

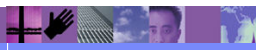
Objectives

- **You will learn to manage applications**
 - Install an application
 - Update an application
- **You will understand the application repository**
- **You will learn about the classloader**

Agenda

- **Application Management**
 - Application Installation
 - using wsadmin
 - using Admin Console
 - Application update
 - using wsadmin
 - using Admin Console
- **Configuration and Application Repository Overview**
- **Classloader**
- **Summary**

Application Management



Concepts in Application Management

▪ Application Installation

- Common install architecture across all WebSphere editions
 - In WebSphere 4.x there are various flavors of installation process.

▪ Application Distribution

- transfer of application configuration and business logic to servers that application runs on in multi-node environment
 - In WebSphere 4.x application binaries are not distributed to individual nodes as a part of application installation

· Application Installation - involves validation of application archive, collection of configuration information from the end-user and finally registration of application in the WebSphere cell. The driving force behind the application installation design in this release is the requirement to have a common install architecture across all WebSphere editions (Application Server, Network Deployment, and WebSphere Enterprise). In WebSphere 4.x there are various flavors of installation process. The Single Server edition (AEs) has a GUI-based install tool in its Web-based administrative client. It also has a text-based SEAppInstaller. The Advanced Edition (AE) has an App Install Wizard in its Swing-based administrative console. The scripting interface provided by wscp also has application installation ability. These various implementations do not share a common design and hence differ in their feature offering and end-user experience.

· Application Distribution - deals with transferring application configuration information and business logic to all the servers that the application runs on in a multi-node environment. The application should be ready to run once it is installed on the WebSphere cell. In WebSphere 4.x the administrator was expected to copy and expand the application binaries on all the nodes that the application was installed on before the application could run. Actions such as moving a module to a server / server group or adding a clone to a server group could not be completed successfully without user intervention. In WebSphere 5.0, the task of application distribution in all such cases is automated.

Concepts in Application Management (cont)

- **Operational Control**
 - start/stop application
 - view/edit configuration information
- **Application Reinstall/upgrade**
- **Application Monitoring**

- Operational Control - includes start/stop of an application, view/edit configuration information etc.

- Application Reinstall / Upgrade - allows administrators to configure and install newer versions of the application with or without interruption in service.

- Application Monitoring - tracks the status of individual application components.

Application Management: Tools

▪ **Assembly Tools**

- Application Assembly Tool (AAT)
 - Support for J2EE 1.3
 - Usability enhancements
- WebSphere Studio Family of Products
 - Full IDE for developing and unit testing J2EE Enterprise applications

▪ **Enterprise Application Management Tools**

- Browser based Admin Console
 - For WebSphere, attach the browser directly to the App Server
 - For Multi-node topology, attach the browser to the Cell

▪ **Command line tools**

- wsadmin

Browser based admin console is struts based.

Install New application using Admin Console

Preparing for the application installation

Specify the EAR/WAR/JAR module to upload and install.

Path: Browse the local machine or a remote server:

Local path:

Server path:

Context Root: Used only for standalone Web modules (*.war)

[Step 2](#) Provide JNDI Names for Beans

Step 3 Provide default datasource mapping for mod

Step 4 Map datasources for all 2.0 CMP beans

[Step 5](#) Map EJB references to beans

[Step 6](#) Map virtual hosts for web modules

[Step 7](#) Map modules to application servers

[Step 8](#) Ensure all unprotected 2.0 methods have the correct level of protection

[Step 9](#) Summary

Generate Default Bindings:

Prefixes:	<input type="radio"/> Do not specify unique prefix for beans <input checked="" type="radio"/> Specify Prefix: <input type="text" value="ejb/MyBank"/>
Override:	<input checked="" type="radio"/> Do not override existing bindings <input type="radio"/> Override existing bindings
EJB 1.1 CMP bindings:	<input checked="" type="radio"/> Do not default bindings for EJB 1.1 CMPs <input type="radio"/> Default bindings for EJB 1.1 CMPs: JNDI name: <input type="text"/> username: <input type="text"/> password: <input type="text"/> verify password: <input type="text"/>
Connection Factory Bindings:	<input type="radio"/> Do not default connection factory bindings <input checked="" type="radio"/> Default connection factory bindings: JNDI name: <input type="text" value="eis/jdbc/MyBank_CMP"/> Resource authorization: <input type="text" value="Per connection factory"/>

Go to appropriate step and enter information

8
WebSphere Version 5.0 Application Administration and Classloader
© 2002, 2003 IBM Corporation

Use the Install New Application wizard to install an application (EAR file) or module (JAR or WAR file).

Follow the steps in the Install New Application wizard to install an application or module. You must complete some or all of the steps, depending on whether you are installing an application, EJB module, or Web module.

Path - The fully qualified path to the .ear, .jar, or .war file for the enterprise application.

Use Local path if the console and application files are on the same machine (whether or not the server is on that machine, too).

Use Server path if the application files already reside on the server machine, and the console is being run from a remote machine.

During application installation, application files are typically uploaded from a client machine containing the administrative console to the server machine, where they are deployed. In such cases, the Web browser running the administrative console is used to select EAR, WAR, or JAR modules to upload to the server machine.

In some cases, however, the application files will already reside on the file system of the machine running the application server. To have the application server install these files (bypassing the upload stage), use the Server path option.

Installing a new application using wsadmin

- **wsadmin can be used to install a new application**
- **Launch wsadmin by running the wsadmin.bat file**
- **Interactive install**
 - wsadmin>\$AdminApp installInteractive
c:/WebSphere/AppServer/installableApps/sample.ear
- **Non-interactive invocation**
 - wsadmin>\$AdminApp install
c:/WebSphere/AppServer/installableApps/sample.ear

Installing a new application using wsadmin

- **Use the -conntype NONE option, with the server stopped**
 - wsadmin>\$AdminApp installInteractive
d:/WebSphere/AppServer/installableApps/sample.ear
- **To list all installed applications (you need the server up to do this no local mode)**
 - wsadmin>\$AdminApp list
- **Invoking help text**
 - wsadmin>\$Help help
 - wsadmin>\$AdminApp help
 - wsadmin>\$AdminApp help installInteractive

Application Upgrade - Flow

- **The application needs an upgrade.**
- **Software developers code and unit test the new software.**
- **Assembler assembles the changes into the EAR file**
 - Application Assembly Tool may be used to re-package the EAR file
 - Start assembly tool by running assembly.bat
- **Administrator tests new software in a testing configuration (stand-alone mode).**
- **The new software is ready to move into production.**
 - Administrator performs re-install steps from Admin Console

Application Update

- **From end-user perspective, the update process will be very similar to install**
 - It will additionally take the name of the existing app that is to be upgraded
 - **No continuous availability of the application.**
- The reinstall process:-**
- Stops the current application if it is running
 - uninstalls the current application
 - installs the new application

Application upgrade performs entire application upgrade without introspecting on its individual components.

Application Re-install using Admin Console:

Select the Application in the Admin Console and click Update.

The panels in these steps will be prepopulated with bindings and other data from previous version wherever applicable. The bindings that the user enters at re-install time overrides the bindings from the currently installed version. Empty bindings / deployment info are filled with bindings from the currently installed version. Default Bindings Generator can be used to populate bindings for new artifacts in the new version.

Application Update (continued)

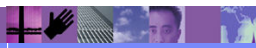
▪ Using Admin Console

- The user interface will guide the users through the same install steps for reinstall
- Applications->Enterprise Applications. Select the Application and click Update

▪ Using wsadmin

- use update option in AdminApp
- example: `$AdminApp install C:/MyBanknew.ear { -appname MyBankold -update -update.ignore.new}`
 - -update option specifies that the application named MyBankold is to be redeployed using the supplied EAR file
 - -update.ignore.new option specifies that the bindings from the new version are ignored

Application Repository Overview



Repository - Base Application Server

DDs = Deployment Descriptors

- deployments folder contains only the DDs
- Contains the original EAR
 - Original DDs
 - All application binaries

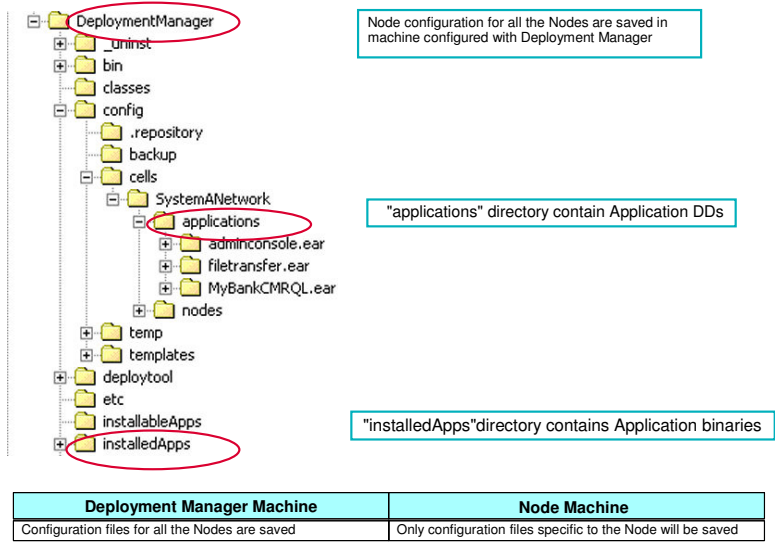
- DDs in config dir. contain all updates provided after installation of the Application EAR Example: Could have provided a different JNDI name for an EJB or resource
- DDs in InstalledApps dir. are the original DDs in the EAR file (unmodified DDs)
- App Server takes the modified DDs (from config dir.) to start the application

15 WebSphere Version 5.0 Application Administration and Classloader © 2002, 2003 IBM Corporation

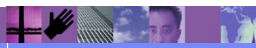
The config/applications subdirectory, holds a subdirectory for each application deployed in the cell. The names of the applications subdirectories match the names of the deployed applications. Each deployed application subdirectory holds a deployment.xml file that contains configuration data on the application deployment. Each subdirectory also holds a META-INF subdirectory that holds a J2EE application deployment descriptor file as well as IBM deployment extensions files and bindings files. Deployed application subdirectories also hold subdirectories for all .war and entity bean .jar files in the application. Binary files such as .jar files are also part of the configuration structure.

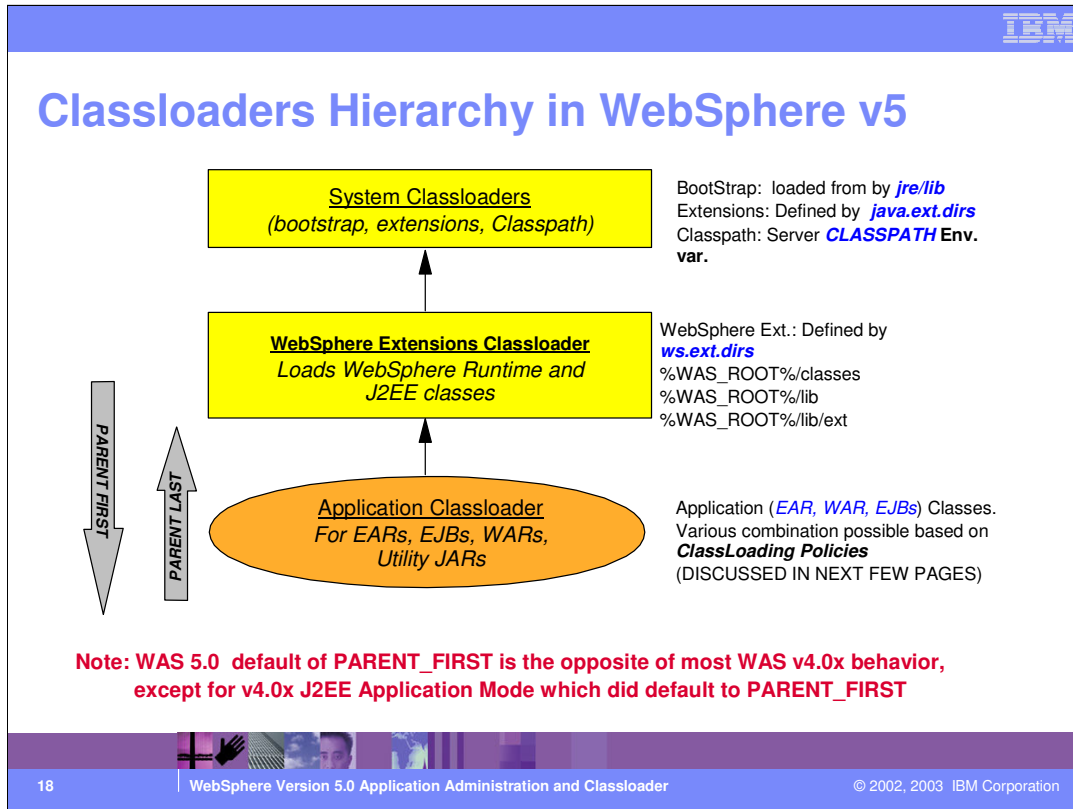
Keeping modified Deployment Descriptors at the config level allows System Management. to make Application update more automatic, without much user intervention like reentering the JNDI names, etc.

Repository – Network Deployment



Classloaders





Classloaders are organized in hierarchy. This means that a child classloader can delegate class finding and loading to its parent, should it fail to load a class.

The root of the hierarchy is occupied by the system classloaders (in particular, by the Classpath classloader)

The leaves are occupied by the Web Module classloaders.

WebSphere App Server 5.0 allows you to configure classloaders with various degrees of granularity as we will discuss in a subsequent chart.

The bootstrap classloader uses the bootclasspath (typically classes in jre/lib) to find and load classes. The extensions classloader uses the system property java.ext.dirs (typically jre/lib/ext) to find and load classes and the CLASSPATH classloader uses the CLASSPATH environment variable to find and load classes. This CLASSPATH classloader contains the J2EE APIs of the WebSphere Application Server product (inside j2ee.jar). Because the J2EE APIs are in this classloader, you can add libraries that depend on J2EE APIs to the classpath system property to extend a server's classpath. However, a preferred method of extending a server's classpath is to add a shared library.

The extensions classloader loads the WebSphere run time and all user code and J2EE classes that are required at run time. The extensions classloader uses a ws.ext.dirs system property to determine the paths used to load classes. The ws.ext.dirs system property is derived from the WS_EXT_DIRS environment variable set in the setupCmdLine script file. Each directory in the ws.ext.dirs classpath is added to the classpath and every JAR file or ZIP file in the directory is added to the classpath.

One or more application module classloaders that load elements of enterprise applications running in the server.

The application elements can be Web modules, EJB modules resource adapters, and dependency JAR files. Application classloaders follow J2EE class loading rules to load classes and JAR files from an enterprise application. The WebSphere run-time enables

Application Classloader policy

- **Determines how the applications share classloader**
- **At the Application Server level - choose SINGLE or MULTIPLE**
 - SINGLE means the EJB, RAR modules and dependent JARs for all the EARs are loaded by one classloader called the Application classloader
 - MULTIPLE means the EJB, RAR modules and dependent JARs for each EAR are loaded by its own classloader
 - Whether the WAR is loaded by this Application classloader is dictated by the WAR classloader policy

WAR Classloader policy

- **Determines how the WAR modules are loaded per Application**
- **At the Enterprise Application level, choose APPLICATION or MODULE**
 - APPLICATION: all the Web modules in the application EAR use the Application classloader (dictated by the Application Classloader Policy)
 - MODULE: every WAR uses its own classloader, different than the Application Classloader
 - Selection can be made at application install time

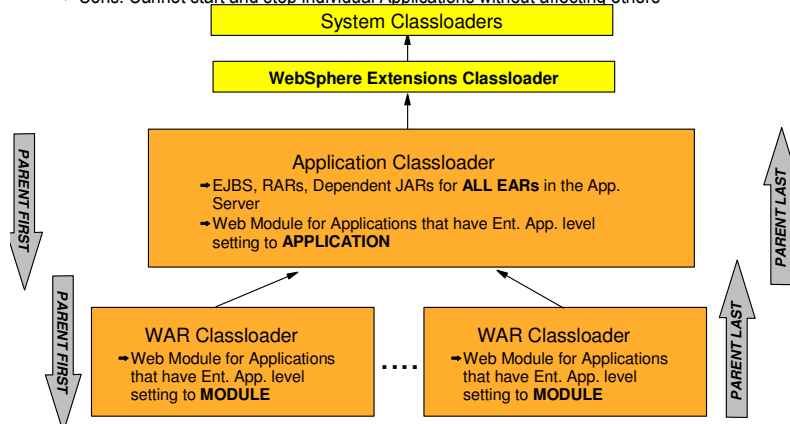
ClassLoader Mode

- **Set for Application Classloader policy and WAR Classloader policy**
- **PARENT_FIRST - default**
 - Search the immediate parent first and then its policy would determine if that got search first or its parent
- **PARENT_LAST**
 - Tries to find and load the class from its own classloader and if the class was not found, it delegates to its immediate parent classloader and then the immediate parent's classloader policy would take control

Application Classloader Policy - SINGLE

- Application Classloader policy of **SINGLE** and WAR classloader policy of **APPLICATION** or **MODULE**

- Pros: Each Application EJBs, RARs, Dep. JARs can reference other classes in other Applications
- Cons: Cannot start and stop individual Applications without affecting others



22

WebSphere Version 5.0 Application Administration and Classloader

© 2002, 2003 IBM Corporation

Classloaders are organized in hierarchy. This means that a child classloader can delegate class finding and loading to its parent, should it fail to load a class.

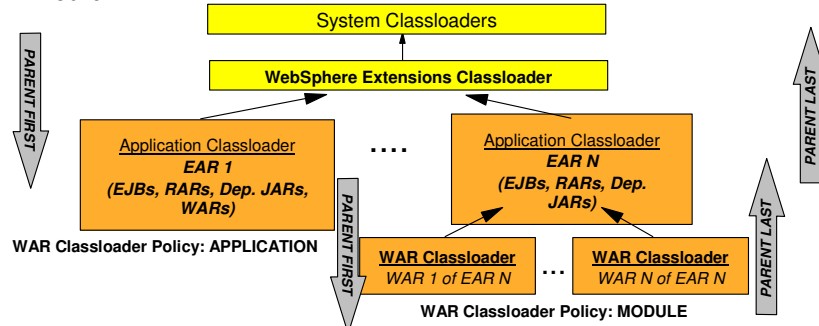
The root of the hierarchy is occupied by the system classloaders (in particular, by the Classpath classloader)

The leaves are occupied by the Web Module classloaders.

WebSphere Application Server 5.0 allows you to configure classloaders with various degrees of granularity as we will discuss in a subsequent chart.

Application Classloader Policy - MULTIPLE

- Application Classloader policy of **MULTIPLE**
- WAR classloader policy of **APPLICATION** or **MODULE**
- Pros:
 - ▶ Can restart each application without affecting others
 - ▶ Classes within each EAR can reference all the classes with the same EAR even if in different modules of the EAR (classes in WAR may or maynot be referenced, depending on Ent. Application level policy)
- Cons: Classes within one EAR cannot reference classes in another EAR



Example 1

- Application classloader policy: **SINGLE**

- Application 1

Module: EJB1.jar

Module: WAR1.war

MANIFEST Class-Path: Dependency1.jar

WAR Classloader Policy = **APPLICATION**

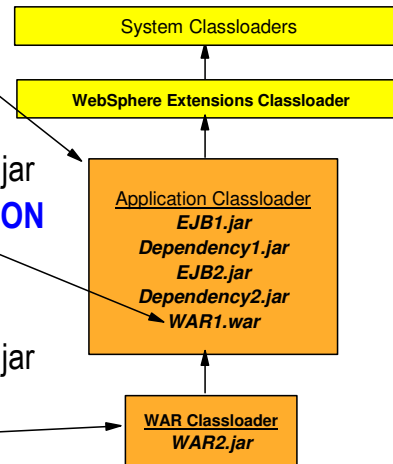
- Application 2

Module: EJB2.jar

MANIFEST Class-Path: Dependency2.jar

Module: WAR2.war

WAR Classloader Policy = **MODULE**



Example 2

- Application classloader policy: **MULTIPLE**

- Application 1

Module: EJB1.jar

Module: WAR1.war

MANIFEST Class-Path: Dependency1.jar

WAR Classloader Policy

= **APPLICATION**

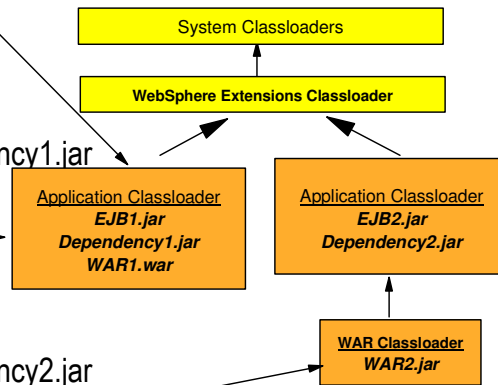
- Application 2


Module: EJB2.jar

MANIFEST Class-Path: Dependency2.jar

Module: WAR2.war

WAR Classloader Policy = **MODULE**





v4 vs. v5 Classloading Policies - Comparison

	v4	v5
	4.0 Classloader Policy	v5 Application Classloader Policy
		v5 WAR Classloader Policy
SERVER	SINGLE	APPLICATION
COMPATIBILITY	SINGLE	MODULE
APPLICATION	MULTIPLE	APPLICATION
MODULE	N/A	N/A
J2EE Application Mode (4.0.3)	MULTIPLE	MODULE

26 | WebSphere Version 5.0 Application Administration and Classloader | © 2002, 2003 IBM Corporation

SERVER

All applications and modules installed on an App Server share the same classloader
 SINGLE/APPLICATION

MODULE

Each J2EE module has its own classloader
 No exact equivalent in Version 5

APPLICATION

Classloader scoped to the EAR file
 Loads EJBs and WARs
 Equivalent to MULTIPLE/APPLICATION

COMPATIBILITY

EJBs and WARs have separate classloaders
 One classloader per server for all the EJB and one classloader per WAR module
 SINGLE/MODULE

Reloading Classes in Version 5

- **Web Modules can be reloaded independently**
 - Stop/Restart the Web Module
 - Also, can configure Web Modules to restart automatically if contents changes
- **EJB Modules**
 - Cannot be stopped/restarted independently in Version 5
 - The entire application has to be restarted
- **Best practice: stop/restart the Enterprise App**
 - This will ensure that all the dependencies are also reloaded

These are some of the reloading options in Version 5.

Notice that EJB modules cannot be started or stopped independently in V5 - unlike Version 4.

Application Specific Libraries

- **Can be defined for libraries of code that needs to be shared across multiple applications within a server - Support for java and native libraries.**
 - Environment -> Shared Libraries -> New
- **On the Enterprise Application, specify the shared libraries to be used**
 - Enterprise Applications -> Application -> Libraries -> Add (add one of the defined Shared Libraries)

In WebSphere v4, if you had JARs that needed to be shared by more than one application, typically you would put it in the %WAS_ROOT%/lib/app directory (this is already defined in the ws.ext.dirs for WebSphere v4). The drawback is that the JARs in the %WAS_ROOT%/lib/app directory are exposed to all the applications.

Shared Libraries is a better mechanism, where only applications that need the JARs use them. Other applications do not get affected by the shared libraries.

Shared Libraries

Environment

[Update Web Server Plugin](#)

[Virtual Hosts](#)

[Manage WebSphere Variables](#)

[Shared Libraries](#)

Naming

DEFINE New Shared Libraries

Configuration	
General Properties	
Name	* MySharedCode
Description	
Classpath	C:\MyCode*
Native Library Path	

Can be a Single JAR file or DIRECTORY that contains multiple JAR files

Native Library used by the Application - DLLs, lib*.so, etc.

Summary

- **Application Management uses JMX-based System management component to manage applications within WebSphere**
- **The Admin Clients wsadmin and Admin Console can be used in the Application Management tasks**
- **Classloaders are part of the Java virtual machine (JVM) and are responsible for finding and loading class files:**
 - System Classloaders
 - WebSphere Extensions Classloader
 - Application Classloaders

Trademarks and Disclaimers

© Copyright International Business Machines Corporation 1994-2003. All rights reserved. References in this document to IBM products or services do not imply that IBM intends to make them available in every country. The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

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

Lotus, Domino, Freelance Graphics, and Word Pro are trademarks of Lotus Development Corporation and/or IBM Corporation in the United States and/or other countries.

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 trademarks of Microsoft Corporation in the United States, other countries, or both. 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. Other company, product and service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information in this presentation concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information in this presentation addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. 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 2003. 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.