



IBM Software Group

IBM WebSphere® Application Server V6.0.1

Java™ 2 Enterprise Edition (J2EE) Application Security



@business on demand.

© 2004, 2006 IBM Corporation
Updated October 28, 2006

This presentation will focus on the J2EE Application Security.

Goals

- Understand Application level J2EE security in WebSphere Application Server
- Understand IBM extensions to the specification



The goal of this presentation is to help you understand the J2EE Security at the Application level. Some of the IBM extensions will also be discussed.

Agenda

- J2EE Application Security overview
- J2EE Application Security of EJB components
 - ▶ Run-As options
- J2EE Application Security of Web components
- Application Security Tasks and Roles



The agenda for this presentation is to provide an overview of J2EE security, discuss application security as it relates to EJB and Web components, and discuss tasks and roles.

Section

J2EE application security: Specifying authorization



This section will cover how to specify authorization information for your applications.

Creating secure J2EE applications: Overview

- J2EE application level security is specified using security roles
- Security roles allow developers to specify security at an abstract level
- Security roles are applied to the Web and EJB application components
 - ▶ EJB methods or Web URIs
- Security can be specified in the following ways:
 - ▶ Declaratively at assembly time, through the deployment descriptors
 - ▶ Programmatically using standard APIs at development time
- Binding of users and groups to J2EE security roles is usually done at the application deploy (install) time

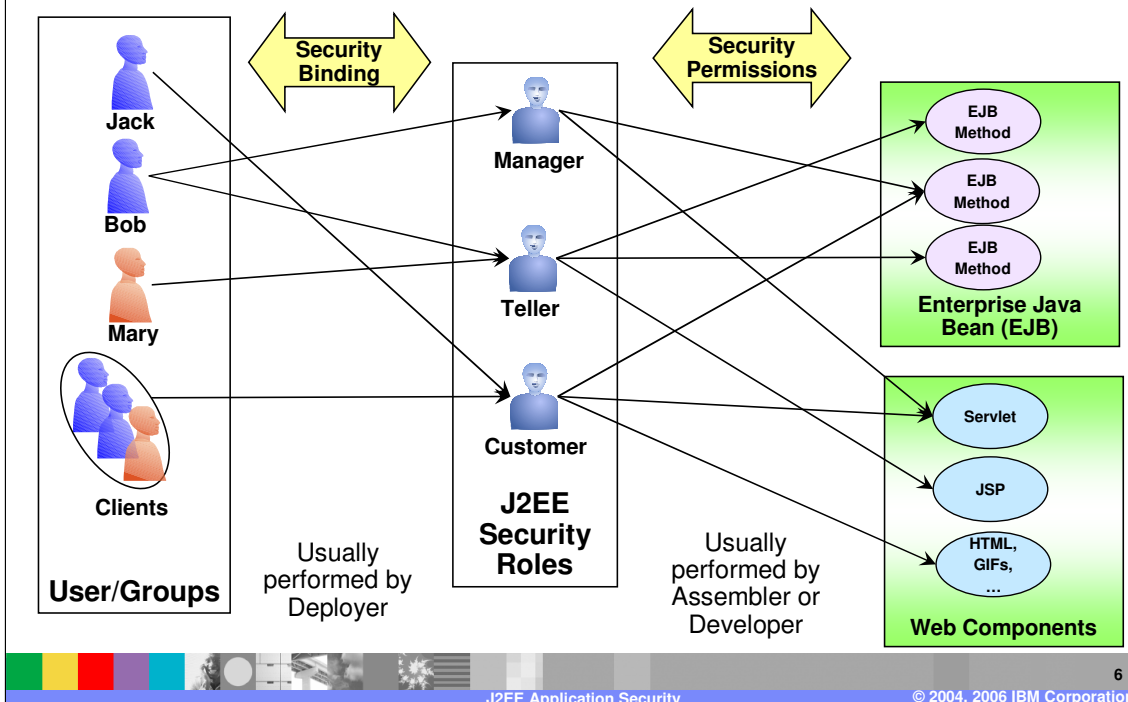


Developers generally do not know the actual users and groups that need access to a particular application. As a result, J2EE defines Security Roles that developers can use to provide authorization information. Developers provide access permission to the Security Roles for different parts of the application. They can do this Declaratively or Programmatically. This will be explained in the next few slides.

The binding or association of the roles to the actual users, groups, or both is done by the System Administrator during application deployment on the Application Server, or later once the application has been deployed. On z/OS®, if SAF authorization is being used, EJBROLE profiles must be added and users given access to the various roles as appropriate.

J2EE Security roles apply to the entire application and all its modules. This information is saved in the Application and the module deployment descriptors.

Securing J2EE application artifacts



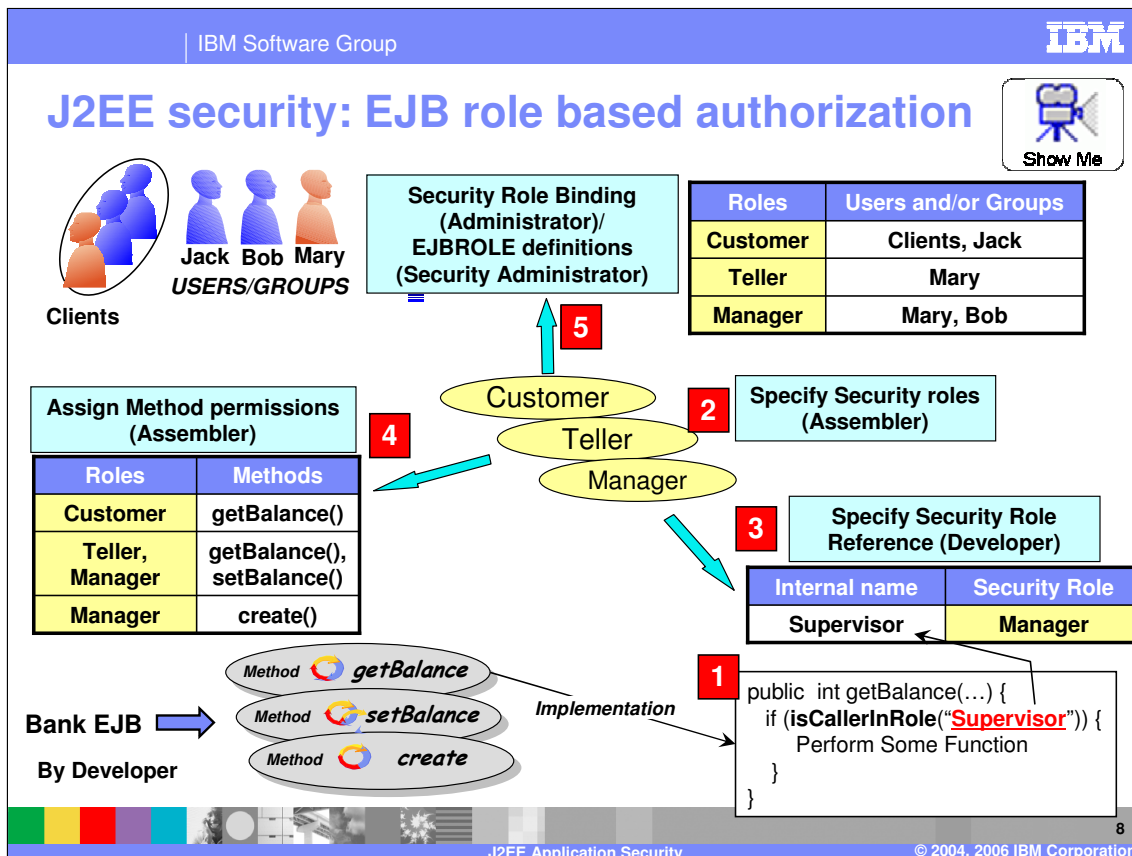
This page shows the actual process. The Developer or the Assembler will define the J2EE Security roles and specify permissions of what application artifacts can be accessed by these roles. The deployer then binds the J2EE Security roles to the users or groups. Again, if SAF authorization is being used on z/OS, this would be done by the Security Administrator, who would need to give the appropriate users access to the various EJBROLES (for example Manager, Teller, Customer).

Section

J2EE application security: EJB components



This section will cover how to specify authorization information for EJB components.



For a demonstration on how to add J2EE security roles for Web applications, pause this presentation and click the Show Me icon on the upper right side of the slide.

The goal is for a developer or an assembler to specify who can access the methods within an EJB.

In some cases, the developer might not be the same person as the Assembler of the application. A developer might not know the Security roles that may be defined later by an assembler. In that case, the developer can use an internal security role name when making programmatic API calls to check for security, as shown in step 1. It will be up to the assembler to tie the internal role name to the security role by defining a security role reference, as shown in step 3.

In the example, "Supervisor" is the internal local name. Later, the assembler will have to map the internal name to the Role defined by the assembler.

In the life cycle of the application, step 2 shows an assembler defining a set of J2EE Security Roles for the application

Now that the real Security Roles have been defined, the assembler must map any internal Role name used by the developer to the Security Role reference, as shown in step 3.

The Assembler can assign EJB method permissions, as shown in step 4. This is associating the methods to the Security roles. Once done, any user or group that maps to the Security role will have permission to access the method. You can use wild cards to specify multiple methods.

The application is now ready for deployment into an Application Server. During deployment, as shown in step 5, the System Administrator can bind the security roles to the users or groups that are in the Security User Registry for the Application Server. The Administrator can change the role to user or group binding at any time after the installation. If SAF authorization is being used on z/OS, step 5 would be done by the Security Administrator. The Security Administrator would have to ensure that each role is defined as an EJBROLE and then give the appropriate users and groups access to those roles.

EJB applications programmatic APIs

- **isCallerInRole** (String role-name)
 - ▶ Returns true if the bean caller is granted the specified security role
 - ▶ If the caller is not granted the specified role, or if the caller is not authenticated, it returns false
 - ▶ If the specified role is granted **Everyone** access, it always returns true
 - ▶ Must have security role reference defined in the deployment descriptor
- **getCallerPrincipal()**:
 - ▶ Returns the java.security.Principal object containing the bean caller name
 - ▶ If the caller is not authenticated, it returns a principal containing UNAUTHENTICATED name

Example:

```
public void myEJBmethod() {
    ...
    // to get bean's caller using getCallerPrincipal()
    java.security.Principal principal =
    context.getCallerPrincipal();
    String callerId= principal.getName();

    // to check if bean's caller is granted Mgr role
    boolean isMgr = context.isCallerInRole("Mgr");
    ...
}
```

9

J2EE Security defines two APIs that can be used by the EJB developer:

The **isCallerInRole()** method is used to determine if the authenticated caller is in a specific Role defined by the application. This lets you decide that a certain part of the code is to be allowed to run only for a user in a specific Role.

The **getCallerPrincipal()** method is used to return the authenticated caller.

The example provides how the developer could use these APIs.

Changing identity: “Run-As” option

- EJB methods have the ability to change identity when calling downstream processes or EJBs
- There are several different “Run-As” identities that you can choose from
- Run-As specification applies to all the methods of the EJB
 - With IBM extension, you can specify different “Run-As” options for different methods within the same EJB

“Run As” options	Description
Client Identity	<ul style="list-style-type: none"> ▪ Bean takes on the same identity as the caller
Another Specified Role	<ul style="list-style-type: none"> ▪ Bean takes on identity of a specified user within the specified role ▪ The specified role is part of the deployment descriptor and performed by the assembler ▪ The specific user in the “Run-As” role is usually specified at deploy time
Server Identity	<ul style="list-style-type: none"> ▪ Bean takes on the identity of the user under which the server is running ▪ This is an IBM extension to the specification



The Run-As option is a way to change the identity of the caller from an EJB when calling a downstream EJB.

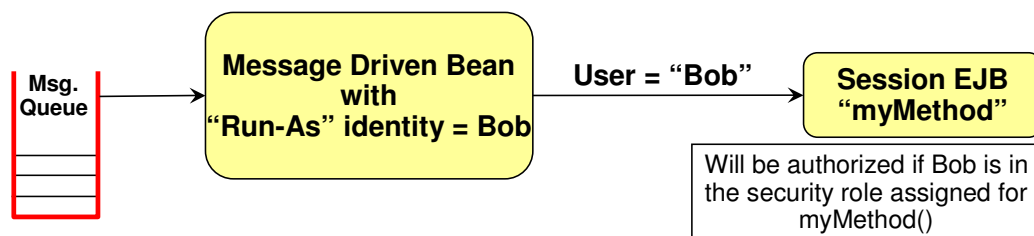
An example is where a client cannot directly call a downstream EJB, but can call an upstream EJB, which can call a downstream EJB by changing its identity.

There are three Run as options, as shown in the table. The downstream EJB can be called as the same identity as the original client identity, or can be called as the Server identity, which is the user under which WebSphere Application Server is running. The last option is to run as another specified role.

The Server Identity Run as option is an IBM extension, carried forward from V4 and V5.

Security for Message Driven Bean (MDB)

- Messages arriving at MDB have no client credentials
 - ▶ However, when MDB needs to call a secure EJB, it needs security credentials
- Provide “Run-As” identity for the MDB



Message Driven EJB (MDB) are called by the EJB container when there is a message in the queue. Therefore, there is no client identity. So, if an MDB must call another downstream EJB that requires authentication, providing Run-As on the MDB allows the assembler to send an identity to the downstream EJB, as shown in the example.

Section

J2EE application security: Web components (Servlets/JSPs/HTMLs)



This section will discuss how to specify authorization information for Web components.

Configuring Web components security

- Authentication method – specify how to obtain authentication information for the Web module
 - ▶ Basic authentication, Client certificate authentication and Form-based authentication
- Data constraints – allows you to specify the required transport guarantee that defines the communication between the client and the Web application
 - ▶ None – no transport guarantee requires
 - ▶ Integral – ensures data cannot be changed in transit – SSL used
 - ▶ Confidential – ensures data cannot be viewed in transit – SSL used
- Web resource collection to be protected
 - ▶ Web resources is a set of URL patterns and HTTP methods
 - ▶ For static resources (HTMLs), valid HTTP methods are GET and POST
 - ▶ For dynamic resources (Servlet or JSP), valid HTTP methods are GET, POST, PUT, DELETE, HEAD, OPTION, TRACE

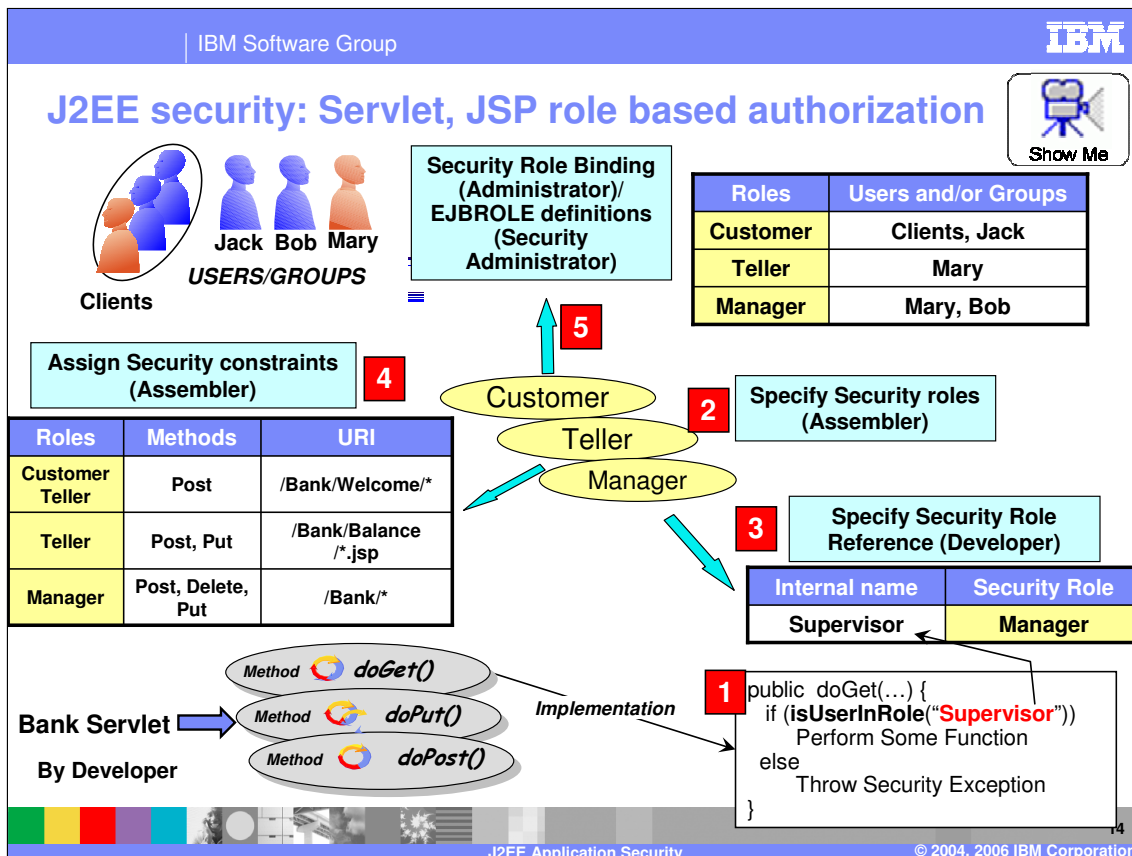
Configuring Web Component security is a little more involved than EJB components.

The Web component must let the server know what type of authentication the Web client should provide, whether basic authentication, client certificate, or form-based custom login.

Next, the Web component developer must specify whether integrity or confidentiality is required. This is specified by the Data constraints.

Finally, the assembler must specify who has access to the set of Web resources within the component. The Web resource is a set of URLs and the HTTP method. A Web resource is accessed using a URL and one of the six methods specified. The assembler can then authorize the different J2EE Security roles to have permission to the Web resource.

All this information is stored in the Web Deployment descriptor.



Applying J2EE Security to Web components is very similar to EJBs.

Steps 1, 2 and 3 are similar to what was explained for the EJB methods.

Step 4 however, is different. For EJBs, the permission is on EJB methods, whereas for Web components, it is on the Web resource. As indicated in the previous page, the Web resource is the collection of the URL and the HTTP methods. Once Web resources are defined, the J2EE Security Roles can then be assigned to the Web resource collection, thereby creating the security constraints.

Again, step 5 of binding the J2EE security roles to the users or groups is similar to EJBs.

Click the Show Me icon for a demo on how to add J2EE security roles for Web application using the tools.

Web applications programmatic APIs

- **isUserInRole (String role-name)**: Returns true if the remote user is granted the specified security role. Returns false, if the remote user is not granted the specified role, or no user is authenticated
- **getUserPrincipal()**: Returns the `java.security.Principal` object containing the remote user name
- **getRemoteUser()**: Returns the user name the client used for authentication.

Example:

```
public void doGet(HttpServletRequest request, HttpServletResponse response) {  
  
    // to get remote user using getUserPrincipal()  
    java.security.Principal principal = request.getUserPrincipal();  
    String remoteUser = principal.getName();  
  
    // to get remote user using getRemoteUser()  
    remoteUser = request.getRemoteUser();  
  
    // to check if remote user is granted Manager role, using isUserInRole  
    boolean isMgr = request.isUserInRole("Manager");  
  
}
```

J2EE Security defines three APIs that can be used by the Web developer. These are similar to the APIs used by EJB developer, but not exactly the same.

The **isUserInRole()** method is used to determine if the authenticated caller is in a specific Role defined by the application. This lets you decide that a certain part of the code is to be allowed to run only for a user in a specific Role.

The **getUserPrincipal()** method is used to return the Principal of the authenticated caller.

The **getRemoteUser()** method is used to return the remote web client user.

The example on this page shows how you can use these APIs.

Types of authentication for Web applications

- **Basic**
 - ▶ Application server sends back a 501 challenge to the Web client (browser) allowing the client to pop up user ID, password dialog to the client
- **Form based**
 - ▶ Allows Web developer to provide a custom form login for the authentication challenge
- **Client certificate**
 - ▶ The client certificate is sent to the Application server using SSL secured connection



There are three types of authentication that a Web client can perform to access the Web applications. It is the assembler that specifies the authentication method that is to be used by the Web client.

This information is stored in the Web module deployment descriptor and is used by the Web container to challenge the Web client for authentication.

Changing identity: Run-As

- The Web application Servlet or JSP has ability to change identity when calling downstream processes or EJBs
 - This is similar to the function provided in EJB methods
- This is called “Run-As” identity
- The following are the 2 “Run-As” options

Run-As options	Description
Client Identity	<ul style="list-style-type: none"> ▪ Bean takes on the same identity as the caller
Another Specified Role	<ul style="list-style-type: none"> ▪ Bean takes on identity of a specified user within the specified role ▪ The specified role is part of the deployment descriptor and performed by the assembler ▪ The specific user in the “Run-As” role is usually specified at deploy time

Run as provides a means to change the identity of the caller from a servlet or JSP when calling a downstream EJB.

An example is where a client cannot directly call a downstream EJB, but can call a servlet, which can then call a downstream EJB by changing its identity.

There are two Run as options, as shown in the table. The downstream EJB can be called with the same identity as the original Web client identity that called the Servlet, or the identity can be changed by the servlet assembler by providing run as as another specified role.

Application security tasks and roles

Tasks	Role	Tools used	Files modified
Define J2EE Security Roles	Assembler	Rational Tools, AST	Application Deployment Descriptor, application.xml
Security check using programmatic API	Developer	Rational Tools, AST	Java code
Specifying Security permission or constraints	Assembler	Rational Tools, AST	ejb-jar.xml web.xml
Specify Security Role Reference	Assembler	Rational Tools, AST	Module level IBM Binding files: ibm-ejb-jar-bnd.xmi lbn-web-bnd.xmi
Specify Security Role binding to users, groups or both	Administrator	Application Server (production) or Rational tool (dev.), AST	Server security.xml file (production) or ibm-application-bnd.xml (for development) or JACC provider
Define EJBROLEs and grant users/groups access	z/OS Security Administrator	SAF based security product	SAF based database
Specifying Authentication type	Administrator	Application Server	Server security.xml file

This table provides a summary of different tasks associated with J2EE Security Roles, the role that typically performs the task, the tools used to perform the task, and the files that are modified.

Section

Summary

The next slide will provide a summary of this presentation.

Summary

- Using J2EE Application level security, users can provide access control to who can access the application components
 - ▶ Method level for EJBs, URI level for Web components
 - ▶ Can be defined at a programmatic or declarative level
- J2EE Security roles provide a developer an abstract way of specifying the authorization



In summary, this presentation has focused on J2EE Security roles and how they are used to authorize the J2EE Application artifacts, namely the EJB and Web applications.

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.

© Copyright International Business Machines Corporation 2004, 2006. All rights reserved.

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.