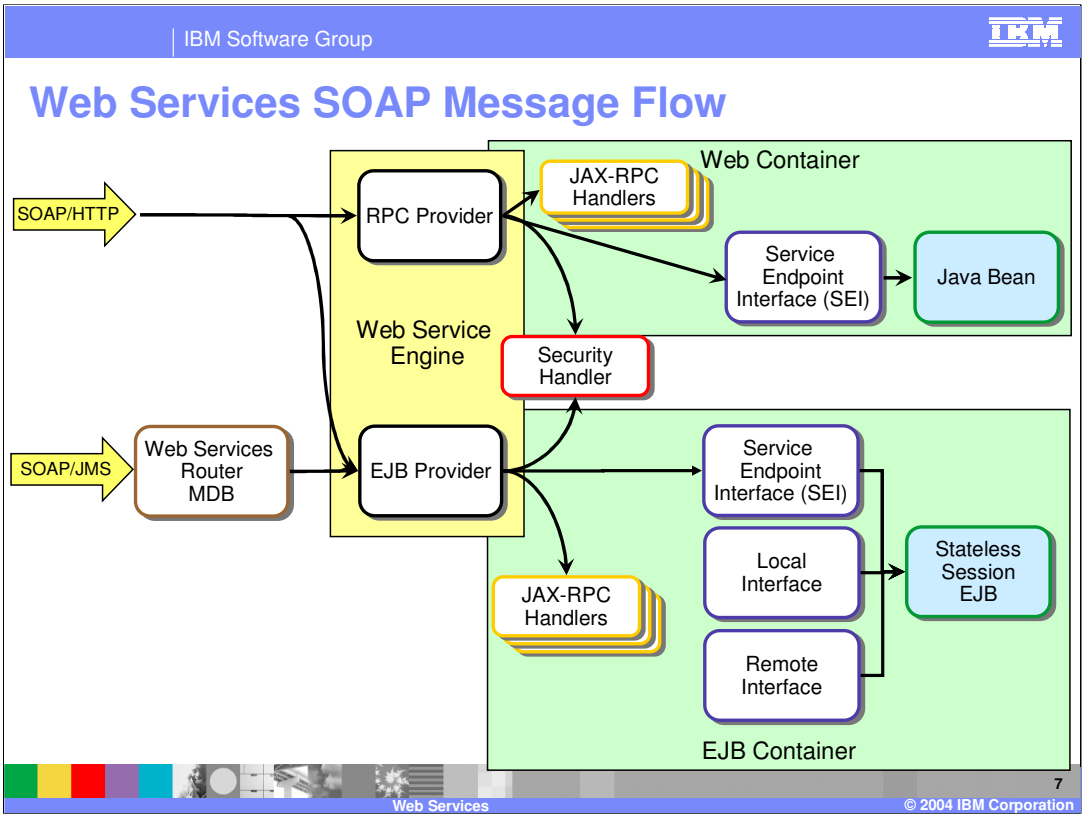


The WebSphere Application Server version 6 contains a web services engine which supports the J2EE 1.4 standards for applications. The web services engine was designed to support both JSR 101, or JAX-RPC and JSR 109, or J2EE web services. These specifications were covered in greater detail in the Web Services overview lecture. By adhering so closely to these standards, the web services engine focuses on interoperability and providing interfaces to internal functions in the runtime.

The web services engine is SAX based, for increased performance. Prior to version 5.02 the engine was based on Apache AXIS technology. To ease development and administration Web Services are tightly integrated into IBM Rational Application Developer and application server support. WebSphere provides support to integrate Web Services with JMS to ensure message delivery. There are a number of other functions and enhancements that have been made in V6 that will be covered in greater detail in another lecture.



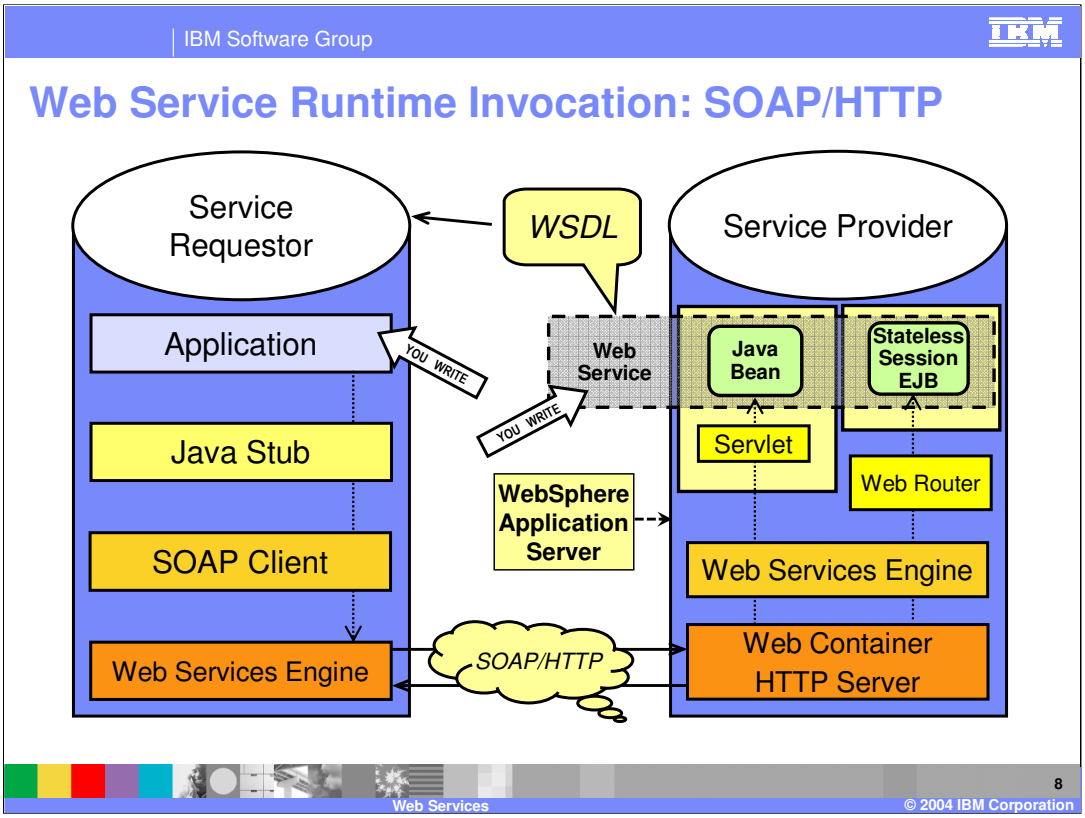
This table shows the continued support for Web Services in WebSphere Application Server since V4.0. As various specifications have moved through the Java community process to be approved and added to the J2EE specification they have been added to WebSphere Application Server. This slide details the support added in V6 for the Web Service Interoperability 1.1 profile, as well as support for the Java API for XML based Registries. There is also continued support for the other standards as they mature, JSR 101 1.1 SAAJ 1.2 and WS-Security.



This slide shows how a SOAP message being sent to a Web Service, enters the J2EE runtime that's part of the Application Server. There are two separate flows here, depending on the transport protocol for the SOAP message, either HTTP or JMS. WebSphere Application Server version 6 supports both.

For a SOAP message being sent via HTTP the web services Engine is fronted by a special Router servlet, that will forward web services calls on to the appropriate provider within the web services engine. Within the web services engine the XML of the SOAP message will be deserialized into Java objects so that calls can be made to the appropriate Service Endpoint Interface for the target service being called. Before the call to the SEI is made, any appropriate JAX-RPC handlers will be executed on the message, one of these being the WS Security handler. For SOAP over HTTP calls, the target SEI and handlers will operate within the Web Container of the J2EE environment, where the SEI will forward the call onto a target bean.

With a SOAP message arriving over a configured Java Messaging Service, things work a little differently. A Message Driven Bean router will receive the SOAP message and forward it onto the EJB provider in the Web Services engine. SOAP over JMS only supports EJB web services, so the RPC provider in the Web Services engine will not be involved. The EJB provider will then forward the message on, through the appropriate JAX-RPC handlers, to the target SEI for the EJB. Both the EJB and handlers will operate within the EJB container.



This slide shows a deeper look at the how information flows between various components during a Web Service invocation. On the Service requestor the application has to be created to act as a client to the target Web Service. The Java Stub and SOAP client will be created for the client by the developer. On the Service Provider side, the Bean representing the Web Service has to be created. This will communicate with the Web Services engine through a servlet running in the Web Container. The Web Services engine will receive the messages from the HTTP server, and handle invoking the target service.

Section

Support for Java Messaging Service (JMS)

Next will be more details on the support WebSphere Application Server provides for using Java Messaging Service as your transport protocol for a Web Service.

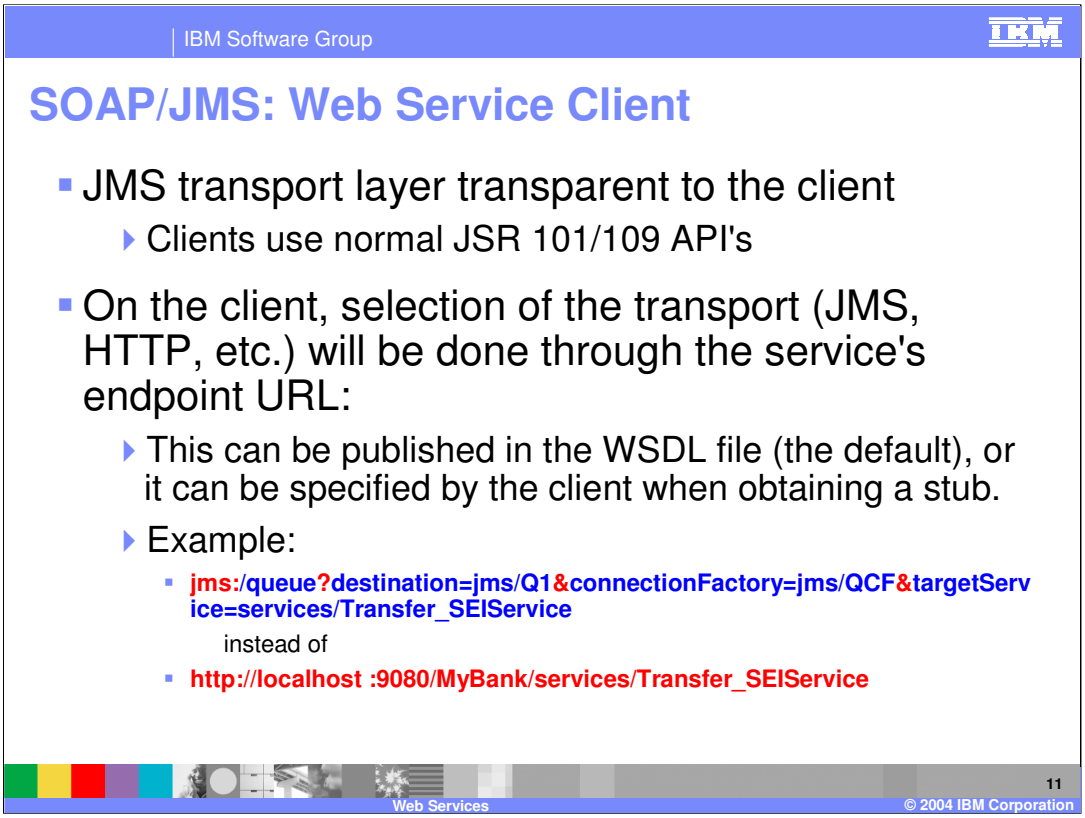
SOAP/JMS Support

- JAX-RPC has been extended by IBM to allow SOAP messages to be sent by a messaging protocol (JMS)
  - Queuing the messages ensures they will be delivered
- Only EJB Web Service supported for JMS transport
- WSDL service definition

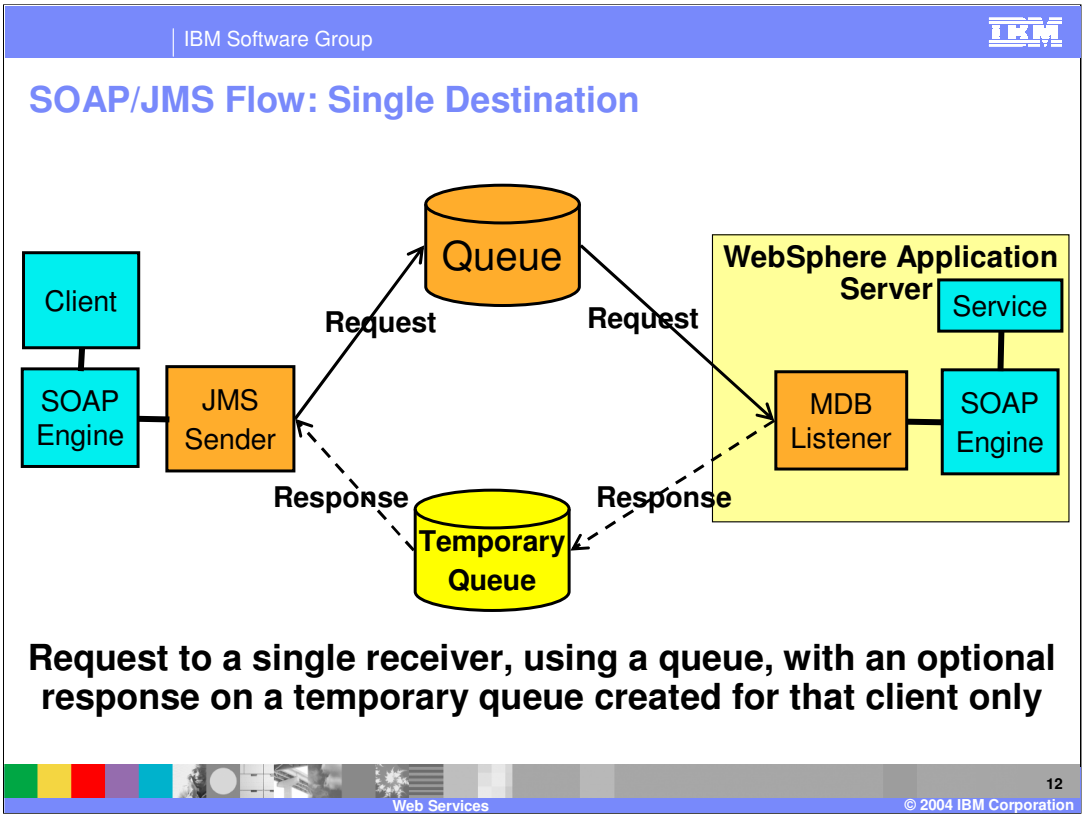
```
<wsdl:service name="Transfer_SEIService">
 <wsdl:port name="MyBank"  binding="impl:MyBankSoapBinding">
  <wsdlsoap:address
   location=" jms:/queue?destination=MyBankQueue... "/>
 </wsdl:port>
</wsdl:service>
```


The support for SOAP via JMS in WebSphere Application Server, is actually an IBM extension of the JAX-RPC standard, allowing SOAP messages to be sent via a messaging protocol. Using a messaging protocol to send a message ensures delivery of the SOAP message for a business critical process. HTTP does not ensure a message will be received, the most common problem is that the HTTP server can be down when the message is sent. This would result in a lost message. Using a messaging protocol to place a message on a queue avoids these problems.

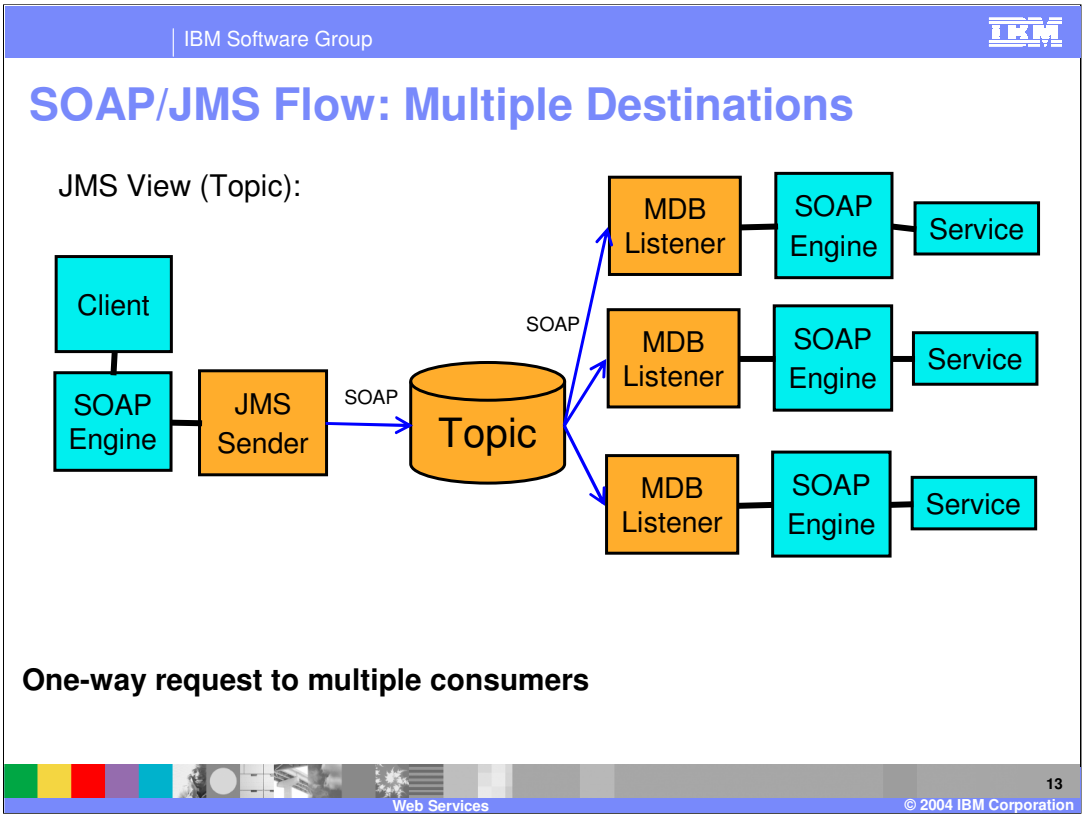
JMS is supported only for Enterprise Java Bean services, so this enhancement can be restricting to a design. Using JMS changes the target address in the WSDL description as you see here on the slide. Instead of an internet address, there is a target queue.



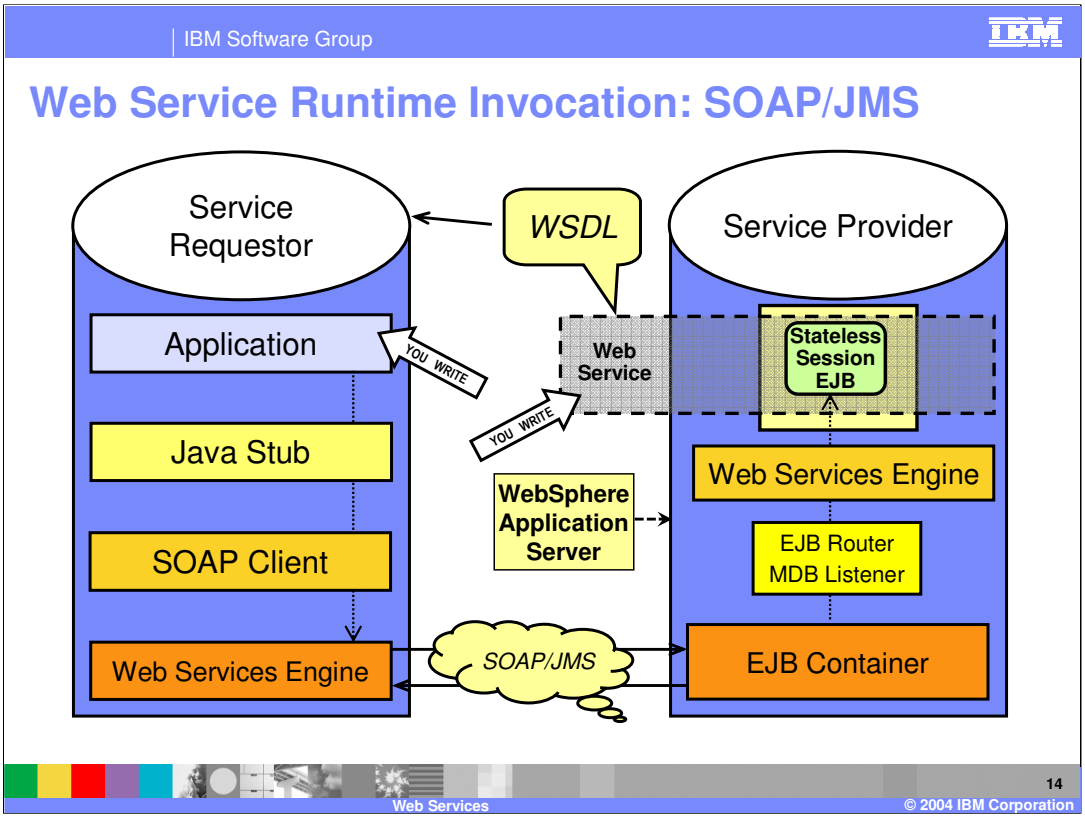
Using JMS as the transport layer, instead of HTTP, requires no changes within the Web Services client. Clients using JMS still use the standard JSR 101 and 109 APIs. The selection of the transport protocol occurs through the target services endpoint URL. This can be done both statically if the address is published in the WSDL file or dynamically by specifying it when obtaining a stub. Listed here are two examples of the same service exposed over the different protocols. The top example shows the address for a JMS call, the bottom shows a traditional HTTP call.



Here is an example flow of a call being made using JMS to a single Web Service destination. The Web Service engine will use a JMS sender to send the SOAP message to a target queue or topic. A queue must be used if a result is needed, a topic can be used if no result will be returned. A Message Driven Bean Listener will consume the message from the queue and send the soap message to the Web Services SOAP engine on the provider side. If a result is sent a temporary queue will be created for the result message. The client will block until it receives the response.



There is also support with WebSphere Application Server to have multiple Web Services consuming messages from the same topic. Because a topic must be used for this scenario, no results can be returned to the client. Each separate Web Service will have its own message driven bean to consume messages from the same topic in this example. The primary use for this scenario is to improve performance by using multiple Web Services.



This slide shows the differences in a call to a Web Service using JMS. The primary difference is that instead of using an HTTP client to send the message, a JMS client is used. And on the provider side, the message is received by a message driven bean running within the EJB container, rather than by a servlet running within the Web container.

SOAP/JMS: Web Service Provider Steps

- Developer
  - Use -bindingTypes jms option for Java2WSDL
    - Creates a JMS binding in the resulting WSDL document
  - Use -transport jms option for EndptEnabler
    - This will create a MDB for each EJB Web Service module
- Deployer
  - Configure the JMS resources for the MDB to listen to the specific Queue/Topic
  - Install the application and map the MDB to the appropriate listener port
  - Publish the WSDL for the client

This slide details the differences for a Developer and Deployer when creating a Web Service using JMS. The developer uses a different binding option for Java2WSDL, and also uses a separate option when creating the endpoint. If you are using IBM Rational Application Developer, these can be enabled from the preferences section when creating a new Web Service. The Deployer must create and configure the JMS resources for the Web Service's MDB to listen to the appropriate queue or topic. Also when installing the application the Deployer must map the MDB to the appropriate listener port.

Section

Web Services Gateway

Now for information on the Web Services Gateway.

WSGW Overview

- Known as the Service Integration Bus Web Services Enablement (SIBWS) in V6
- Provides integrated support for Web Services communications using the Service Integration Bus
- Separate installation from the Application Server install
  - Available under <install_root>/installableApps
  - Detailed install instructions available in Information Center

In WebSphere Application Server V6 the Web Services Gateway has been renamed to the Service Integration Bus Web Services Enablement. This presentation will refer to it as the gateway. The gateway provides support for Web Services using the new Messaging technology in V6, in particular it integrates with the Service Integration Bus. The gateway is a separate installation from WebSphere Application Server, as it has been in previous releases. In V6 the user interface for the gateway is provided within the Administrative console after it has been installed, instead of as a separate program.

SIBWS Benefits

- Take an internal service that is available at a service destination, and make it available as a Web Service
- Take an external Web Service, and make it available at a service destination
- Map an existing service – internal or external - to a new Web Service that appears to be provided by the gateway
- Change the transport protocol used by the Web Service at the gateway
- Enable WS-Security on services at the gateway

The gateway provides a number of enhancements for Web Services in a WebSphere Application Server topology. It allows an administrator to map an existing Web Service to one provided by the gateway. At the Gateway security can be altered or added to a Web Service or the transport protocol can be changed. As part of the gateway's integration with the Service Integration Bus, a service available on the Bus as a service destination can be made available as a Web Service using the gateway. An external service can also be added as a service destination to the Service Integration Bus using the gateway.

Section

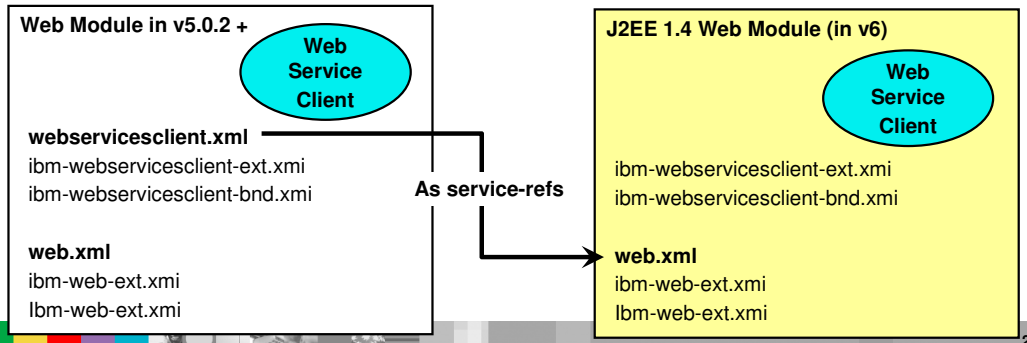
Migration from V5 J2EE 1.3 Web Services Application

Now for details on migration issues concerning existing Web Services applications.

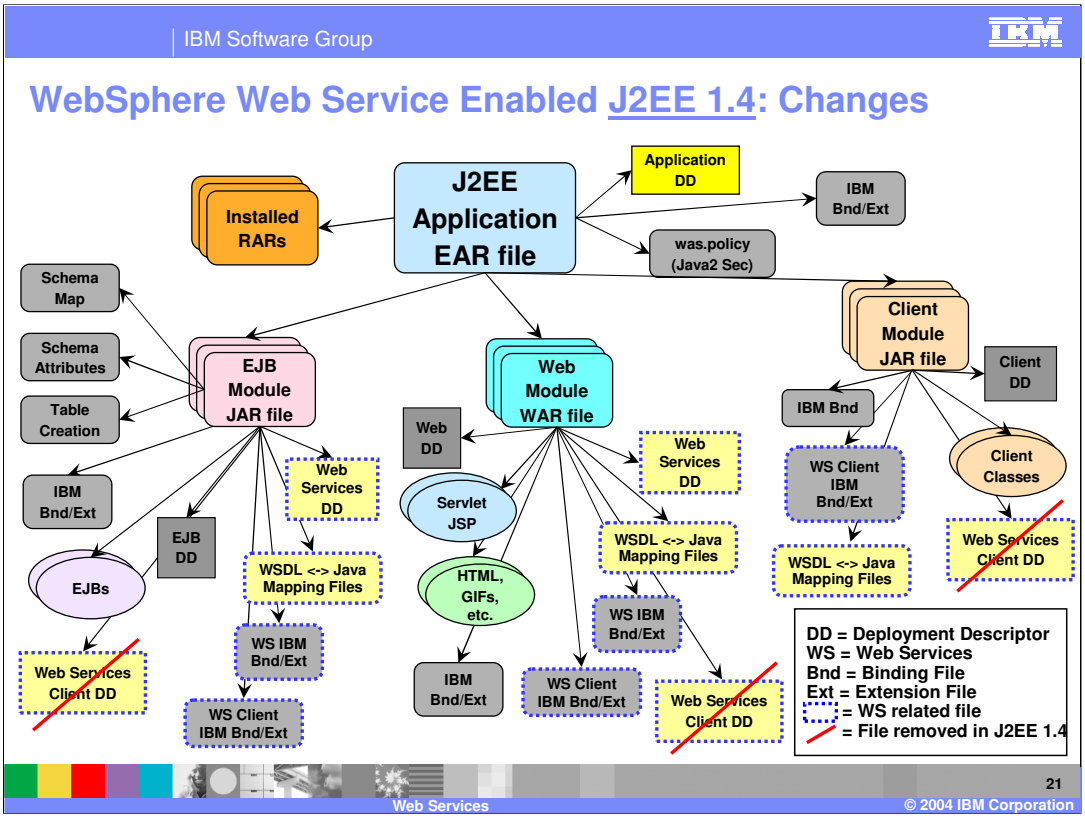
J2EE 1.4 Web Services changes

- **J2EE 1.4 Web Services deployment descriptors defined by XSD**
  - XSD replaces DTD's for J2EE Web services deployment descriptors definition
- **Changed to Web Services defined in J2EE application deployment descriptors**
  - Eliminate **webservicesclient.xml**
  - References to Web Services are contained in the **web.xml**, **ejb-jar.xml**, and **application-client.xml** files as **service-refs**
  - No changes to IBM specific Web Services client bindings or extensions files

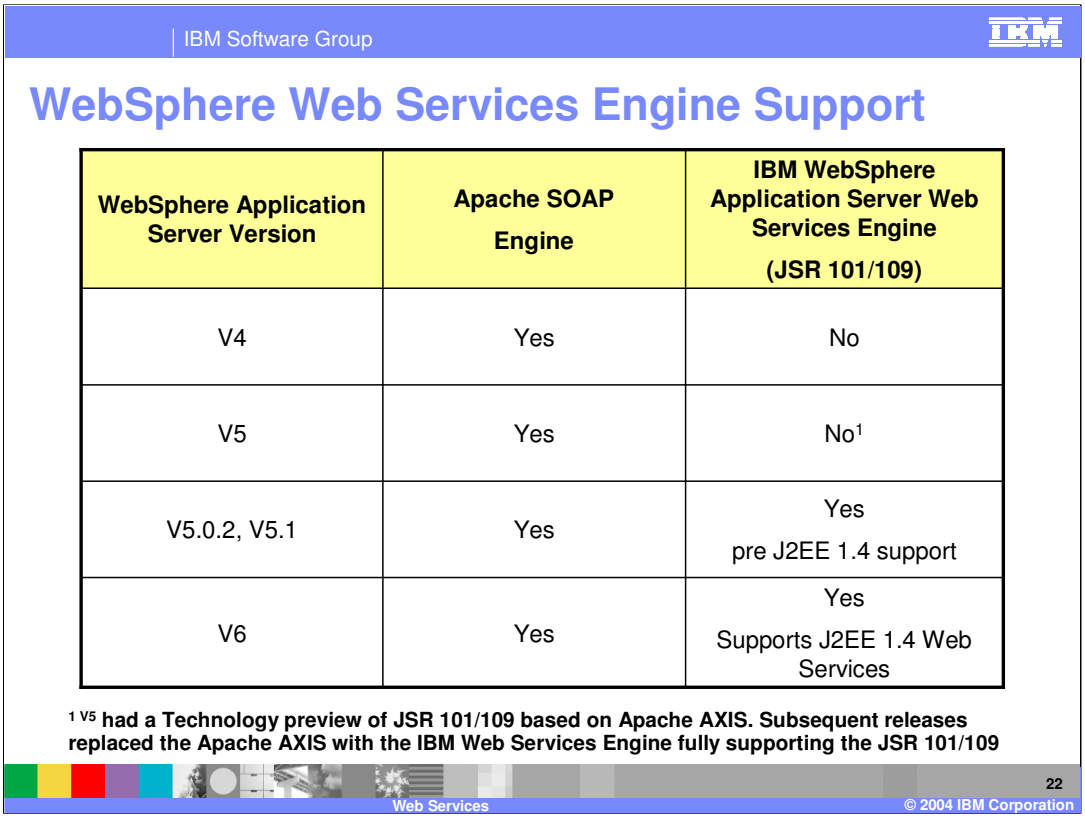
EXAMPLE, Web Service Client in Web module:



This slide describes the J2EE 1.4 changes, primarily the deployment descriptors have been changed to be defined by XSD. The deployment descriptors associated with Web Services clients have changed, this is due to the inclusion of Web Services in the 1.4 specification. Information that was once stored in the Web Services client.xml file have been moved to the web.xml, ejb jar.xml and client application.xml files. There haven't been any changes to the IBM specific binding and extension files in this release. Security Bindings have also changed due to the maturation of the Security specification, which will be detailed in the Security presentation.



This slide is a visual example of the deployment descriptor changes. Showing the removal of the Web Services client.xml file. The rest of the files associated with Web Services have not been changed.



This slide details the support in WebSphere Application Server for the Apache SOAP engine. Starting in V4, there has been support for services and that continues into V6.

This slide also shows the historical support for the J2EE web service standards. IBM included early support for JSR 101 and 109 back in V5.0.2; now those Java Specification Requests (JSR) are included in the J2EE specifications.

Summary

- WebSphere Application Server provides support for J2EE Web Services
- Support for Web Services in WebSphere Application Server is extended in a number of ways
  - JMS support
  - Service Integration Bus Web Services Enablement

This presentation covered the support offered for Web Services in WebSphere Application Server V6. Separate presentations will cover other Web Services details in greater depth. Following this slide are a number of resources to find more information on Web Services and WebSphere Application Server.

Resources: JSR 101 and 109

- JSR 101 (JAX-RPC)
  - http://java.sun.com/xml/jaxrpc/index.html
- JSR 109
  - http://jcp.org/jsr/detail/109.jsp
  - http://www-106.ibm.com/developerworks/webservices/library/ws-jsr109/index.pdf
- Introduction to Web Services
  - http://java.sun.com/webservices/docs/ea2/tutorial/doc/IntroWS.html
- WS-I Basic Profile
  - http://www-106.ibm.com/developerworks/webservices/library/ws-basicprof.html?dwzone=webservices

Resources: Standards and Organizations

- http://www.w3.org/TR/SOAP/
- http://www.w3.org/TR/wsdl
- http://www.WS-I.org
- http://xml.apache.org/soap

Resources

- http://www.ibm.com/software/ad/studioappdev
- http://www.ibm.com/software/webservices
- http://www.ibm.com/developerworks/webservices
- http://www.alphaworks.ibm.com/webservices
- http://www.redbooks.ibm.com
  - SG246891 - WebSphere V5 Web Services Handbook
- http://www.eclipse.org

Trademarks, Copyrights, and Disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	CICS	IMS	MQSeries	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.