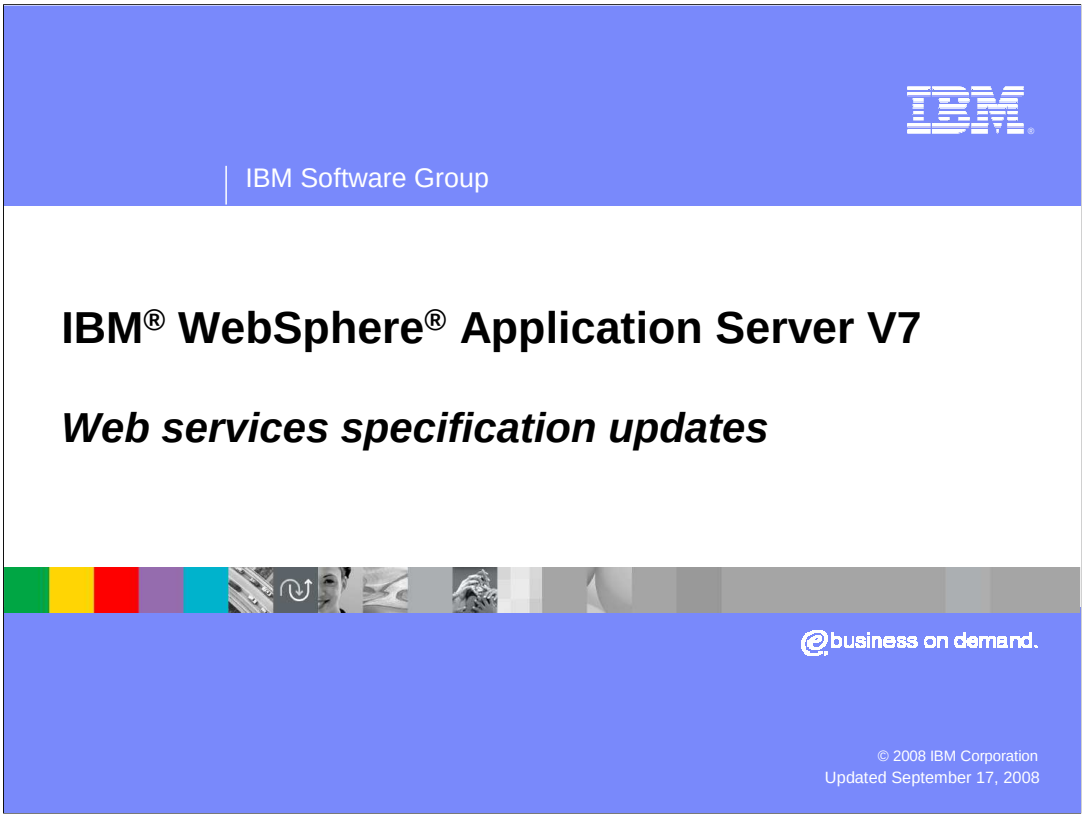


This presentation discusses numerous updates to Web services specifications and programming models provided by WebSphere Application Server version 7.

Agenda

- Feature Pack for Web Services integration
- WS-Addressing
- JAX-WS 2.1
- JAX-B 2.1
- Support for JSR 109 and JAX-WS
- WS-Security enhancements
- Other features
- MTOM updates

This presentation will begin by discussing features that have been integrated in WebSphere Application Server version 7 from the feature pack for Web services. Next it will explain updates to that content across various areas, including; Web services addressing, support for the Java API for XML based Web services version 2.1, the Java API for XML binding version 2.1, new support for using the JAX-WS programming model with Web services in the J2EE specification, enhancements to Web services security and other features.

Section

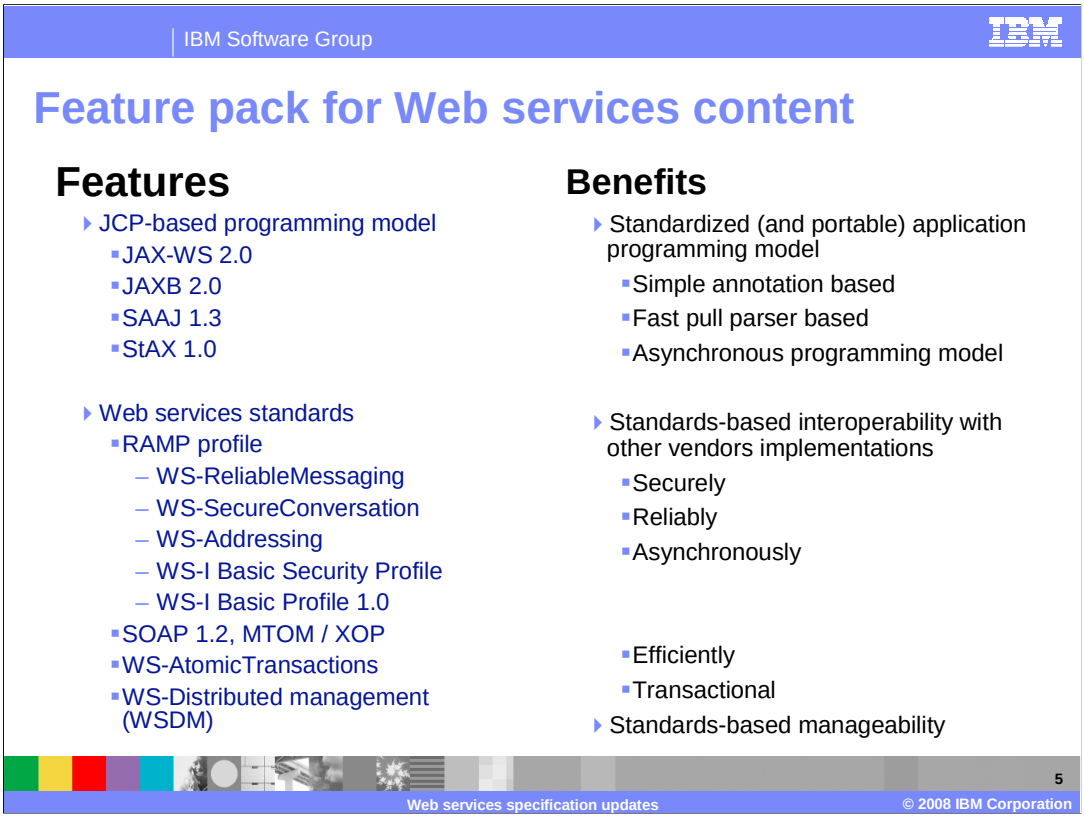
Feature Pack for Web Services integration

The next section will explain the integration of features from the feature pack for Web services.

Integration

- WebSphere Application Server V7 integrates the features provided by the Feature Pack for Web Services
- Adds new features and updates as well

WebSphere Application Server Version 7 integrates the content provided by the IBM WebSphere Application Server Version 6.1 Feature Pack for Web Services. This includes the Apache Axis 2 runtime and the Java API for XML based Web services programming model. More information about the specific content introduced with the feature pack for Web services can be found in the IBM Education Assistant topics for that product. WebSphere Application Server Version 7 than adds new features that this presentation will go into greater detail on.



The feature pack for Web services provided a new Java community process based programming model for the Java API for XML based Web services. This included additional specifications for the Java API for XML Binding, and updates to the SOAP API for Attachments in Java and the SOAP specifications. This programming model provided an updated, standardized way to develop Web service based applications. It included annotations support, and an asynchronous messaging model. The feature pack for Web services also provided support for other Web services standards, such as the Reliable Asynchronous Messaging Profile or RAMP. This included standards for various qualities of server, such as reliable message, secure conversation and Web services addressing. The feature pack also provided support for a new SOAP Message Transmission Optimization Mechanism or MTOM. This standard describes a faster method for dealing with large binary attachments in Web services.

Section

WS-Addressing

The next section will explain some minor updates to Web services addressing.

WS-Addressing

- WebSphere Application Server V7 adds support for the WS-Addressing metadata specification
  - Allows for simplified, interoperable client configuration using WS-Policy
- Replaces the WS-Addressing WSDL binding specification
  - Which is now deprecated in WebSphere Application Server V7

WebSphere Application Server version 7 adds support for the WS-Addressing meta data specification, this replaces the WS-Addressing WSDL binding specification so that is now deprecated. This allows for simplified interoperable client configuration using WS-Policy.

Section

JAX-WS 2.1

The next section explains the updates to the JAX-WS specification in version 2.1.

JAX-WS 2.1 overview

- JAX-WS 2.1 API has integrated support for WS-Addressing
  - Developers can use the standardized API to create and use endpoint references to target a Web service endpoint
- Introduces a new configuration model for JAX-WS services using “Features”
  - Qualities of service specified in JAX-WS code
  - AddressingFeature, MTOMFeature, RespectBindingFeature
- Requires the use of JAXB 2.1

The Java API for XML based Web services (JAX-WS) version 2.1 introduces several updates to the specification. First, the API has been updated to now include support for WS-addressing, this allows developers to use the standardized API to create and use endpoint references to target a specific Web service endpoint. The updates also introduce an additional way to configure qualities of services for JAX-WS based services. Features can be used to specify addressing, bindings and MTOM capabilities of a service. These same qualities of service can also be identified using policy sets or annotations, both of which will be override use of features. This specification also uses the Java API for XML Bindings version 2.1 (JAX-B).

Section

JAX-WS 2.1 API

The next section will explain the JAX-WS version 2.1 API in more detail.

JAX-WS 2.1

- New classes to support the JAX-WS 2.1 API
  - Used to represent endpoint references
- JAX-WS 2.1 also introduces new annotations to support the mapping of actions to WSDL operations
  - javax.xml.ws.Action and javax.xml.ws.FaultAction

JAX-WS version 2.1 introduces new classes that are used to represent endpoint references. It also introduces new annotations that allow developers to map actions to WSDL operations.

New annotations

- The Action annotation allows explicit association of a WSAddressing action message addressing property
  - Associate with input, output, and fault messages for the mapped WSDL operation
- The FaultAction annotation is used inside an Action annotation
  - Allows an explicit association of a WS-Addressing Action message addressing property with the fault messages of the WSDL operation, using an association with the exception class

The action annotations can be used to associate a WS-Addressing action message addressing property with the input, output or fault messages for the WSDL operation. The JAX-WS 2.1 specification does not require that the action mapping in the annotations be reflected in the WSDL for a deployed Web service endpoint. The FaultAction annotation is used within the Action annotation; it allows a developer to associate a message addressing property with the fault messages of a WSDL operation.

Co-existence

- The Feature Pack for Web Services shipped a proprietary WS-Addressing API/SPI for use with JAX-WS
  - Continues to be available, and is not deprecated
  - Utility methods are provided to convert between the two
- The JAX-WS 2.1 API allows limited control over the WS-Addressing headers
  - Only the wsa:To header can be configured directly
- For more direct control over the other WS-Addressing headers, then the proprietary WS-Addressing API/SPI must be used
- Developers should try to consistently use one API or the other for the best results

The feature pack for Web services shipped a proprietary WS-Addressing API for use with the JAX-WS programming model. This continues to be available in WebSphere Application Server version 7, with utility methods provided to convert between the two models. The JAX-WS version 2.1 API is limited in it ability to control the WS-Addressing headers, with only the ability to directly configure the To header. The proprietary API should be used for more direct control over the WS-Addressing headers, though developers should try to consistently use a single API so as to not run into problems.

Section

JAX-WS 2.1 "Features"

The next section further explains the JAX-WS version 2.1 introduction of features.

Features

- A feature is a way to programmatically control certain functions or behaviors
- Features are derived from the class javax.xml.ws.WebServiceFeature
  - Allows a client application developer to pass different types of WebServiceFeatures to other APIs
- Each feature also has a corresponding annotation
  - Inserted in the SEI or implementation bean to control the relevant behavior on the server-side

Specific to the JAX-WS version 2.1 specification, a feature is a new way to programmatically control certain functions or behaviors of a service. All of the features supported by the specification are derived from the class javax.xml.ws.WebServiceFeature, this allows a client developer to pass different types of features to other APIs. Each feature has a corresponding annotation, which can be inserted into the service endpoint interface or implementation bean to control the associated function or behavior for the service.

Types of features

- Associated annotation is @Addressing
  - Supports a “Required” property
    - If enabled requests must include WS-Addressing headers
- Control the use of WS-Addressing by JAX-WS, in compliance with the WS-Addressing 1.0 - Core and SOAP Binding specifications
- Allows a developer to enable/disable the use of WS-Addressing by JAX-WS

The addressing feature controls the use of WS-Addressing by JAX-WS; its associated annotation is @addressing. If this is enabled requests made to this service must include WS-Addressing headers. On both the client and server-sides the addressing feature can only be used with a SOAP 1.1 over HTTP or SOAP 1.2 over HTTP binding. If it is used with another binding, such as XML over HTTP, then this will fail. On the client-side an exception will be thrown, and on the server-side the Web service will fail to deploy.

AddressingFeature

- AddressingFeature
  - Associated annotation is @Addressing
- SubmissionAddressingFeature
  - Associated annotation is @SubmissionAddressing
- MTOMFeature
  - Associated annotation is @MTOM
- RespectBindingFeature
  - Associated annotation is @RespectBinding

The submission addressing feature allows a developer to control the use WS-Addressing specific to compliance with the WS-Addressing Member Submission specification. The MTOM feature is used by a developer to specify whether binary content in the body of a SOAP message is sent using MTOM. If it is enabled, binary content will be sent in the MTOM attachment form, if disabled, binary content will be sent as a base-64 encoded string. It also Includes a “Threshold” property that indicates to the runtime that binary data larger than the threshold should be sent using MTOM. The respect binding feature controls whether the JAX-WS implementation inspects the wsdl:binding for an endpoint at runtime. This insures that the parameter and return data complies with the binding type. It also checks that all wsdl:extensions that have the required attribute set to “true” are understood and are being used.

Co-existence with policy sets

- JAX-WS 2.1 introduces a new configuration model for Web services based on “features”
  - Potential for conflict with configured policy sets
- If there is a conflict the policy set configuration will override the programmatic configuration of “features”
- The JAX-WS 2.1 specification makes no mention of WS-Policy
  - In WebSphere Application Server V7 WS-Policy settings may also override programmatic “features”

Since features also provide a configuration model for JAX-WS based services, similar to policy sets, there is a potential for conflict. The policy set configuration will override programmatic configuration data stored in features. WS-Policy setting may also override features as well; the JAX-WS specification makes no mention of this other specification at this time.

Section

JAXB 2.1

The next section briefly explains the minor updates in the Java API for XML bindings version 2.1 specification.

JAXB 2.1 enhancements

- Schema generation and compiler tools now have the option to not generate certain artifacts
  - xjc will not generate classes
  - schemagen will not generate schema
- @XMLSeeAlso annotation
  - Used to instruct JAXB to bind specific additional classes
  - Sub-classes might be overlooked without this annotation

```
@XmlSeeAlso({Dog.class, Cat.class})
class Service {
   Animal getAnimal();
}
```


There have been two major enhancements to the JAX-B specification. The first is that the schema generation and compiler tools now have an option to not generate specific artifacts. The compiler tool, xjc, can use an option to not generate class files. The schema generation tool can use this option to not generate the XML schema. The second major change is that a new annotation has been added, that instructs JAX-B to look for a bind additional classes. The @XMLSeeAlso annotation allows for specific sub-classes to be included in the binding operations that previously may have been overlooked.

Section

JSR 109

The next section will talk about enhancements with the JAX-WS programming model supporting JSR 109, or the Web services in the J2EE specification, within WebSphere Application Server version 7.

JAX-WS support for JSR 109

- WebSphere Application Server V7 provides support for JSR 109 V1.2
- JAX-WS services may use the Web services deployment descriptor to define end points
  - Overrides endpoint information specified by annotations
- Support for managed JAX-WS clients
  - Adds client side deployment descriptors and annotations
- JAX-WS services can use the handler model defined in JSR 109

WebSphere Application Server V7 adds support for JSR 109 V1.2, or the Web services in J2EE, to be used along with the JAX-WS programming model. This allows JAX-WS services to use the Web services deployment descriptor to define end points rather than using annotations. Information configured in the deployment descriptor will override the annotations. This also adds support for managed JAX-WS based clients, with the addition of client side deployment descriptors and annotations. It also allows JAX-WS services to use the handler model described by JSR 109.

JAX-WS support for client annotations

- The @WebServiceRef and @Resource annotation can be used to request injection of JAX-WS service (javax.xml.ws.Service) or port instances

```
public class JAXWSJ2EEClient {

  @WebServiceRef(name="service/EchoService", type=EchoService.class)
  // name is used to bind this reference into the JNDI namespace
  // the type represents the javax.xml.ws.Service class
  private static EchoService jaxwsService;
  // the 'jaxwsService' field is injected with an instance of EchoService

  @Resource(name="service/EchoService_resource", type=EchoService.class)
  // same as above
  private static EchoService jaxwsServiceResource
  // the 'jaxwsServiceResource' field is injected with an instance of EchoService

  @WebServiceRef(name="service/EchoPort", value=EchoService.class, type=EchoPort.class)
  // same as above and type indicates port class
  private static EchoPort jaxwsPort
  // the 'jaxwsPort' field is injected with an instance of EchoPort
  ...
```


Two important annotations have been added to support client applications, @WebServiceRef and @Resource can be used to request injection of JAX-WS service and port instances. Examples of these annotations and how to use them are shown here.

Section

WS-Security

The next section will explain some updates to WS-Security in WebSphere Application Server version 7.

WS-Security

- WebSphere Application Server V7 adds WS-Security Kerberos Token Profile support for both JAX-RPC and JAX-WS applications
  - For JAX-RPC applications, the Kerberos token can only be used as an authentication token
  - For JAX-WS applications, the Kerberos token can be used for both authentication and message protection (signing and encrypting)
- Support for LTPAv2 tokens
- May experience interoperability issues between WSFP level services and those developed for WebSphere Application Server V7
  - Bindings may need to be updated

WebSphere Application Server version 7 adds WS-Security Kerberos Token Profile support for both JAX-RPC and JAX-WS based applications. For JAX-RPC applications, the Kerberos token can only be used as an authentication token. With JAX-WS applications, the Kerberos token can be used for both authentication and message protection. Support is also added for LTPA version 2 tokens. Services developed with the feature pack for Web services may need to have their bindings updated to be compatible with WebSphere Application Server version 7.

Section

Other features

The next section describes several updates to Web services development.

JAX-WS enhancements: EJB

- WebSphere Application Server V7 adds the ability to develop JAX-WS services based on enterprise beans
  - Adding the @WebService or @WebServiceProvider annotation defines the bean as a JAX-WS Web service
  - JAX-WS services can optionally use a service endpoint interface (SEI)
- JAX-WS Web service applications containing enterprise beans have certain considerations
  - Must be a stateless session bean
  - Must be packaged in EJB 3.0 or higher modules
  - Must be deployed with the **endptEnabler** command
  - Supported over an HTTP or JMS transport

WebSphere Application Server version 7 adds the capability to develop JAX-WS services based on enterprise beans, this was not available in the feature pack for Web services. By adding the appropriate JAX-WS annotations to the bean implementation, it will be defined as a JAX-WS service. JAX-WS services can optionally use a service endpoint interface as well. This is only possible for stateless session beans, and must be packaged as EJB 3.0 or higher modules. Beans defined as JAX-WS services can use either the HTTP or JMS transports.

JAX-WS enhancements: SOAP/JMS

- WebSphere Application Server V7 introduces support for a new proposed industry standard SOAP over JMS protocol
  - Based on the Worldwide Web Consortium (W3C) SOAP over JMS specification
  - Provides a standard set of interoperability guidelines
- Previous IBM specific implementation of SOAP over JMS protocol has been deprecated
  - May still be used by JAX-RPC and JAX-WS services
  - Must be used if your client invokes enterprise bean-based Web services that are supported by an earlier version of WebSphere Application Server

WebSphere Application Server version 7 introduces support for a new proposed industry standard SOAP over JMS protocol. Based on the Worldwide Web Consortium's SOAP over JMS specification, this provides a standard set of interoperability guidelines. The previous IBM specific implementation of a SOAP over JMS protocol is still available and may be used by both JAX-WS and JAX_RPC applications, it is deprecated in this release though. If there is a client that will invoke a enterprise bean based services supported by an earlier version of WebSphere Application Server, then your client needs to use this earlier version of the protocol.

Section

MTOM

The next section describes some enhancements made to the MTOM support in WebSphere Application Server version 7.

MTOM enhancements

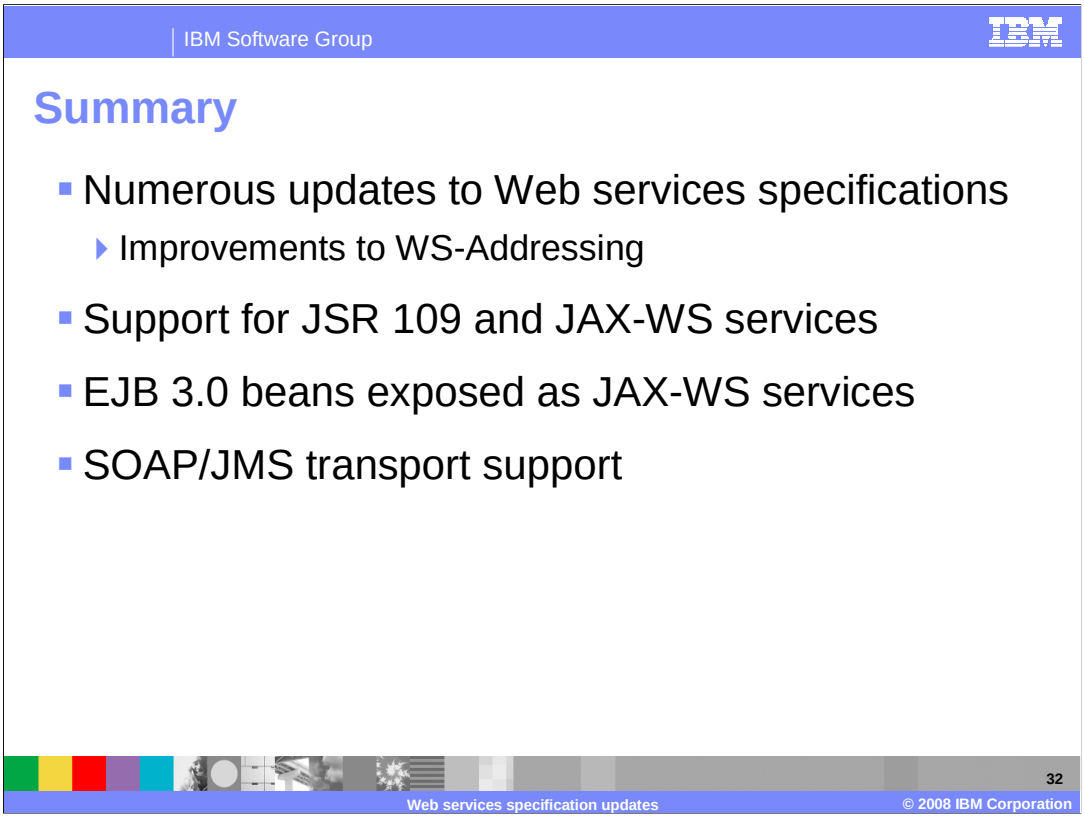
- By default WebSphere Application Server V7 will use HTTP chunking when sending MTOM attachments
  - Allows for greater scalability and improved performance
  - Supports up to 1GB attachments
- Can be disabled with an option in the HTTPTransport policy
- No changes needed to applications

WebSphere Application Server version 7 introduces a new chunking behavior for MTOM attachments. By default MTOM attachments will be sent using HTTP chunking. Attachments are chunked into 32 kilobyte pieces. This size was chosen for the best performance. This change allows for greater scalability and performance when using MTOM, it allows for attachments of up to 1 gigabyte to be sent using MTOM. This feature can be disabled on the HTTP transport policy. No application changes are needed to change the behavior used.

Section

Summary

Next is the summary for the presentation.



There have been significant updates to Web services in WebSphere Application Server version 7. The content from the feature pack for Web services has been integrated into this release, and been improved on in numerous ways. Updates to the support for various Web services specifications have occurred, including JAX-WS 2.1, JAX-B 2.1 and others. The JAX-WS development model has seen enhancements in the support of JSR 109, and the ability to expose EJB 3.0 beans as services, and the inclusion of a SOAP over JMS transport option for JAX-WS services.

Section

Appendix

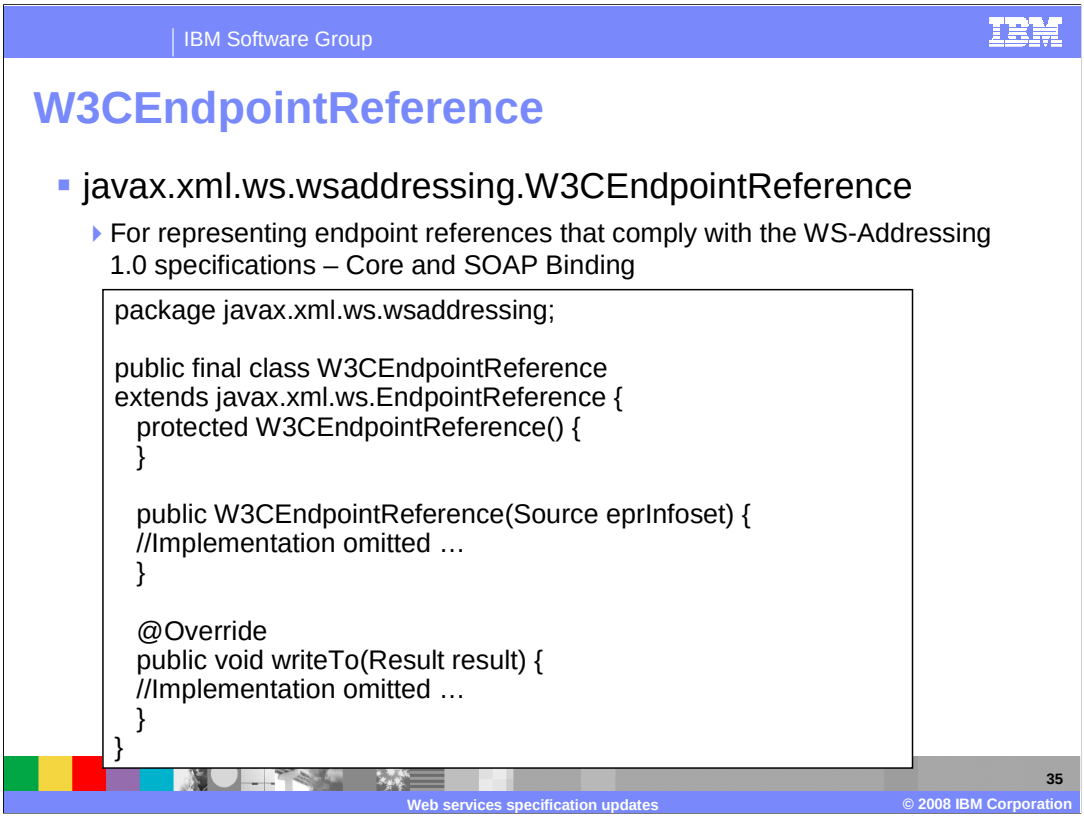
Following is the appendix for this presentation.

EndpointReference

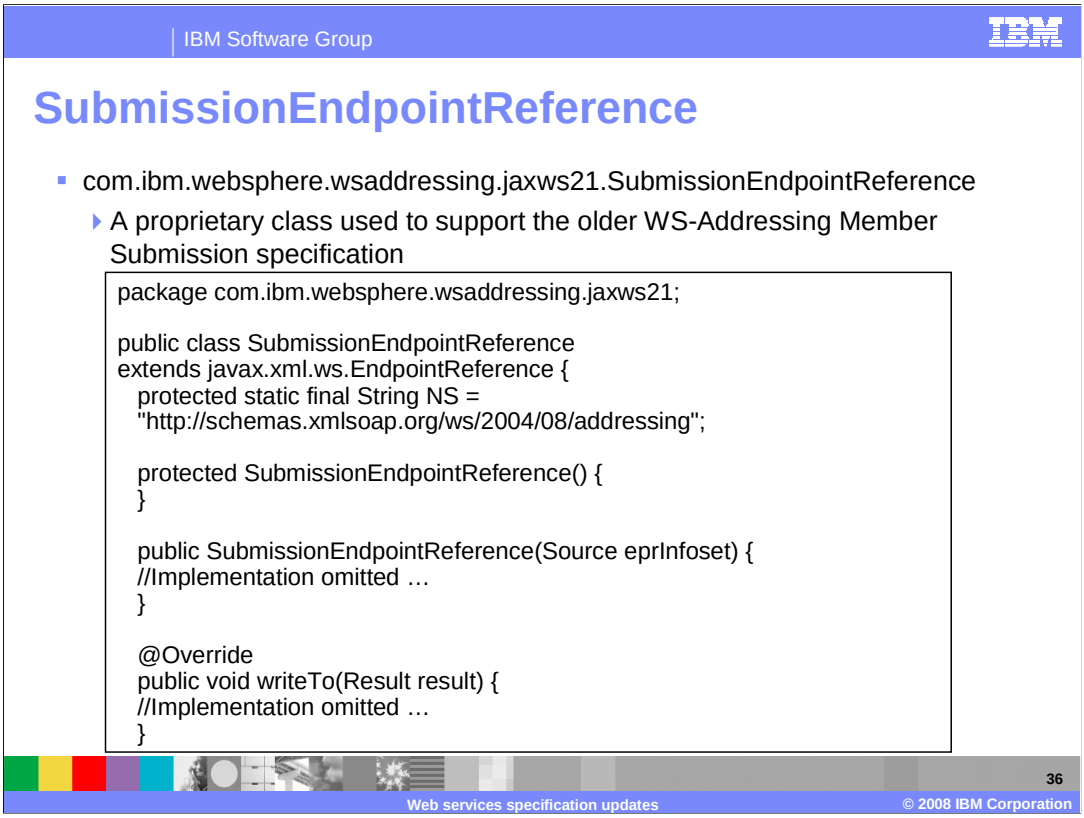
- javax.xml.ws.EndpointReference
  - For representing endpoint references

```
package javax.xml.ws;

public abstract class EndpointReference {
   protected EndpointReference() {
   }

   public static EndpointReference readFrom(Source epr) {
   //Implementation omitted …
   }

   public abstract void writeTo(Result result);

   public <T> T getPort(Class<T> sei, WebServiceFeature... features) {
   //Implementation omitted …
   }
}
```


A client application developer will be able to pass an instance of one of the supported subclasses of EndpointReference to the JAX-WS 2.1 client-side API in order to invoke a Web service endpoint. An application developer will also be able to create new EndpointReferences using client-side or server-side APIs. On the server-side, it will also be possible to return an EndpointReference as the response to a Web service invocation. The content of the EndpointReference is then sent to the client in the body of the SOAP message.



The W3C endpoint reference class is a subclass of javax.xml.ws.EndpointReference it can be used anywhere that the superclass can be used. JAXB 2.1 will automatically bind instances of W3CEndpointReference to the WSAddressing 1.0 schema for endpoint references.



JAXB 2.1 will not automatically bind instances of SubmissionEndpointReference to the WS-Addressing Member Submission schema for endpoint references.

WebSphere Application Server version 7 ships an external bindings file that can be passed to the JAXB tools, using the command line interface, so that the proprietary endpoint reference type can be used in generated code.

Action annotation

- Allows explicit association of a WSAddressing Action message addressing property
  - With input, output, and fault messages of the mapped WSDL operation

```
package javax.xml.ws;

public @interface Action {

   public FaultAction[] fault() default {};
   public String input() default "";
   public String output() default "";
}
```


The action annotation allows explicit association of a WSAddressing Action message addressing property with input, output, and fault messages of the mapped WSDL operation. You can see its usage in the sample shown here.

FaultAction annotation

- Used inside an Action annotation
  - Allows an explicit association of a WS-Addressing Action message addressing property with the fault messages of the WSDL operation

```
package javax.xml.ws;

public @interface FaultAction {

   public Class className();
   public String value() default "";
}
```


The fault action annotation is used inside an Action annotation to map the WS-Addressing Action message addressing property with the fault messages of the WSDL operation.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WASv7_WebServicesPMEnhancements.ppt

This module is also available in PDF format at:
../WASv7_WebServicesPMEnhancements.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

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 WebSphere

A current list of other IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

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

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 or changes in the products or programs 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 (for example, 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.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.