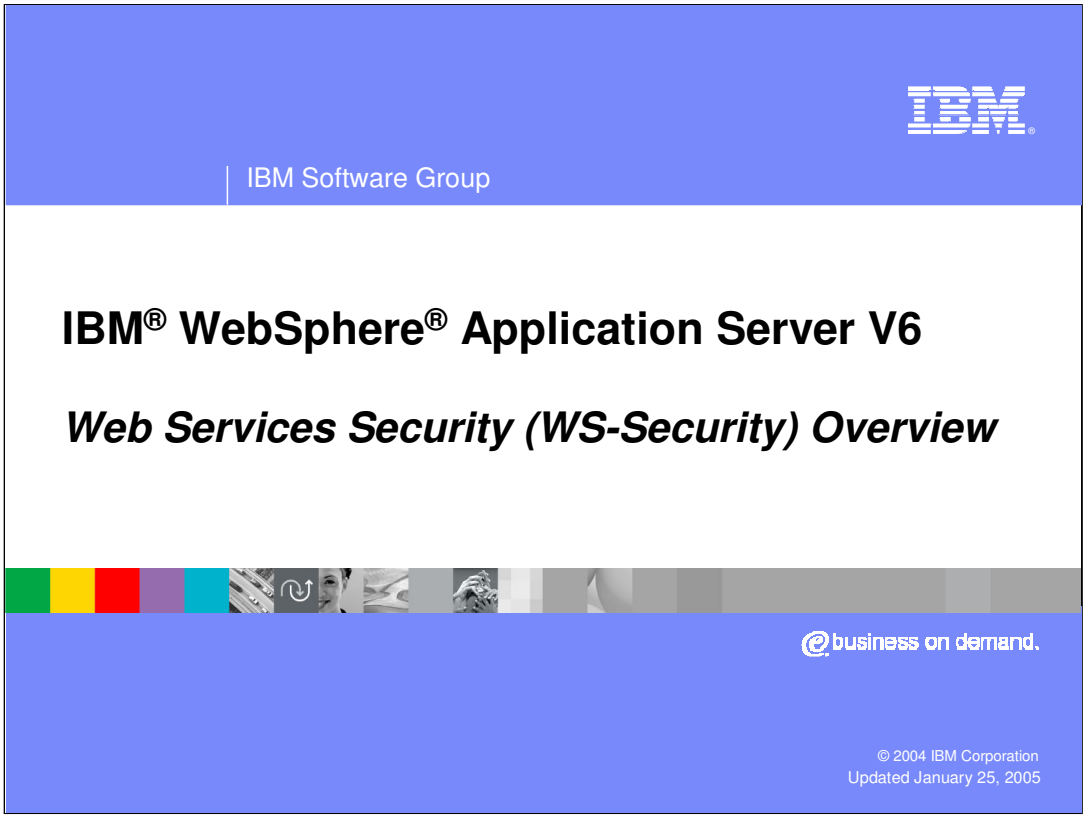
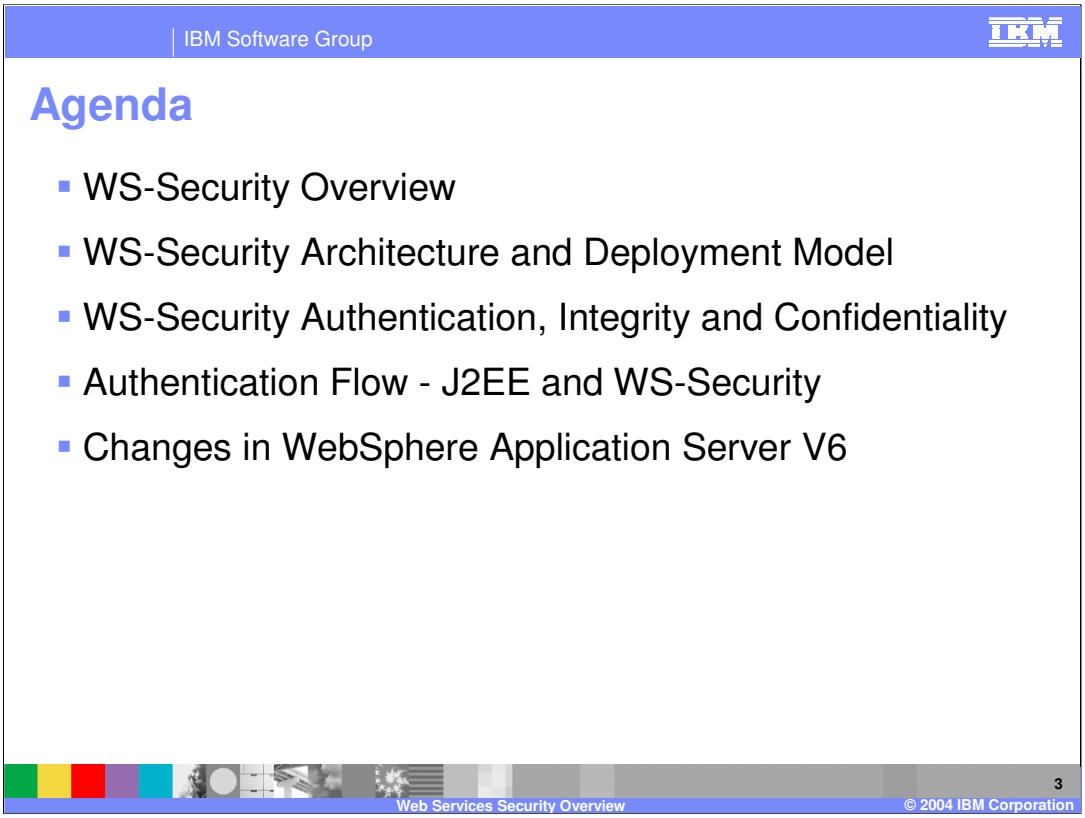


This presentation will focus on Web Services Security support in WebSphere Application Server.

Goals

- Provide an overview of
  - WebSphere Application Server implementation of WS-Security
- Prerequisite:
  - Basic understanding of Java™ 2 Enterprise Edition (J2EE) 1.4 Web Services (JSR 101 and 109)



This presentation will begin by discussing the WS-Security overview, authentication, integrity and confidentiality.

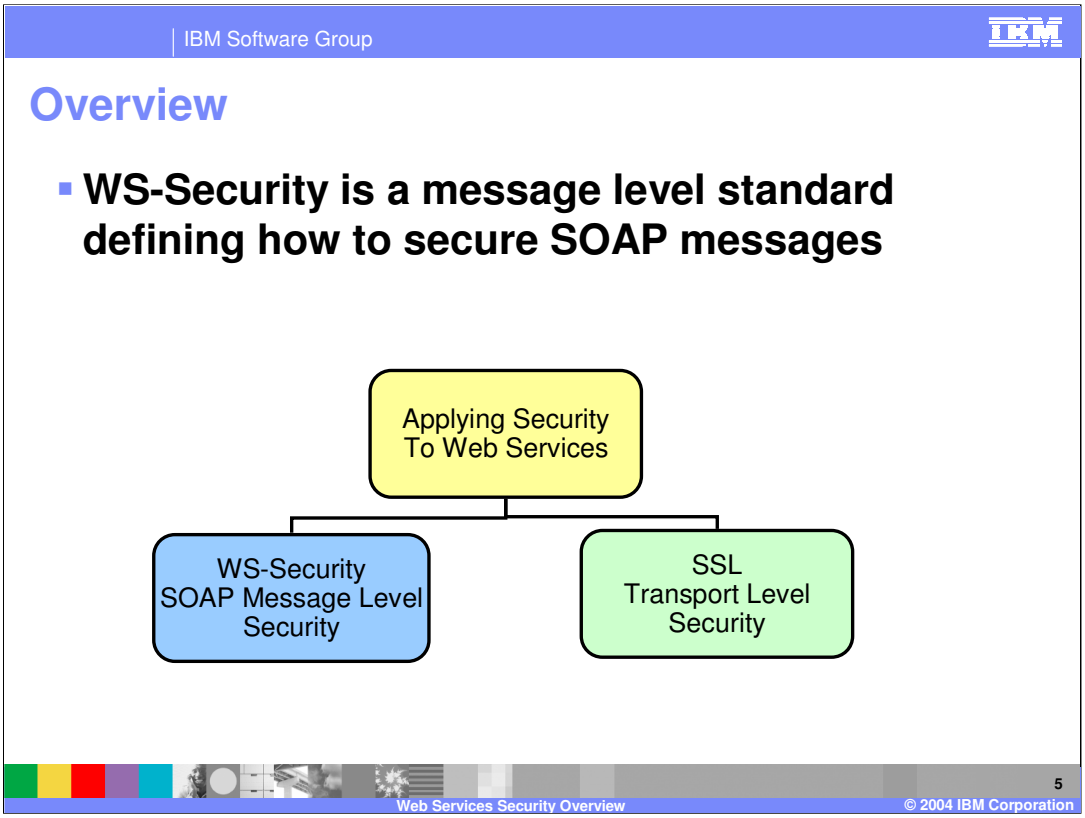
It will also discuss the authentication flow, since with WS-Security, the security information is flowing through the SOAP message. There are two authentication flows to be considered, the J2EE authentication flow and WS-Security authentication flow.

This presentation will also discuss the WS-security deployment descriptors which are comprised of Web Services bindings and extension files which are used exclusively for specifying the WS-Security constraints on the client and server side.

Section

WS-Security Overview

This section is an Overview of Web Services Security.



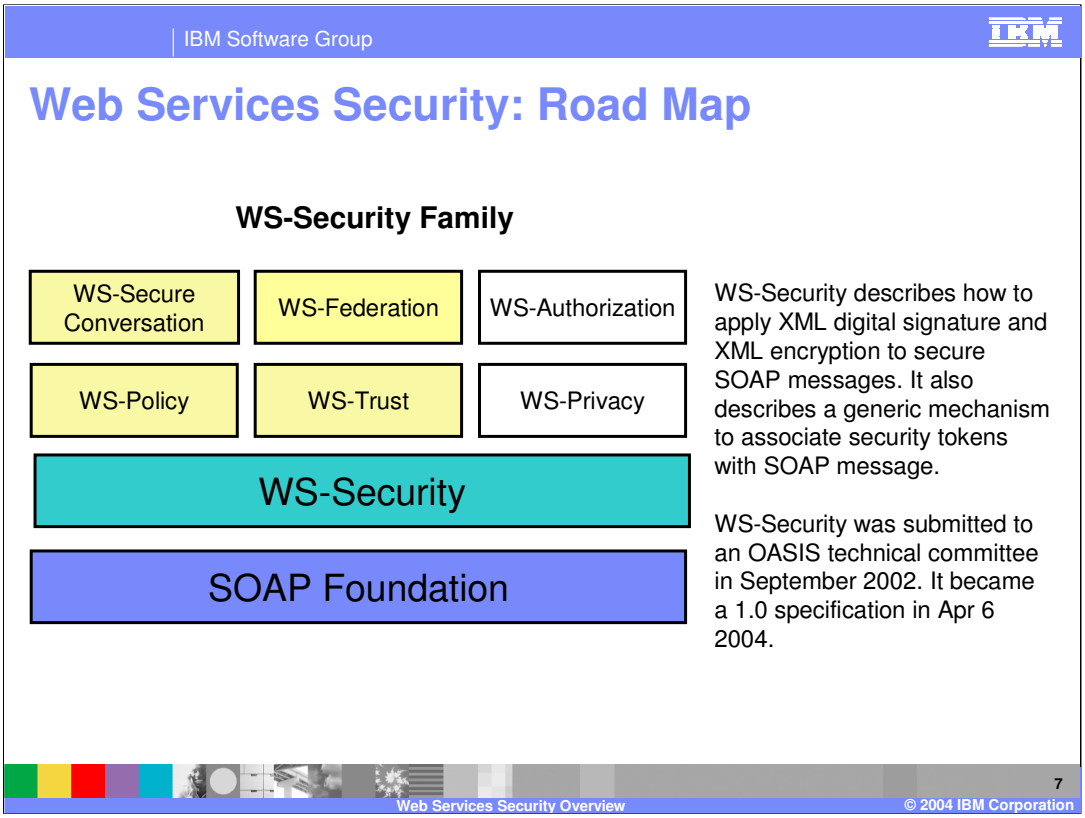
Web Services security for WebSphere Application Server is based on standards included in the Web services security (WS-Security) specification. Web services security is a message-level standard, based on securing Simple Object Access Protocol (SOAP) messages through XML digital signature, confidentiality through XML encryption and credential propagation through security tokens.

Transport level security, like SSL, can also be used to secure Web Services. Transport channel security encrypts the entire message, For HTTP this means encrypting the entire HTTP message. WS-Security provides an alternative method for security on services.

WS-Security is a message level standard which means that security information is part of the SOAP message sent from the client. The client, based on the constraints in the web services binding and extension files will insert WS-Security information in the SOAP message. And the server, based on the constraints in the server-side Web Services binding and extension files will check for the security constraints in the incoming SOAP message's header.

Web Services security defines the core facilities for protecting the integrity and confidentiality of a message and provides mechanisms for associating security-related claims with the message. The security constraints can specify XML Digital Signature, for Message Integrity, XML Encryption, for Message Confidentiality, and credential propagation. WS-Security actually applies to both SOAP/HTTP as well as SOAP/JMS. It can be used in any protocol that sends a SOAP message.

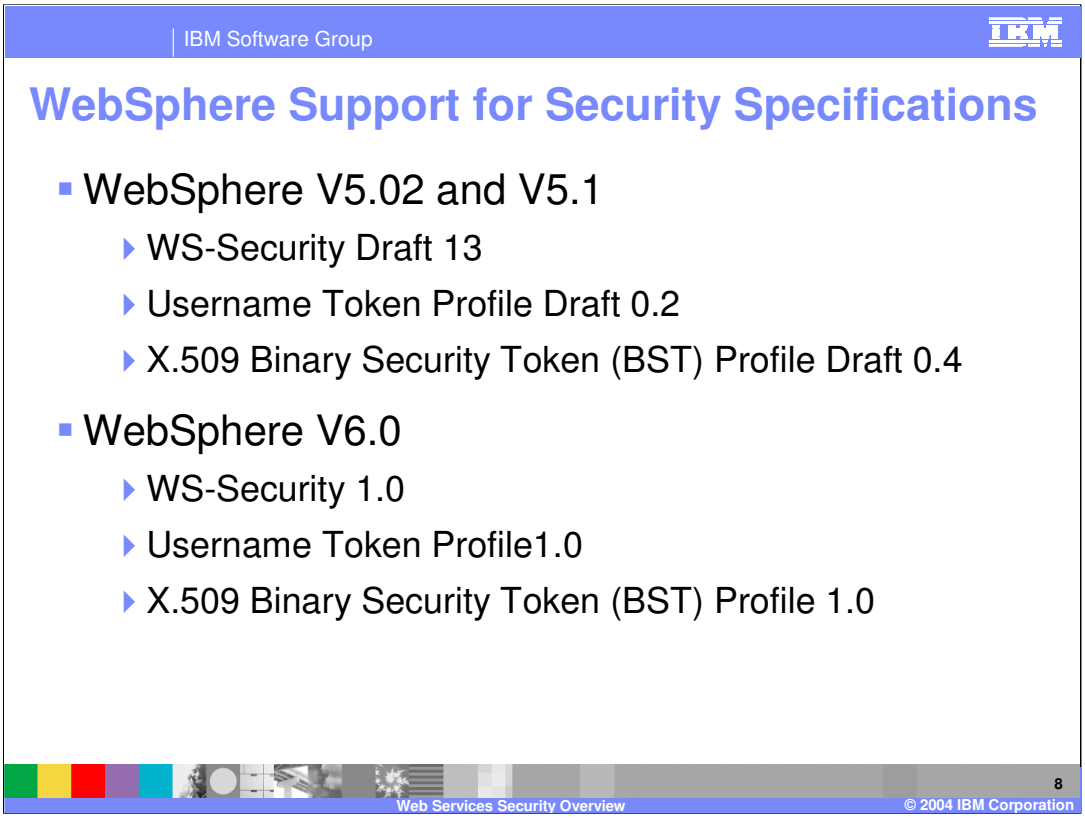
The WS-security constraints have to be specified within the Web Services bindings and extensions.



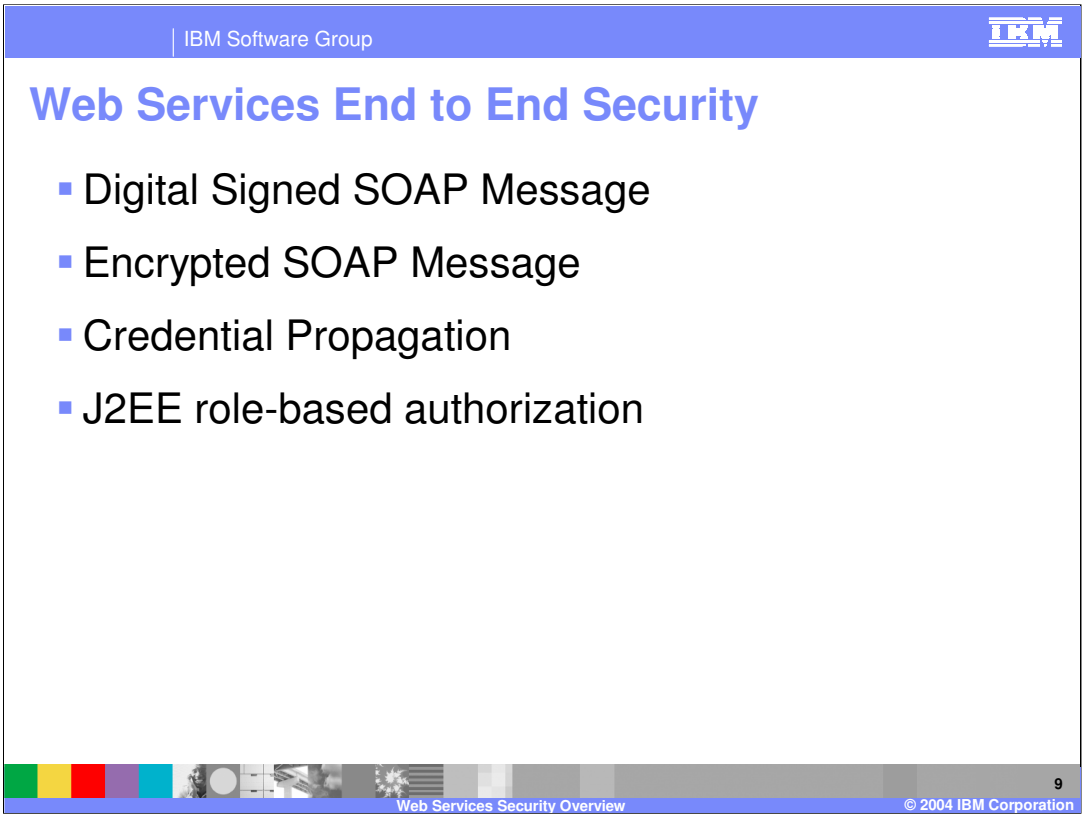
The SOAP foundation is in specifications through 1.1, with 1.2 being worked on now.

The WS-Security specification is based on a draft proposed by IBM, Microsoft™ and Verisign in 2002. This has been finalized by the OASIS consortium on April 6 2004 and has become the 1.0 specification.

The higher level boxes, or the more advanced security specifications are still being worked on in the community process.



This slide shows the progress made on Web Services Security support in WebSphere Application Server V6. The three main specification for Web Services Security, including the Username Token Profile and Binary Security Token Profile have all reached 1.0 status. Where as in previous releases of WebSphere Application Server WS-Security was based on draft profiles. A number of other specifications, including XML digital signature and XML encryption also impact WS-Security.



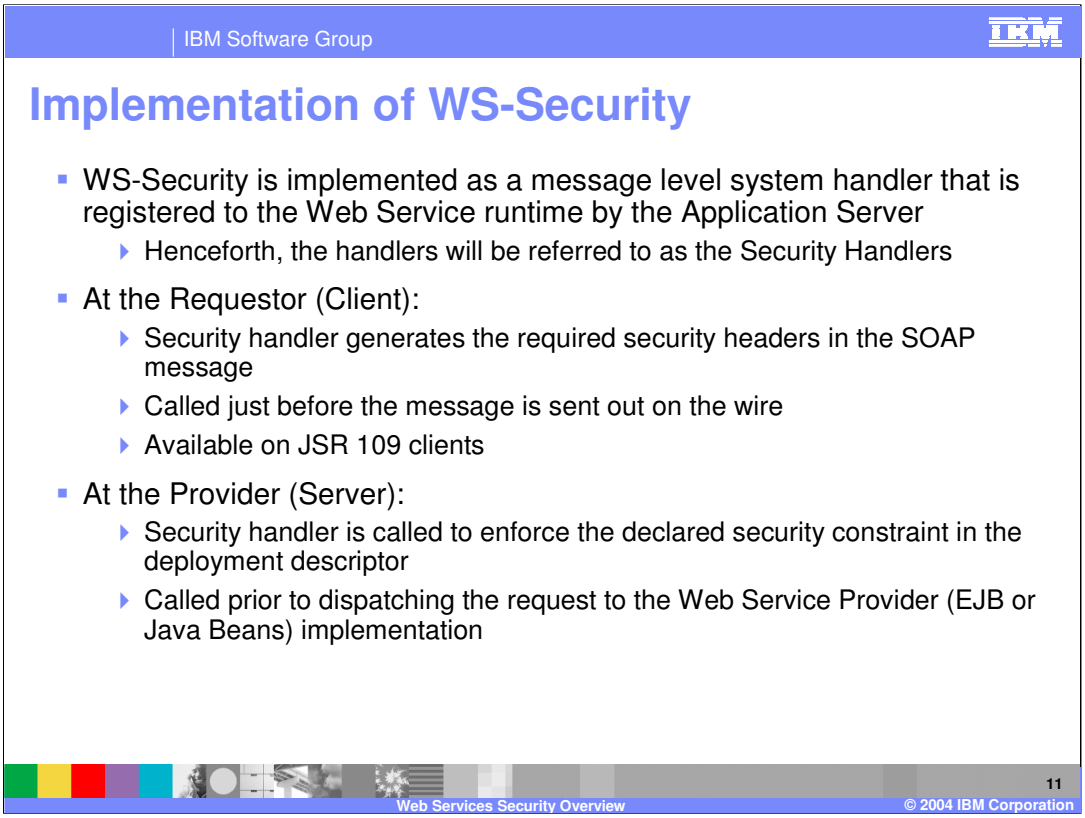
Web Services end-to-end security involves digital signed SOAP message, encrypting the SOAP message, credential propagation and J2EE role-based authorization.

To realize the benefits of Web services security, it is recommended that an implementation of the specification is integrated with underlying security mechanisms. This implementation is fully integrated with the WebSphere Application Server, Version 6.0 security infrastructure. Authorization, for example, is based on the J2EE security model. When a user ID and password are embedded in a request message, authentication is performed with the user ID and password. If successful, a user identity is established in the context of execution and further resource access is authorized based on that identity. The authentication process is similar to the process for HTTP basic authentication. Once the user ID and password are authenticated, authorization is performed by a J2EE container.

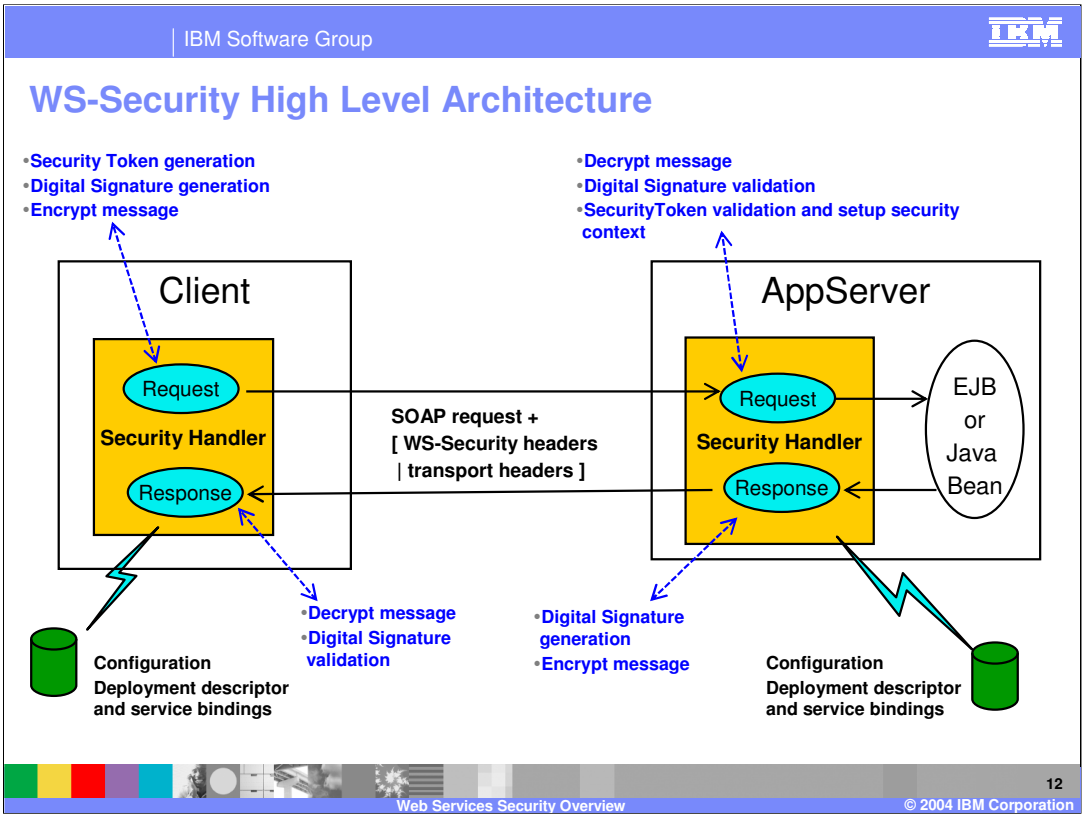
Section

WS-Security Architecture
and
Deployment Model

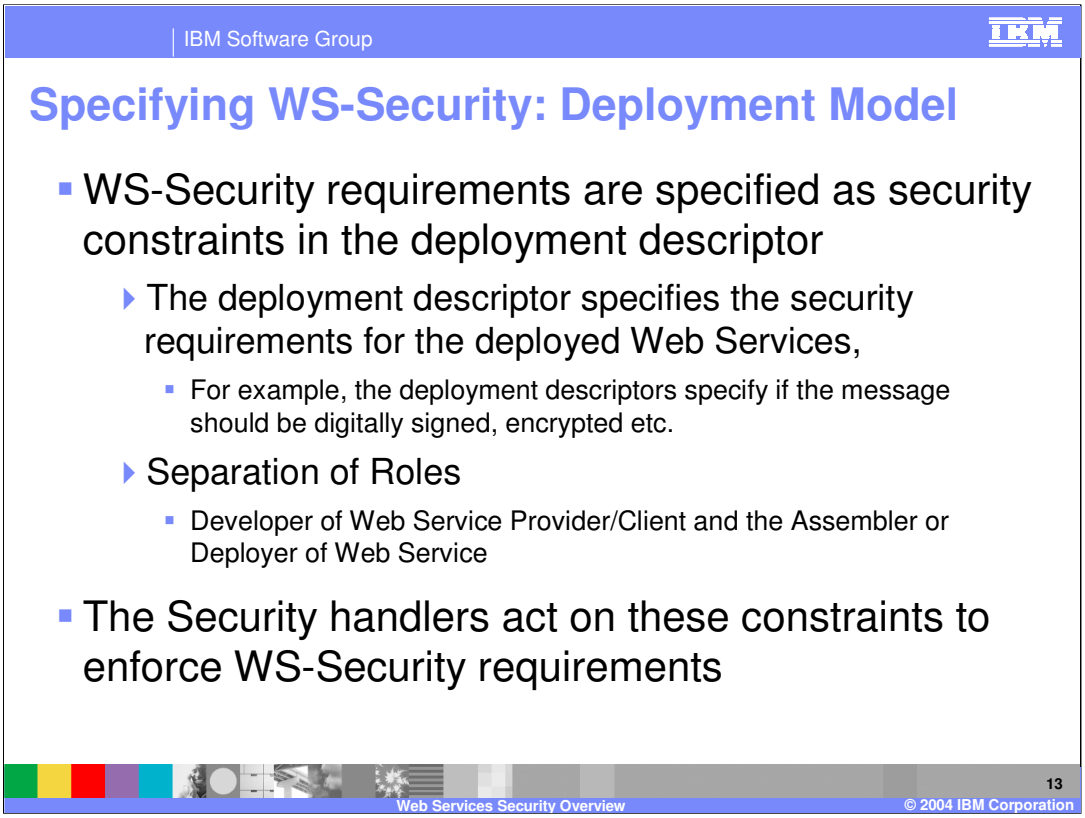
This section covers the architecture and deployment model for Web Services security.



WS-Security is designed and implemented as message level handlers of the Web Services engine, as a system handler. WS-Security handler is a “system handler” and is registered to the Web Service runtime. On the client side, the WS-Security handler is invoked to generate the required security headers in the SOAP message before the message is sent out over the wire. The security handler generates the security constraints defined in the deployment descriptor and packages the security information (digital signature, encrypted data and security tokens) into the SOAP message. On the server side, the WS-Security handler is called to enforce the declared security constraints in the deployment descriptor prior to dispatching the request to the Web Service EJB or Java Beans implementation. The security constraints of the request sender and request receiver must match. Also, the security constraints of the response sender and response receiver must match. For example, if you specify integrity as a constraint in the request receiver, then you must configure the request sender to have integrity applied to the SOAP message. Otherwise, the request is denied because the SOAP message does not include the integrity specified in the request constraint.



This slide has a high level look at the security architecture as it applies to Web Services. On the left side of the slide is a client application, making a request to the Web Service provider on the right side of the slide. The security handler associated with the request from the client application creates and applies security tokens, digital signatures, and encrypts the message. The information for which security operations to perform on the message are stored within deployment descriptors and binding files associated with the client. This information is accessed by the security handler through the runtime. The SOAP message with the security information is then sent to the service provider. Within the security provider's application server, the security handler decrypts the message and validates security tokens and digital signatures. The handler will again check information stored in the deployment descriptors and binding files to determine the appropriate security to expect on the message. If the appropriate security is not found within the message, the request will be denied. If a response is made, the security handlers will add and check security to the response message in the same way.



The Web services security model employed by WebSphere Application Server is a declarative model. You can secure an application with Web Services Security by defining security constraints in the IBM extension deployment descriptors and IBM extension bindings. The development life cycle of a Web services security-enabled application is similar to the Java 2 Platform, Enterprise Edition (J2EE) model since it enables separation of roles, whereby the component provider creates the J2EE module, then the assembler adds declarative security constraints to the J2EE module. The deployer then takes the Web Service enabled J2EE application with Web Services security and deploys it to the runtime environment. The Web Services security handler acts on the security constraints defined in the IBM extension deployment descriptor and enforces the security constraints accordingly.

WS-Security SOAP Faults

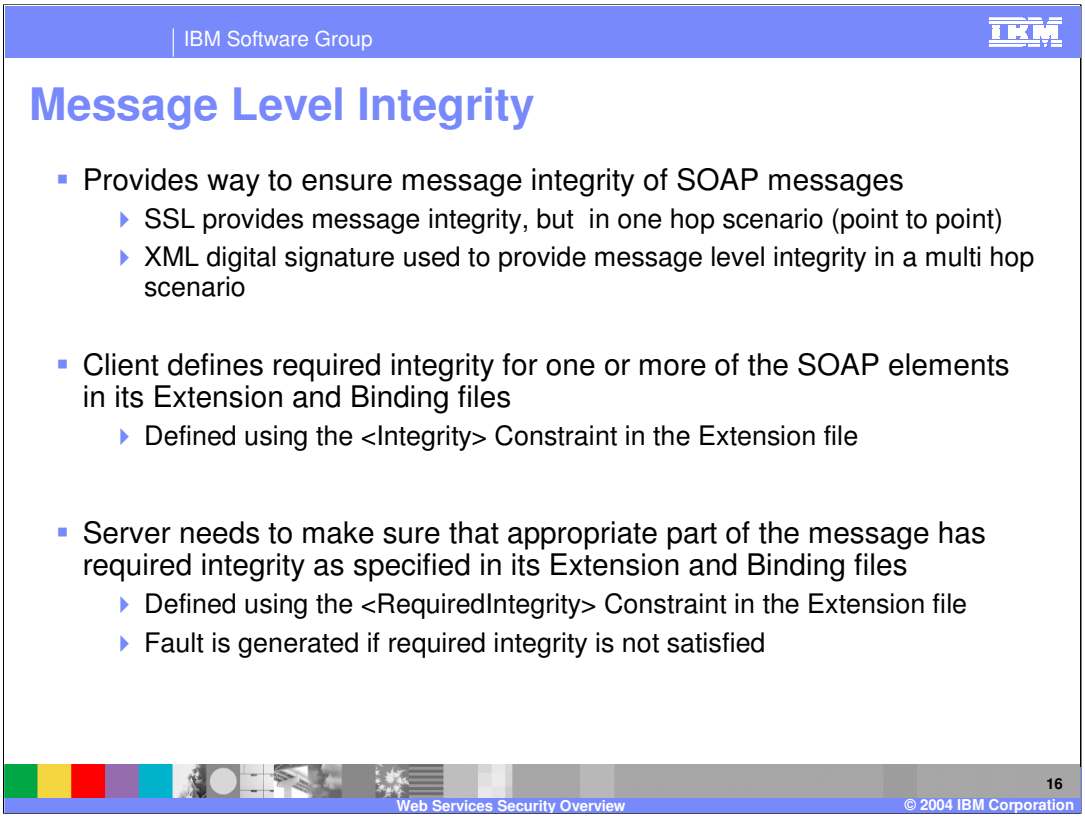
- If the Security constraints requirements, as defined in the deployment descriptor, are not satisfied, a SOAP fault in the SOAP response will be sent to the client
- Errors could result from:
  - Invalid or unsupported type of security token, signing or encryption algorithms
  - Invalid or unauthenticated or invalid security token (token that can not be authenticated)
  - Signature verification failures
  - Decryption failures
  - Referenced security token could not be located

There are many circumstances where an error can occur while processing security information. A number of these are listed on the slide. When a failure occurs, then the failure is reported to the client or sender using the SOAP specifications Fault mechanism.

Section

WS-Security
Authentication, Integrity, Confidentiality

This section discusses the support for authentication, integrity and confidentiality information in Web Services Security.



Integrity is the concept that data has not been changed or lost in an unauthorized or accidental manner.

The Secure Socket Layer (SSL) along with the de facto Transport Layer Security (TLS) can be used to provide transport level security for Web Services applications. These offer several security features including authentication, data integrity and data confidentiality. However SSL/TLS enables only point-to-point secure sessions.

XML digital signature is used to provide integrity in a multi-hop scenario.

You can select multiple parts of a message to be digitally signed. The security handler on the server side of the SOAP message enforces these security specifications.

Message Level Confidentiality

- Encryption provided by WS-Security is based on the XML Encryption specification
- Client defines required Confidentiality for one or more SOAP elements in its Extension and Binding files
  - Defined using the <Confidentiality> Constraint in the Extension file
- Server needs to make sure that appropriate part of the message has required Confidentiality as specified in its Extension and Binding files
  - Defined using the <RequiredConfidentiality> Constraint in the Extension file
  - Fault is generated if required confidentiality is not satisfied

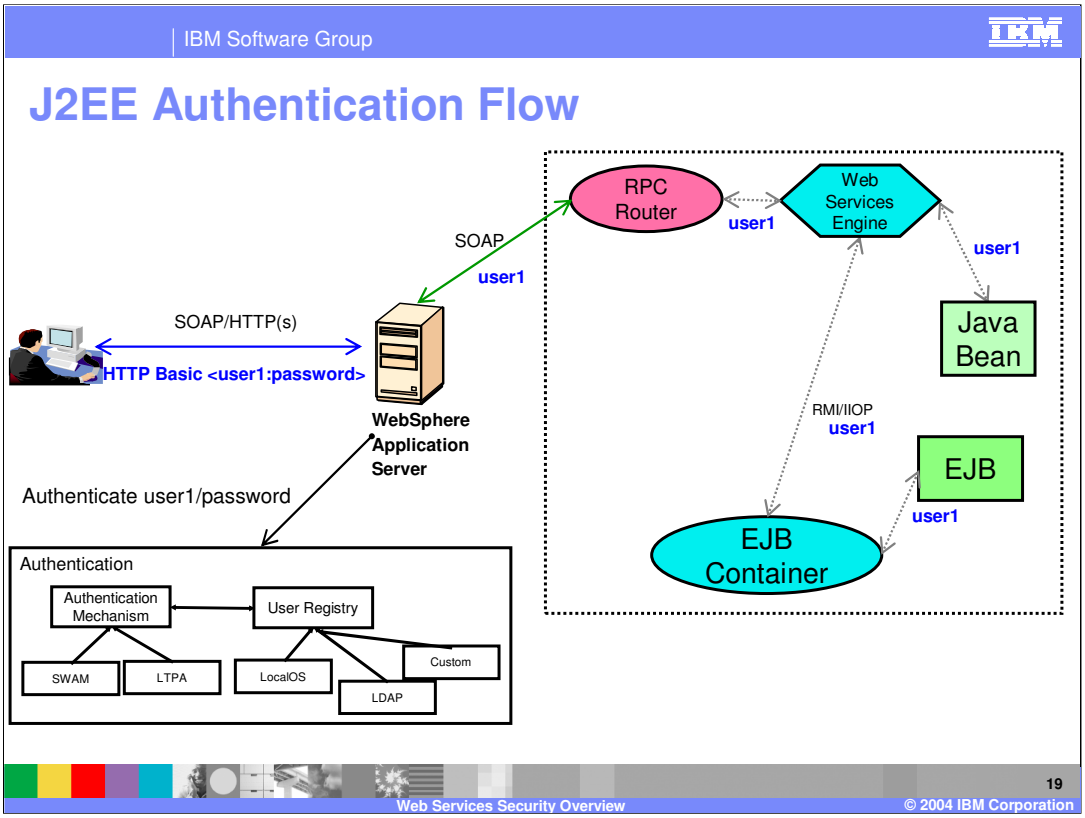
Confidentiality is the process by which data is protected such that only intended recievers can view the data. Message Confidentiality is provided by leveraging XML encryption to keep portions of SOAP messages confidential.

In WebSphere Application Server confidentiality has to be specified declaratively as constraints within the client and server web services extension files.

Section

Authentication Flow of Security Tokens

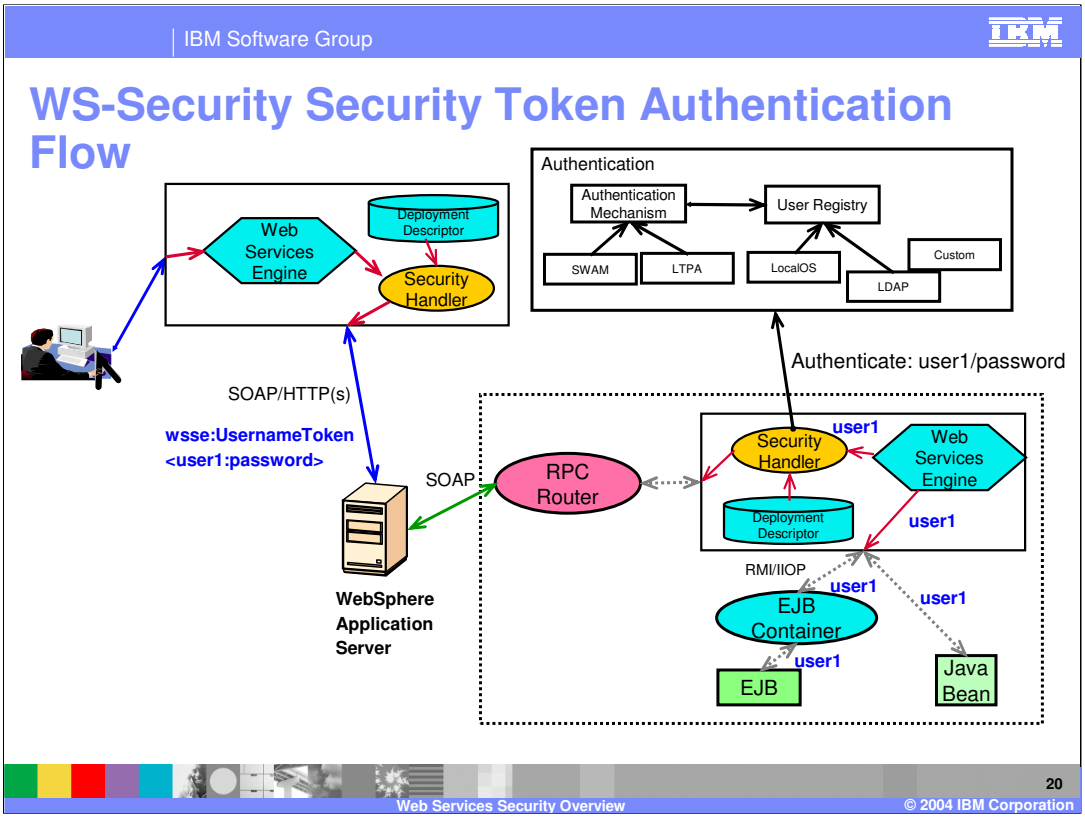
This section explains the flow of security tokens in WebSphere Application Server.



You can secure Web services using the existing security infrastructure of WebSphere Application Server, J2EE role-based security, and SSL transport level security.

The Web services endpoint can be secured using J2EE role-based security. The Web services sender can be used to send the basic authentication data in the HTTP header. SSL can be used to secure the transport level. When the WebSphere Application Server receives the SOAP message, the Web Container authenticates the user (in this example, "user1") and sets the security context for the call. After this is complete, the SOAP router servlet sends the request to the implementation of the Web Services. For Enterprise Java Beans implementations, the EJB container performs an authorization check against the identity "user1".

The Web services endpoint can also be secured using the J2EE role. In this case the authorization check is performed before the request is sent. This might be the only way to get to "coarse grain authorization" for Java Bean Web services implementation.



You can also secure Web Services using Web Services security at the message level. In this case, you can digitally sign or encrypt a certain part of the SOAP message. Web Services security also supports security token propagation within the SOAP message. This scenario assumes that the Web services endpoint is not secured with J2EE role-based security and that the Enterprise Java Bean is secured with J2EE role-based security.

In this case the Web Services endpoint is not secured with J2EE role-based security. The Web Services engine processes the SOAP message before the client sends the message to the Web Services endpoint. The Web Services security runtime acts on the security constraints, such as digitally signing, encrypting, or generating and inserting a security token in the SOAP header. In this case a token is generated using "user1" and "password". On the server-side, the Web Service processes the incoming message and Web Services security enforces security constraints. This includes making sure messages are properly signed, properly encrypted, and decrypted, authenticating the security token, and setting up the security context with the authenticated identity. Finally, the SOAP message is sent to Web Service implementation.

As you can see, Web services security can complement J2EE role-based security. For example, SSL can be enabled at the transport level to provide a secure channel, and if the Web services implementation is Enterprise Java Bean you can leverage the EJB authorization by performing authorization checks. The Web services security runtime leverages the security infrastructure in order to set the authenticated identity in the security context. The authenticated identity can be used in the downstream call to J2EE resources (or other resource types).

Note that if you use SOAP over JMS, Web Services security is the only way to propagate security token from sender to the receiver.

Section

WS-Security Enhancements in V6

This section details Web Services Security enhancements in WebSphere Application Server V6.

WS-Security Changes

- WS-Security support updated due to maturation of the specification
- Focus on making WS-Security extensible in WebSphere Application Server V6
  - A pluggable architecture to allow signing / encrypting with custom security tokens, and sending and processing multiple security tokens
- No APIs exposed at this time

One of the major design goals is to make WS-Security implementation extensible. This is an important requirement, as the WS-Security specification is very flexible and IBM cannot cover all the possible combinations allowed in the specification. Also, the pluggable architecture allows others to implement other Web Services security specifications like WS-Trust client or WS-SecureConversation, or WS-SecurityKerberos profile. Because of this there will be a technical preview of base Kerberos support as part of WebSphere Application Server V6. Since the JSRs are still in progress or review status, there are no APIs exposed for WS-Security in WebSphere Application Server V6. You should still use the deployment model to express the security constraints.

WS-Security Extensibility

- Pluggable Signing / Encryption algorithms (based on the JCE framework)
- Pluggable Token
  - Enhanced to support multiple tokens and tokens can be used for signature and encryption
- Pluggable KeyLocator
  - Abstraction for locating a key for signature or encryption
- Signing or encryption any elements in the SOAP message
  - Have to use XPath to specify the items within the message
- Order of signature or encryption is performed

In addition to support for the specification, a highly flexible architecture is required so that developers can implement other specifications such as WS-Trust, WS-SecureConversation and WS-SecurityKerberos on the current architecture. In order to take advantage of these Web Services Security features, the application must be a J2EE 1.4 application.

Although a base framework is provided, default implementation classes also needed to support basic functions of WS-Security and profiles so that developers can use these functions without any development.

Backward Level Support for Services

- Web Services with WS-Security in WebSphere Application Server V6 have different deployment descriptors than services in V5.X
- WebSphere Application Server V6 will include support for J2EE 1.3 services using earlier versions of WS-Security
- The Administrative Console will provide different screens to configure back-level security for back-level services

WebSphere Application Server V6 will provide backward level support for J2EE 1.3 Web Services. Part of this includes support for the old IBM implementation of WS-Security. Depending on the level of the service you are configuring the Administrative console will offer appropriate options for a J2EE 1.3 or J2EE 1.4 service.

Summary

- Discussed the WebSphere Application Server implementation of the WS-Security standard
  - Architecture
  - Deployment Model
  - Authentication, Integrity and Confidentiality

This presentation explained the concepts behind Web Services Security. It also explained the changes that have occurred to the support for WS-Security in WebSphere Application Server V6.

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.