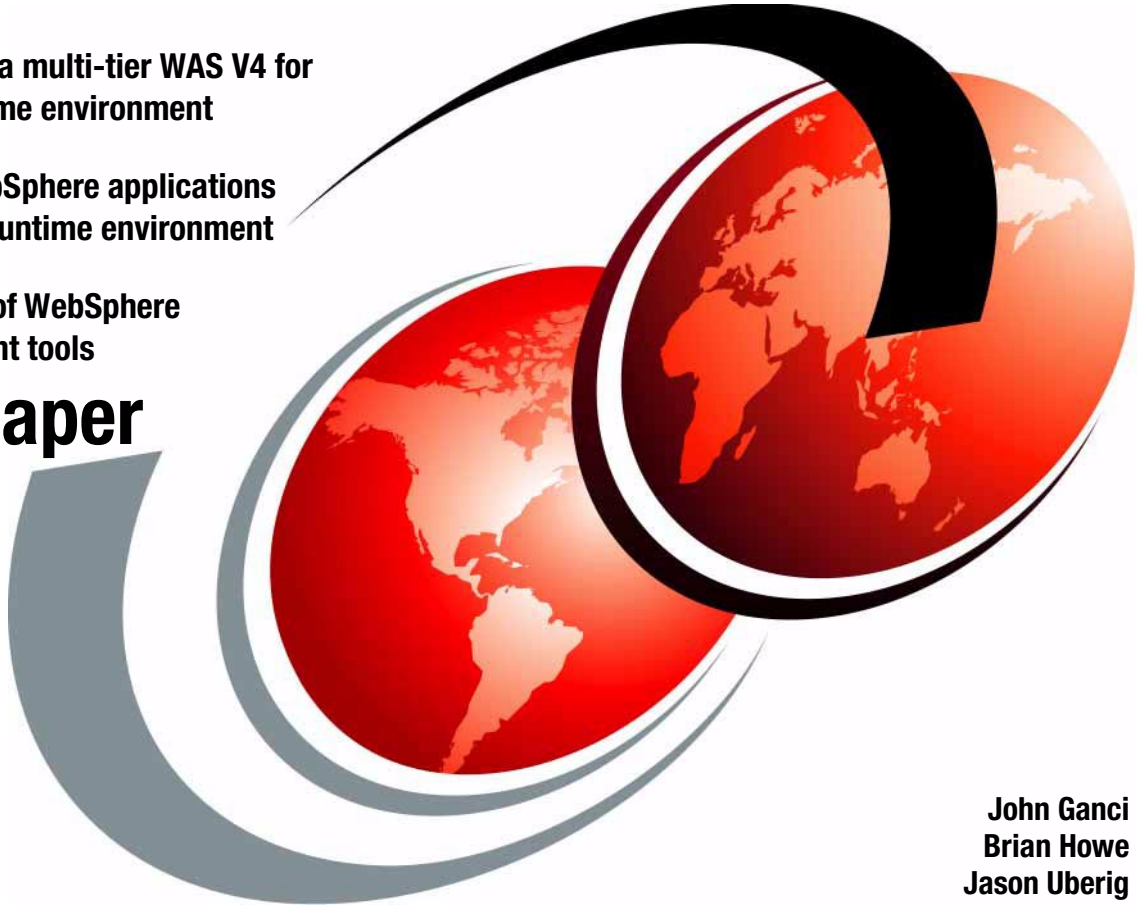
WebSphere Application Server V4 for Linux

Implementation and Deployment Guide

- Implement a multi-tier WAS V4 for Linux runtime environment
- Deploy WebSphere applications to a Linux runtime environment
- The future of WebSphere development tools

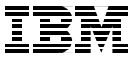
Redpaper



John Ganci
Brian Howe
Jason Uberig

Redbooks

ibm.com/redbooks



International Technical Support Organization

WebSphere Application Server V4 for Linux Implementation and Deployment Guide

October 2001

Take Note! Before using this information and the product it supports, be sure to read the general information in “Special notices” on page 137.

First Edition (October 2001)

This edition applies to WebSphere Application Server V4.0.1, Advanced Edition for Linux, for use on the RedHat Linux V7.1 operating system.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 662
P.O. Box 12195
Research Triangle Park, NC 27709-2195

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Preface vii
The team that wrote this Redpaper vii
Special notice viii
IBM trademarks ix
Comments welcome ix

Chapter 1. Introduction 1
1.1 IBM Framework for e-business 2
1.2 Structure of Redpaper 3

Chapter 2. WAS V4 for Linux single-tier runtime environment 5
2.1 Planning 6
2.1.1 Hardware and software prerequisites 6
2.1.2 Hardware used in our test environment 8
2.1.3 Software used in our test environment 8
2.1.4 Runtime environment 9
2.2 Red Hat Linux installation 11
2.2.1 Linux kernel 12
2.2.2 File systems 12
2.2.3 Linux software package requirements 13
2.3 Install the DB2 Server 13
2.3.1 Pre-requisite Linux packages for DB2 14
2.3.2 Pre-installation tasks 15
2.3.3 Install the DB2 Server 16
2.3.4 Verify the DB2 Server installation 20
2.3.5 Configure the DB2 Server 23
2.3.6 Create the WAS repository database 26
2.4 Install the WebSphere Application Server 28
2.4.1 Pre-installation tasks 28
2.4.2 Install the WebSphere Application Server 30
2.5 Configure and verify the IBM HTTP Server 37
2.5.1 Create HTTP Server admin account 38
2.5.2 Create an HTTP Administration Server runtime user account 39
2.5.3 Create an HTTP Server runtime user account 40
2.5.4 Configure the HTTP Server for SSL 40
2.5.5 Configuring the ServerName 42
2.5.6 Restart the IBM HTTP Server 43
2.5.7 Verify the IBM HTTP Server 43

2.6 Configure and verify WAS 45
2.6.1 Verify the WebSphere Application Server installation 46
2.6.2 Configure the WebSphere Application Server 47
2.6.3 Configure WAS HTTP transport for SSL 53

Chapter 3. WAS V4 for Linux 2-tier runtime environment 61
3.1 Planning for a WAS V4 for Linux 2-tier runtime 63
3.1.1 Overview 63
3.1.2 Hardware and software prerequisites 64
3.1.3 Hardware used within our test environment 64
3.1.4 Software used within our test environment 64
3.1.5 Install Red Hat Linux 65
3.2 Implementing the Remote Web Server 2-tier 65
3.2.1 Install the DB2 Server 65
3.2.2 Install IBM WebSphere Application Server V4.0.1 66
3.2.3 Install IBM HTTP Server V1.3.19 66
3.2.4 Update the WebSphere plug-in file 68
3.2.5 Verify WebSphere plug-in configuration 69
3.3 Implementing the Remote Database Server 2-tier 69
3.3.1 Install the DB2 Server 70
3.3.2 Install the DB2 Client 70
3.3.3 Configure the DB2 Client 75
3.3.4 Install and configure the WebSphere Application Server 78
3.3.5 Configure and verify IBM HTTP Server 78

Chapter 4. WAS V4 for Linux 3-tier runtime environment 79
4.1 Planning for a WAS V4 for Linux 3-tier runtime 80
4.1.1 Overview 80
4.1.2 Hardware and software prerequisites 80
4.1.3 Hardware used within our test environment 81
4.1.4 Software used within our test environment 82
4.1.5 Install Red Hat Linux 82
4.2 Install the DB2 Server 82
4.3 Install the DB2 Client 82
4.4 Install the WebSphere Application Server 82
4.5 Install the IBM HTTP Server 83

Chapter 5. Deploy applications to a WAS V4 for Linux runtime 85
5.1 PiggyBank sample application overview 86
5.1.1 Download the sample application zip file 86
5.1.2 Contents of the sample application zip file 86
5.2 Deployment considerations 88
5.2.1 Deployment assets 88
5.2.2 General deployment considerations 89

5.2.3 Deployment stages 90
5.2.4 Deployment nodes and tiers 91
5.2.5 Clustered environments 92
5.2.6 Security considerations 92
5.2.7 File transfer methods for deployment 92
5.2.8 File system considerations 93
5.3 Set up Linux runtime for deployment 94
5.4 Deploying VisualAge for Java assets 96
5.4.1 VisualAge for Java assets 96
5.4.2 VisualAge for Java deployment considerations 97
5.4.3 Deploying PiggyBank VAJ assets 97
5.5 Deploying WebSphere Studio assets 101
5.5.1 WebSphere Studio assets 101
5.5.2 WebSphere Studio deployment considerations 101
5.5.3 Deploying PiggyBank WebSphere Studio assets 102
5.6 Deploying the enterprise application 109
5.6.1 Configure the WAR file 109
5.6.2 Create the EJB references 111
5.6.3 Create and assemble the enterprise application 112
5.6.4 Install the enterprise application using the Admin Console 114
5.7 Running the deployed enterprise application 116

Chapter 6. The future of WebSphere development tools for Linux 117
6.1 WebSphere Studio Workbench 119
6.1.1 Why a new development platform? 120
6.1.2 WebSphere Studio Site and Application Developer 121
6.1.3 Planned platform support - Linux and Windows 124
6.2 WebSphere Studio Site Developer 124
6.2.1 Web development environment 125
6.2.2 XML tools 125
6.2.3 Web Services development environment 126
6.2.4 The evolution of WebSphere Studio 127
6.3 WebSphere Studio Application Developer 127
6.3.1 Enterprise JavaBeans development environment 128
6.3.2 Database tools 128
6.3.3 Monitoring and profiling tools 129
6.3.4 The evolution of VisualAge for Java 129
6.3.5 PiggyBank in WebSphere Studio Application Developer 130
6.3.6 Summary 134

Special notices 137

Preface

The current IBM development tool set for WebSphere (VisualAge for Java, WebSphere Studio) is available for the Windows NT and Windows 2000 platforms. WebSphere applications developed on the Windows platform can be deployed to a number of WebSphere Application Server V4 operating system platforms, including Linux.

This Redpaper provides detailed procedures for implementing multi-tier runtime environments for WebSphere Application Server V4.0.1, Advanced Edition for Linux. In addition, we describe in detail the steps necessary to deploy an enterprise application developed in VisualAge for Java and WebSphere Studio to a WebSphere Application Server V4.0.1, Advanced Edition for Linux runtime environment.

The future IBM strategy for development tools provides developers a complete set of WebSphere application development tools for both Windows and Linux. The upcoming release of the development tools will be packaged as the WebSphere Studio Application Developer and WebSphere Studio Site Developer.

The team that wrote this Redpaper

This Redpaper was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

John Ganci is Senior Software Engineer, WebSphere Specialist at the IBM ITSO, Raleigh Center. John has 14 years of experience in product and application design, development, system testing, and consulting. His areas of expertise include e-commerce, personalization, pervasive computing, and Java programming. Before joining the ITSO, he developed e-commerce sites for IBM.

Brian Howe is a Software Engineer with the IBM Linux Integration Center in Austin, TX. He has four years of experience designing Linux, Java, and e-business solutions. He holds a Bachelor's degree in Computer Engineering from the University of Oklahoma. His area of expertise includes Java and e-business solutions on both Linux and Windows platforms.

Jason Uberig is a Senior I/T Specialist with IBM Global Services in Canada. He has over 10 years of experience in the I/T field. He holds a Bachelor's degree in Electrical Engineering from The University of Western Ontario. His areas of expertise include e-infrastructure, UNIX integration and implementation, and systems management.

Thanks to the following people for their contributions to this project:

Jakob Carstensen, IBM Linux Solutions Marketing

Jared Jurkiewicz, IBM Raleigh

Steve Francisco, IBM Canada

John Kellerman, IBM Raleigh

Duffy Fron, IBM Linux Solutions Marketing

Jeffrey Heyward, IBM Australia

Mark Endrei, IBM ITSO, Raleigh Center

Margaret Ticknor, IBM ITSO, Raleigh Center

Gail Christensen, IBM ITSO, Raleigh Center

Linda Robinson, IBM ITSO, Raleigh Center

Special notice

This publication is intended to help I/T architects, I/T specialists, developers and consultants who implement and deploy WebSphere applications in a WebSphere Application Server V4.0.1, Advanced Edition for Linux runtime environment. The information in this publication is not intended as the specification of any programming interfaces that are provided by WebSphere Application Server V4.0.1, Advanced Edition for Linux.

See the PUBLICATIONS section of the IBM Programming Announcement for WebSphere Application Server V4.0.1, Advanced Edition for Linux for more information about what publications are considered to be product documentation.

IBM trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

e (logo)® 
IBM ®
AIX
DB2
DB2 Universal Database
DFS
Home Director
Netfinity
NetVista
OS/390
OS/400
Redbooks
Redbooks Logo 
S/390
SP
VisualAge
WebSphere
400
Lotus
Domino

Comments welcome

Your comments are important to us!

We want our Redpapers to be as helpful as possible. Send us your comments about this Redpaper or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to the address on page ii.

1

Introduction

When developing enterprise Web applications, it is critical to have a flexible, scalable, and cost-effective architecture that is highly reliable. Medium-to-large sized businesses are often faced with the additional complexity of having multiple operating system platforms as part of their e-infrastructure, and need to integrate many technologies within their enterprise applications.

To address these e-infrastructure and application needs, the IBM Framework for e-business has been developed with the underlying principles of industry-standard technologies (Java, Linux, and XML), proven methodologies (Patterns for e-business), and leadership products (IBM WebSphere middleware, DB2, Lotus, and Tivoli).

IBM has invested tremendous a amount of resources in developing IBM software and solutions for the Linux operating system platform consistent with the goals of the IBM Framework for e-business.

This Redpaper demonstrates how to implement WebSphere Application Server V4.0.1, Advanced Edition for Linux solutions, and provides a step-by-step sample enterprise application deployment from a Windows development environment to a Linux runtime environment.

1.1 IBM Framework for e-business

The IBM Framework for e-business has been developed with the following underlying principles:

- Industry-standard technologies

  The Framework is based on multi-platform, multi-vendor standards such as Java, XML, HTML, and Linux. It includes the client, application server, network, data and infrastructure standards that make it possible for a client to access services and data anywhere in the network. This model simplifies application development and deployment by enabling developers to write an application once and deploy it to any supported platform.

- Proven methodologies

  When building on existing I/T investments and mission-critical applications, it is highly beneficial to reuse proven methodologies for quick action to capture business opportunities and respond to market challenges.

  The methodologies are based on the experiences of IBM architects, specialists, and consultants and tested solutions. In addition, the proven methodologies are further researched, refined and documented by the IBM Patterns for e-business to make it easier to apply the technologies, standards, and products of the IBM Framework for e-business.

  The Patterns for e-business leverage the Framework by providing guidelines for selecting the Business pattern, Application pattern, Runtime pattern, product mapping, design, and development.

- Leadership products

  IBM WebSphere middleware, DB2 data management software, Lotus collaboration software, and Tivoli management software provide industry-leading solutions and a solid foundation for e-business applications. The IBM software products can be implemented on a wide range of servers with scalability in mind as the needs of your e-business grow.

The IBM Framework for e-business, often referred to as the Framework, is at the core of IBM's e-business strategy to help the customers build, run and manage e-business applications (see Figure 1-1).

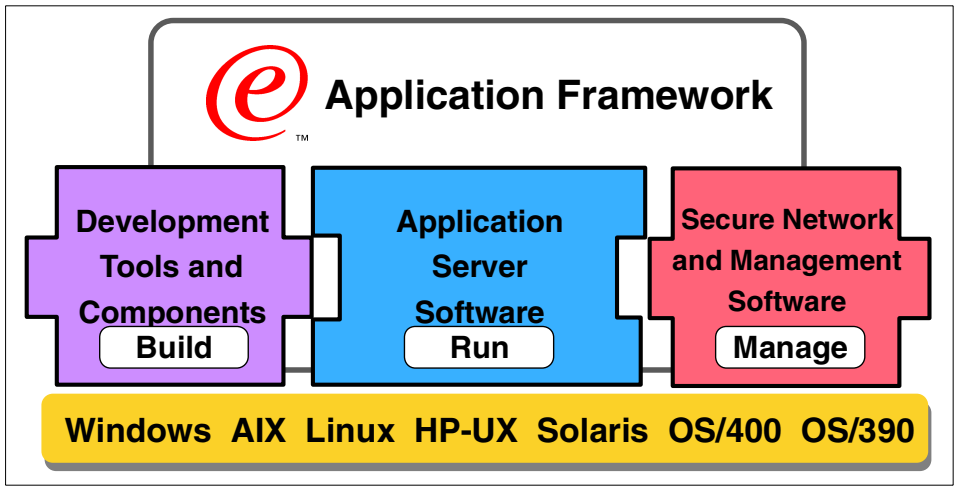


Figure 1-1 IBM Framework for e-business

The current IBM tools strategy provides a Windows-only development environment, which includes IBM VisualAge for Java and WebSphere Studio. Applications developed in the Windows development environment can be deployed to the wide-ranging WebSphere-supported operating systems, including Linux, Windows, AIX, HP-UX, Solaris, OS/400, and OS/390.

Additional information on the IBM Framework for e-business can be found at:

`http://www.ibm.com/software/ebusiness`

In the near future, IBM will unveil a new tools strategy that will include a development environment that will run on Windows and Linux. This strategy will take hold with the introduction of the WebSphere Studio Application Developer and WebSphere Studio Site developer.

Additional information on IBM Software for Linux can be found at:

`http://www.ibm.com/linux/software`

1.2 Structure of Redpaper

This Redpaper has been designed to help architects, developers, and I/T specialists implement and deploy applications for a stand-alone or multi-tiered WebSphere Application Server V4.0.1, Advanced Edition for Linux runtime environment.

The objectives of this Redpaper are threefold:

- Runtime environment implementation

  Describe in detail how to install a stand-alone, 2-tier, and 3-tier WAS V4 runtime for Linux.

  - Chapter 2, “WAS V4 for Linux single-tier runtime environment” on page 5
  - Chapter 3, “WAS V4 for Linux 2-tier runtime environment” on page 61
  - Chapter 4, “WAS V4 for Linux 3-tier runtime environment” on page 79

- Enterprise application deployment

  Describe how application assets developed in IBM VisualAge for Java and IBM WebSphere Studio for Windows apply to a multi-tiered test and deployed to a WebSphere Application Server V4.0.1, Advanced Edition for Linux runtime environment.

  - Chapter 5, “Deploy applications to a WAS V4 for Linux runtime” on page 85

- Future of WebSphere tools for Linux

  Describe the future strategy for WebSphere tools, which includes a development platform for Windows and Linux.

  - Chapter 6, “The future of WebSphere development tools for Linux” on page 117

This redpaper is intended to complement the following Redbooks on WebSphere V4 by providing a focus on the Linux platform:

- *WebSphere V4 Advanced Edition Handbook*, SG24-6176
- *WebSphere Version 4 Application Development Handbook*, SG24-6134

2

WAS V4 for Linux single-tier runtime environment

This chapter provides procedures for installing, configuring and verifying a WebSphere Application Server V4.0.1, Advanced Edition for Linux in a single-tier runtime environment. Subsequent chapters will cover 2-tier and 3-tier runtime environments, but rely heavily on this chapter for details of the component installation.

In the single-tier environment, we will install all of the WebSphere components on the same server, as seen in Figure 2-1. This is not a recommended scenario for a production server; however, it can be useful for development, testing, or proof of concept work.

The chapter is organized into the following sections:

- Planning
- Red Hat Linux installation
- Install the DB2 Server
- Install the WebSphere Application Server
- Configure and verify the IBM HTTP Server
- Configure and verify WAS

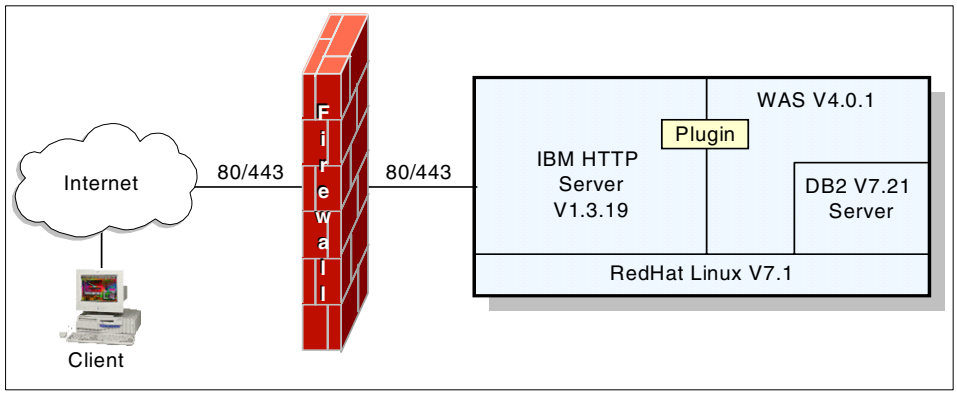


Figure 2-1 single-tier runtime environment

2.1 Planning

This section defines the hardware and software used within the WebSphere Application Server V4.0.1, Advanced Edition for Linux runtime environment single-tier scenario.

This section includes the following topics:

- Hardware and software prerequisites
- Hardware used in our test environment
- Software used in our test environment
- Runtime environment

2.1.1 Hardware and software prerequisites

The WebSphere Application Server V4.0.1, Advanced Edition for Linux has the following hardware and software requirements.

Hardware

The values in Table 2-1 show the minimum hardware resources required for base product installation of each component. The values listed *are a rough guide*. Please consult official product documentation for actual values.

Table 2-1 Hardware resources

Resource type	IBM WebSphere Application Server	IBM DB2 EE	IBM HTTP Server
Disk	150 MB	300 MB	50 MB
Virtual Memory[a]	168 MB[b]	78 MB	15 MB[c]
CPU	single[d]	single[d]	single[d]

Notes:

a. Memory values are based on RSS values. These are virtual memory requirements and may *not* reflect physical memory.

b. Linux's **ps** output shows both processes and threads. The RSS values for the WebSphere JVMs were calculated by isolating the main thread (process) for each JVM and summing the RSS values. This is since the threads share the same resources as the process.

c. Out of the box defaults for IHS. Mileage may vary depending on tuning parameters.

d. In the single-tier architecture a single CPU is sufficient; however, if any significant load is expected, additional CPUs may be desired.

Note: For further details on requirements, see the following documentation:

- IBM DB2 Universal Database

  `http://www.ibm.com/cgi-bin/db2www/data/db2/udb/winox2unix/support/v7pubs.d2w/en_main`

- IBM WebSphere Application Server

  `http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html`

Software

- Red Hat Linux V7.1 Standard Edition + prerequisites

  or

  SuSE Linux 7.1 for Intel 2.4 Kernel

- IBM WebSphere Application Server V4.0.1, Advanced Edition for Linux
- IBM DB2 Universal Database V7.2.1, Enterprise Edition for Linux
- IBM HTTP Server V1.3.19 for Linux

- IBM GSKit V5.0.3.52

Note: The above list of software is not the full list of supported versions of Linux, Web servers, or database software for the IBM WebSphere Application Server V4.0.1, Advanced Edition for Linux. Refer to the product documentation for a complete list.

This paper has been written and tested using the WebSphere Application Server V4.0.1, Advanced Edition for Linux. The WebSphere Application Server V4.0.1, Single-Server Edition stores its repository configuration in an XML file, instead of using a database server such as DB2, as the Advanced Edition. In addition, some of the enterprise application deployment tools are different between the Single-Server and Advanced Edition. Many of the procedures documented in this paper will not apply to the Single-Server edition.

2.1.2 Hardware used in our test environment

We used the following system within our single-tier environment:

- IBM NetVista (6579-A4U)
  - 866 MHz CPU
  - 512 MB RAM
  - 30 GB hard disk
  - 1 IBM Ethernet Adapter
  - 1 Integrated Intel 815e Solano Video Adapter

Note: The stock XFree86 on Linux 7.1 driver for the Intel 815 was not free of problems. On our system any activity on the system would cause the X display to jitter. The solution was to download and install the Intel driver directly from:

`http://appsr.intel.com/scripts-df/Detail_Desc.asp?ProductID=179&DwnldID=2702`

2.1.3 Software used in our test environment

We used the following software in our test environment:

- Red Hat Linux 7.1 Standard Edition + prerequisites
- IBM WebSphere Application Server V4.0.1, Advanced Edition for Linux
- IBM DB2 Universal Database V7.2.1, Enterprise Edition for Linux
- IBM HTTP Server V1.3.19 for Linux

- IBM GSKit V5.0.3.52 (included in IBM HTTP Server package).

Note: Other Web servers and Database software may be used, as documented in the *Product Installation Guide.*

Product installation directories

Table 2-2 describes variable names, default installation directories, and components. We will reference these variables throughout this document.

Table 2-2 Product installation directories

Variable	Default directory	Component
<was_install_path>	/opt/WebSphere/AppServer	WAS
<db2_install_path>	/usr/IBMdb2	DB2 Server
<http_install_path>	/opt/IBMHTTPServer	IBM HTTP Server

Installation variables

Table 2-3 describes the variable names and values used during the installation.

Table 2-3 Common installation settings

Variable	Default value	Description
<db2_instance_owner>	db2inst1	DB2 Server
<db2_instance_path>	/home/db2inst1	DB2 Server

2.1.4 Runtime environment

We used the following information to build our single-tier environment.

Network settings

The network configuration for our setup was as follows:

- Hostname: ganci5.itso.ibm.com
- IP address: 9.24.105.134
- Subnet Mask: 255.255.255.0
- Default Route: 9.24.105.1
- DNS Server: 9.24.104.108

File system layout

The values in Table 2-4 were used during the installation in our test environment. We were generous in most of the file systems to account for growth. Linux's ext2 file systems do not lend themselves well to change. So if you expect or need more room in a given file system in the future, you should plan for it up front. Generally, the file systems you need to watch for are ones that will contain dynamic data such as log files.

Note: The current Linux ext2 file system does not provide the capability to alter the file system size once created; however, there are third-party tools such as PowerQuest Partition Magic that can be used.

Table 2-4 File system layout

Name	Partition Type	FS Type	Label	Size (MB)
hda1	Primary	NTFS		5000
hda2[a]	Primary	Linux ext2	/boot	50
hda5	Logical	Linux ext2	/usr	3000
hda6	Logical	Linux ext2	/opt	2000
hda7	Logical	Linux ext2	/home	1000
hda8	Logical	Linux swap		1000
hda9	Logical	Linux ext2	/var	500
hda10	Logical	Linux ext2	/	400
hda11	Logical	Linux ext2	/tmp	400
hda12	Logical	Linux ext2	/usr/local	200
hda13	Logical	Linux ext2	/media	2000
	Logical	Free Space		13000

a. The /boot partition will be flagged as the bootable partition.

Users and groups

In our setup we added the users and groups listed in Table 2-5 during the process of installing the software components.

Table 2-5 Users and groups

User	Group	Home	Shell
root	root	/root	/bin/bash
jason[a]	jason	/home/jason	/bin/bash
db2fenc1	db2fadm1	/home/db2fenc1	/bin/bash
db2inst1	db2iadm1	/home/db2inst1	/bin/bash
db2as	db2asgrp	/home/db2as	/bin/bash

a. Red Hat Linux 7.1 will refuse remote logons by the root user. So you must have a user other than root to log in with. The good news is that you can do this during the installation of Red Hat.

The last three users, db2fenc1, db2inst1, and db2as, are created as part of the DB2 installation. If you should choose to install with one of the other supported databases you will need to follow its guidelines for database user accounts.

2.2 Red Hat Linux installation

There are many sources of information on how to install Linux. This section will outline the steps taken to build our single-tier environment. For more detailed instructions on possible installation planning, please refer to these other sources:

- IBM ITSO Redbooks site:

  http://www.redbooks.com/

Tip: Search on Linux.

- Red Hat Corporation:

  http://www.redhat.com/

- Red Hat Linux 7.1 gotchas and workarounds:

  http://www.redhat.com/support/docs/gotchas/7.1/gotchas-71.html

- *The Official Red Hat Linux x86 Installation Guide* (V7.1), found at:

  http://www.redhat.com/support/manuals/RHL-7.1-Manual/install-guide/

2.2.1 Linux kernel

The WebSphere Application Server V4.0.1, Advanced Edition for Linux requires a minimum release of 7.1 (running the 2.4 kernel) when using Red Hat Linux.

2.2.2 File systems

During the interactive phase of the Red Hat Linux 7.1 operating system installation, you will need to customize the file systems and allocate the necessary space for the software components installed. See Table 2-4 on page 10 for the actual file system sizes used during installation.

Table 2-6 provides the basic set of guidelines for space required by each component. The values in this table were taken from our installed system after the installation using the file system layout values in Table 2-4 on page 10 and using the **df** and **du -ks** commands on each file system/product directory.

Note: There are many configuration scenarios to consider if you plan on using your system for Linux and Windows operating systems. If you plan to boot multiple operating systems, we recommend that you search for the latest information available on the Web, such as:

`http://www.redhat.com/`

The main issues to consider are related to the location of the Linux /boot file system being within the first 1024 cylinders, the use of LILO (Windows 2000 requires its own boot manager), and the master boot record (MBR).

Table 2-6 Component disk space minimum requirements

Component	File system	Disk space required (MB)[a]	Total disk required (MB)[a]
Red Hat Linux 7.1	/ /boot /tmp /var /home /usr	60 5 5 25 5 1200	1300
IBM HTTP Server	/opt	20	20
IBM WebSphere Application Server 4.0, Advanced Edition	/opt	125	125

Component	File system	Disk space required (MB)[a]	Total disk required (MB)[a]
IBM DB2 Server 7.2.1, Enterprise Edition	/usr /home	290 50	340

a. These values are the minimum required to install the packages we chose to install. Your values may vary depending on selected packages and expected growth rate. If you have the disk space, you should consider being generous with your allocations.

2.2.3 Linux software package requirements

During the installation of Red Hat Linux V7.1 in the single-tier installation we chose not to install the Apache version that comes with the Linux media. If you should decide you do want the official Apache installed instead of the IBM HTTP Server (Apache + some IBM modules), do the following during the WebSphere Application Server installation:

- Deselect IBM HTTP Server.
- Select the Apache module plug-in instead of the IBM HTTP Server plug-in.

Note: We recommend that you simply deselect the Apache Web server during the Linux installation and use the IBM HTTP Server instead.

2.3 Install the DB2 Server

This section provides detailed instructions for installing, configuring and verifying IBM DB2 Universal Database V7.2.1, Enterprise Edition for Linux.

The section is organized into the following tasks:

- Pre-requisite Linux packages for DB2
- Pre-installation tasks
- Install the DB2 Server
- Verify the DB2 Server installation
- Configure the DB2 Server
- Create the WAS repository database

2.3.1 Pre-requisite Linux packages for DB2

The DB2 product documentation should be consulted for the official list of prerequisite software.

Note: For more detailed information on DB2 V7.1 and Linux, refer to:

http://www.linux.org/docs/ldp/howto/DB2-HOWTO/kernel24.html

After installing Red Hat Linux V7.1 we found that the only missing package needed was ncurses4-5.0-2. This package can be installed when initially installing Linux or later using one of the RPM package tools.

Note: DB2 for Linux requires the pdksh 5.2.14-12 package. When installing Red Hat 7.1 Linux this will most likely be installed by default, depending on the options selected during the installation.

In the event that you select the custom installation, it is possible that the options that you select will not include the pdksh package. In this scenario, install the pdksh using RPM.

To see if you have this package on your system, you can perform the following steps:

1. Verify whether the package is installed on the system using:

   ```
   rpm --verify ncurses4-5.0-2
   ```

   If no output is generated then the package is installed correctly; otherwise, the rpm package manager will return an error similar to the following:

   ```
   package ncurses4-5.0-2 is not installed
   ```

2. If the package is not installed, then you must locate and install it before proceeding with the DB2 installation. There are a number of ways to do this. The easiest way to install the package is if you have the original Red Hat Linux V7.1 CD handy; otherwise, you can locate and download it from the Web:

   a. Red Hat Linux 7.1 CD

      i. Insert the second CD into the CD-ROM drive (the ncurses4 package exists on that CD).

      ii. If the CD doesn't automatically mount (as it may if you're running X Windows with either the KDE or Gnome desktops) it can manually be mounted by issuing the following command as root:

      ```
      mount /mnt/CD-ROM
      ```

This should work for a stock installation. If it fails, verify the device that represents your CD-ROM drive and issue the following command instead:

```
mount -r /dev/<device> /mnt/cdrom
```

For instance, on our system, the CD-ROM drive is the first device on the second IDE channel. Thus it is (in Linux) referred to as *hdc* and can be mounted by the following command:

```
mount -r /dev/hdc /mnt/cdrom
```

b. Find the RPM on the Web:

   i. Go to http://rpmfind.net/

   ii. Enter 'ncurses4' into the search box and click **Search**, and download the RPM.

   iii. Find and download nucurses4 rpm for Red Hat Linux 7.1.

c. Alternatively, you can find it in the directories of the Red Hat Linux FTP site:

   ftp://ftp.redhat.com

3. Once you have the RPM file either mounted or saved somewhere on your system, use the following command to install or upgrade the package:

   ```
   rpm -U --nodeps <path to file>/ncurses4-5.0-2.i386.rpm
   ```

   For example, from the product CD:

   ```
   rpm -U --nodeps /mnt/cdrom/RedHat/RPMS/ncurses4-5.0-2.i386.rpm
   ```

2.3.2 Pre-installation tasks

Prior to installing IBM DB2 Universal Database V7.2.1, Enterprise Edition, the following checks need to be completed:

- Verify that there are no existing active services that use the same DB2 TCP/IP ports on the server:
  - 523 (DB2 Server)
  - 50000 (default DB2 instance connection port)
  - 50001 (default DB2 instance interrupt port)
  - 50002 (DB2 Control Server)

  We suggest using the following command for this task:

  ```
  netstat -an | grep LISTEN
  ```

- If you will be running DB2 with multiple instances you will need to increase the IPC limits on the database server:

– Increase it immediately on the command line by running *one* of the following two commands:

```
sysctl -w kernel.msgmni=128
```

or

```
echo “128” > /proc/sys/kernel/msgmni
```

– Also, increase it for each server reboot by adding the lines in Example 2-1 to the end of the /etc/sysctl.conf file (or adding one of the above two commands to your favorite startup file):

Example 2-1 IPC startup parameters for /etc/sysctl.conf file

```
# Sets maximum number of message queues to 128
# Set this to 1024 or higher on production systems
kernel.msgmni = 128
```

2.3.3 Install the DB2 Server

In order to install IBM DB2 Universal Database V7.2.1, Enterprise Edition for Linux, perform the following steps:

1. Log in as root and start a terminal session.
2. Mount the DB2 V7.2.1 CD-ROM:

```
mount /mnt/cdrom
```

Note: If you are running X Windows with either the KDE or Gnome window managers the CD-ROM may automatically be mounted for you. To verify this, simply do a `mount | grep /mnt/cdrom` and if you get any output then the CD-ROM has already been mounted.

3. Start the DB2 installer program:

```
cd /mnt/cdrom
./db2setup
```

4. In the Install DB2 V7.2.1 window, select only the following option:
   – DB2 UDB Enterprise Edition

Tip: Navigation tips:

- Press Tab to move between available options and fields.
- Press Ctrl+L to refresh the window at any point.
- Highlight and press Enter to select an option.

5. Highlight the DB2 Product Library's **Customize** option and press Enter.
6. In the DB2 Product Library window, highlight the appropriate option for your locale under the DB2 Product Library (HTML) section, then highlight **OK** and press Enter.
7. Highlight **OK** and press Enter.
8. In the Create DB2 Services window, select the **Create a DB2 Instance** option, highlight **OK** and press Enter.
9. When the DB2 instance authentication window appears, enter the following:
   - User Name: `<db2_instance_owner>`

     For example, `db2inst1`
   - User ID: `<use default UID>`
   - Group Name: `db2iadm1`
   - Group ID: `<use default GID>`
   - Home Directory: `/home/<db2_instance_owner>`

     For example, `/home/db2inst1`
   - Password: `<user_password>`
   - Verify Password: `<user_password>`

Important: The DB2 installer uses the above information to automatically perform the following operations:

- Create a group db2iadm1
- Create a user <db2_instance_owner> with primary group db2iadm1
- Sets <db2_instance_owner> password to <user_password> value. The password used must meet DB2 requirements: eight characters or less and not containing the characters “<“ or “>”.
- Changes the ownership (owner:group) of the /home/<db2_instance_owner> directory to be <db2_instance_owner>:db2iadm1

10. Highlight **OK** and press Enter.
11. In the Fence User window, enter the following:
    - User Name: `db2fenc1`
    - User ID: `<use default UID>`
    - Group Name: `db2fadm1`
    - Group ID: `<use_default_GID>`
    - Home Directory: `/home/db2fenc1`
    - Password: `<db2fenc1_password>`
    - Verify password: `<db2fenc1_password>`

Important: The DB2 installer uses the above information to automatically perform the following operations:

- Create a group `db2fadm1`
- Create a user `db2fenc1` with primary group `db2fadm1`
- Sets `db2fenc1` password to `<db2fenc1_password>` value. The password used must meet DB2 requirements: eight characters or less and not containing the characters “<“ or “>”.
- Changes the ownership (owner:group) of the `/home/db2fenc1` directory to be `db2fenc1:db2fadm1`

12. Highlight **OK**, and press Enter.
13. In the DB2 Warehouse Control Database window, deselect the Setup DB2 Warehouse Control Database option, then highlight **OK**, and press Enter.

14. In the Create DB2 Services window, highlight the **Create Administration Server** option, then enter the following:
    - User Name: `db2as`
    - User ID: `<use default UID>`
    - Group Name: `db2asgrp`
    - Group ID: `<use default GID>`
    - Home Directory: `/home/db2asgrp`
    - Password: `<db2asgrp_password>`
    - Verify password: `<db2asgrp_password>`

Important: The DB2 installer uses the above information to automatically perform the following operations:

- Create a group db2asgrp
- Create a user db2as with primary group db2asgrp
- Sets db2as password to <db2as_password> value. The password used must meet DB2 requirements: eight characters or less and not containing the characters “<“ or “>”.
- Changes the ownership (owner:group) of the /home/db2as directory to be db2as:db2asgrp

15. Highlight **OK** and press Enter.
16. A message window appears indicating that DB2SYSTEM will be set to <hostname>. Highlight **OK** and press Enter.
17. Back in the Create DB2 Services window, highlight **OK** and press Enter.
18. The Summary Report window is displayed, listing the product components to be installed. Highlight **Continue** and press Enter.
19. A warning window appears indicating This is your last chance to stop. Highlight **OK** and press Enter.
20. The db2setup program installs the selected components. Depending on the speed of your processor, this can take up to 15 minutes.
21. You may be prompted to register the product. Complete the registration then exit back to the install window.
22. When the install completes, a notice window informs you whether the installation was successful. Highlight **OK** and press Enter.
23. Scan the Status Report to ensure that all components were installed successfully. Highlight **OK** and press Enter.

24. In the DB2 Installer window, highlight **Close** and press Enter.
25. A window appears asking `Do you want to exit the DB2 Installer?` Highlight **OK** and press Enter.
26. Unmount the CD-ROM:

```
cd /
umount /mnt/cdrom
```

The DB2 installation is now complete.

2.3.4 Verify the DB2 Server installation

To verify the DB2 Server installation, complete the following tasks:

- Verify home directory permissions
- Verify the DB2 instance owner profile
- Verify the DB2 instance symbolic links
- Verify the DB2 release level
- Verify the DB2 service name
- Verify the database manager configuration
- Create the DB2 sample database

Verify home directory permissions

Check that the home directory ownership has been correctly set up by the db2setup program:

Table 2-7 DB2 home directory required permissions

Home directory path	Owner	Group	Permissions[a]
/home/<db2_instance_owner>	<db2_instance_owner>	db2iadm1	drwxr-xr-x
/home/db2fenc1	db2fenc1	db2fadm1	drwxr-xr-x
/home/db2as	db2as	db2asgrp	drwxr-xr-x

a. Permissions needed so that the DB2 instance owner can read, write, and execute files and directories within the path. Group members' and other users' access rights are up to each company's security policies and business needs.

If a DB2 related home directory has not been correctly configured, perform the following steps:

1. Log in as root, and start a terminal session.

2. Issue the command, substituting values from the table above:

```
chown -fR <owner>:<group> <home_directory_path>
```

Verify the DB2 instance owner profile

The DB2 Server installation should set up the .bashrc environment file of the <db2_instance_owner> so that the DB2 environment is set up when the user logs in.

1. The following content should have been added to `the file`:

```
if [ -f /home/db2inst1/sqllib/db2profile ] ; then
. /home/db2inst1/sqllib/db2profile
fi
```

2. If not present, manually edit the file to add the above content.

Verify the DB2 instance symbolic links

The DB2 Server installation automatically creates a DB2 instance <db2_instance_owner> under the /home/<db2_instance_owner> directory. As part of the instance creation, db2setup should create symbolic links in the /home/<db2_instance_owner>/sqllib directory to files under /usr/IBMdb2/V7.1.

Perform the following steps to check whether the symbolic links have been created:

1. Log in as root, and start a terminal session.
1. Change directory to /home/<db2_instance_owner>/sqllib.
2. Check whether a number of symbolic links exist pointing to files under /usr/IBMdb2/V7.1.
3. If not, issue the following commands:

```
cd /usr/IBMdb2/V7.1/cfg
./db2ln
```

Verify the DB2 release level

Check that DB2 has the correct internal release level to meet WAS requirements:

1. Change to user <db2_instance_owner>:

```
su - <db2_instance_owner>
```

2. Enter the following command:

```
db2level
```

This should generate output similar to the following:

```
DB21085I  Instance "db2inst1" uses DB2 code release "SQL07021" with level
identifier "03020105" and informational tokens "DB2 v7.1.0.43", "s010504"
and "U475381a"
```

Note: An internal release level of 7.1.0.43 should be indicated.

Verify the DB2 service name

1. Open the /etc/services file and locate the entries that have comments referring to the DB2 instance connection port.
2. Locate the service name in the first column that corresponds to the lower port number. For example, if the following services were displayed:

   ```
   db2cdb2inst1 50000/tcp #Connection port for DB2 instance db2inst1
   db2idb2inst1 50001/tcp #Interrupt port for DB2 instance db2inst1
   ```

   Record the `db2cdb2inst1` service name for later use.

Verify the database manager configuration

Check the service name is recorded in the database manager configuration:

1. Change to user <db2_instance_owner>:

   ```
   su - <db2_instance_owner>
   ```

2. Enter the following command:

   ```
   db2 get dbm cfg | grep SVCENAME
   ```

3. Verify that the SVCENAME value matches the service name recorded above from the `services` file. For example, something similar to the following should be displayed:

   ```
   TCP/IP Service name (SVCENAME)=db2cdb2inst1
   ```

4. If the value does not match, update the database manager configuration using the following commands:

   ```
   db2 update dbm cfg using svcename db2cd
   db2stop
   db2start
   ```

   Where the value of `SVCENAME` must be replaced with the service name.

Create the DB2 sample database

The DB2 installation can be tested by creating and connecting to the sample database supplied with DB2 specifically for this purpose:

1. Change to user <db2_instance_owner>:

   ```
   su - <db2_instance_owner>
   ```

   For example:

```
su - db2inst1
```

2. Create the DB2-supplied sample database, named sample, to verify DB2 is working properly by typing the following command:

```
db2sampl
```

3. List all DB2 databases for the DB2 instance:

```
db2 list db directory
```

This should give output containing the following:

```
Database 1 entry:
 Database alias                 = SAMPLE
 Database name                  = SAMPLE
 Database drive                 = /home/db2inst1
 Database release level         = 9.00
 Comment                        =
 Directory entry type           = Indirect
 Catalog node number            = 0
```

4. Test the connectivity to the database:

```
db2 connect to sample
db2 disconnect current
```

2.3.5 Configure the DB2 Server

After the DB2 Server installation, many configuration tasks must be performed in preparation for the WebSphere Application Server installation.

This section is organized into the following tasks:

- Update root administrative groups
- Update JDBC level
- Configure the TCP/IP communication mode
- Verify the DB2 environment
- Update the root environment file

Update root administrative groups

The DB2 Server installation should add the db2asgrp administrative group to the root user.

Perform the following steps to check whether the root account's administrative groups have been amended:

1. Log in as root, and start a terminal session.
2. Issue the following command:

```
groups
```

3. If db2asgrp is not listed as one of the groups assigned to root, you can use the Red Hat GUI tools to reconfigure the root user or you can do the following:

   ```
   vigr
   ```

   This will bring you into a vi editing session with a locked version of the /etc/group file. Find the line in the file that starts with ‘db2asgrp:’ and add the user, root, to the end of it (separated by a comma from other user names). The line should look similar to this after editing:

   ```
   db2asgrp:db2as,root
   ```

   Once you have added that, press Esc+:wq to save and quit vi. The system will then prompt you to edit the shadow group file. Choose Yes and do the same thing to that file, saving and quitting at the end of your editing.

Update JDBC level

The WebSphere Application Server V4.0.1 requires the use of JDBC 2.0, whereas the default installation of IBM DB2 V7.2.1 uses JDBC 1.2. To update the DB2 JDBC level, complete the following steps:

1. Change to user <db2_instance_owner>:

   ```
   su - <db2_instance_owner>
   ```

2. Add the following content to the end of the <db2_instance_owner> .bashrc environment file:

   ```
   if [ -f ~/sqllib/java12/usejdbc2 ] ; then
   . ~/sqllib/java12/usejdbc2
   fi
   ```

Note:

- We found that the usejdbc2 file on our system was missing a semicolon, ‘;’, on lines 32 and 36 after the right brace, ‘]’, and before the word ‘then’. The corrected lines look like the following:

  ```
  Line 32: if [[ `uname` = "AIX" ]]; then
  Line 36: elif [[ `uname` = "HP-UX" ]]; then
  ```

- The ~ in the above example is used to supply the user home directory. For example, when logged in as db2inst1, the ~ is equivalent to /home/db2inst1.

Configure the TCP/IP communication mode

The DB2 Server may need to be re-configured to use TCP/IP as its primary communication method by completing the following steps:

1. Change to user <db2_instance_owner>:

```
su - <db2_instance_owner>
```

2. Check whether TCP/IP is the current DB2 communication method. The following command should return a value of tcpip:

```
db2set DB2COMM
```

3. If not, reset the DB2COMM DB2 environment variable:

```
db2set DB2COMM=tcpip
```

Verify the DB2 environment

After the above configuration steps, we need to check that the environment being set up by the db2profile and usejdbc2 scripts are correct:

1. Change to user <db2_instance_owner> (for example, db2inst1).

```
su - db2inst1
```

2. Issue the following command:

```
set | grep [Dd][Bb]2
```

3. Verify that the environment variables in this output match the values in Table 2-8, where the DB2 instance owner is db2inst1.

Table 2-8 DB2 Server required environment variables

Environment variable	Required value
CLASSPATH	/home/db2inst1/sqllib/function:/home/db2inst1/sqllib/java12/db2java.zip:/home/db2inst1/sqllib/java/runtime.zip
DB2DIR	/usr/IBMdb2/V7.1
DB2INSTANCE	db2inst1
INSTHOME	/home/db2inst1
LD_LIBRARY_PATH	:/home/db2inst1/sqllib/java12:/home/db2inst1/sqllib/lib
LIBPATH	:/home/db2inst1/sqllib/lib
PATH	.....:/home/db2inst1/sqllib/java12:/home/db2inst1/sqllib/bin:/home/db2inst1/sqllib/adm:/home/db2inst1/sqllib/misc

Update the root environment file

The WebSphere Application Server will be run under root and will require access to the DB2 environment so that it can access the WAS administration database.

1. This requires that the root account's environment .bashrc file, found in the /root directory, be edited to add the content as seen in Example 2-2 at the end of the .bashrc file.

Example 2-2 Sample entries in the root .bashrc file for DB2

```
# Setup DB2 environment for root user.
if [ -f /home/db2inst1/sqllib/db2profile ] ; then
. /home/db2inst1/sqllib/db2profile
fi

# Force DB2 to use JDBC 2.0.
if [ -f /home/db2inst1/sqllib/java12/usejdbc2 ] ; then
. /home/db2inst1/sqllib/java12/usejdbc2
fi
```

Where `db2inst1` is the DB2 instance owner in Example 2-2.

2. Log out and log in for the changes to take effect.

2.3.6 Create the WAS repository database

The following steps create the WebSphere Application Server repository database, also known as the WAS database. The database will be populated with WAS schema and default values in a later task.

To set up the WAS database, complete the following steps:

1. Log in as <db2_instance_owner>.
2. Create the WAS repository database.

   ```
   db2 create db was1
   ```

   Note: Although the repository database created here is called `was1`, any valid DB2 database name can be used.

3. Configure the heap size of the WAS repository database as required by the WebSphere Application Server:

   ```
   db2 update db cfg for was1 using applheapsz 256
   ```

4. Verify that the new database is known to DB2:

   ```
   db2 list db directory
   ```

   You should see a message containing the following output:

   ```
   Database 1 entry:
    Database alias                 = WAS1
    Database name                  = WAS1
    Database drive                 = /home/db2inst1
    Database release level         = 9.00
    Comment                        =
    Directory entry type           = Indirect
   ```

```
 Catalog node number             = 0
```

5. Catalog the TCP/IP node.

   In order to access the administration database via TCP/IP, the DB2 node must first be cataloged.

   For example:

   ```
   db2 catalog tcpip node ganci5 remote ganci5 server db2cdb2inst1
   ```

   Syntax:

   ```
   db2 catalog tcpip node <node_name> remote <local_hostname> server
   <service_name>
   ```

   The <service_name> used to catalog the node must be the same as the database instance connection port name in the /etc/services file. The <node_name> chosen can be any valid DB2 nodename.

6. Verify the attach to the TCP/IP node:

   For example:

   ```
   db2 attach to ganci5 user db2inst1 using db2inst1
   ```

   Syntax:

   ```
   db2 attach to <node_name> user <db2_instance_owner> using
   <db2_instance_owner_passwd>
   ```

7. Catalog the database.

   The administration database must now be cataloged as part of this TCP/IP node:

   For example:

   ```
   db2 catalog db was1 as was at node ganci5
   ```

   Syntax:

   ```
   db2 catalog db <database_name> as <database_alias> at node <node_name>
   ```

8. Verify that the database alias is known to DB2:

   ```
   db2 list db directory
   ```

   The output should contain something like the following message:

   ```
   Database 2 entry:
    Database alias                 = WAS
    Database name                  = WAS1
    Node name                      = <node_name>
   ```

```
 Database release level          = 9.00
 Comment                         =
 Directory entry type            = Remote
 Catalog node number             = -1
```

9. Verify a connection to the local database via TCP/IP:

```
db2 connect to was user db2inst1 using db2inst1
db2 disconnect current
```

   where `db2inst1` is the DB2 instance owner password on the local DB2 server.

Tip: When the DB2 administration account is used to create a new database, it is automatically granted DBA access rights. You only need to specifically grant access to another user if you plan to access the database from an account other than <db2_instance_owner>.

10. Reboot the system for the changes to take effect. *Alternatively*, you can restart the DB2 processes by doing the following:

```
su - db2inst1 "-c db2stop"
su - db2as "-c db2stop"
su - db2as "-c db2start"
su - db2inst1 "-c db2start"
```

2.4 Install the WebSphere Application Server

This section provides detailed instructions for installing, configuring and verifying WebSphere Application Server V4.0.1, Advanced Edition for Linux.

The section is organized into the following tasks:

- Pre-installation tasks
- Install the WebSphere Application Server

2.4.1 Pre-installation tasks

Prior to installing the IBM WebSphere Application Server V4.0.1, Advanced Edition for Linux the following checks and tasks need to be completed:

1. Check IP ports are unused.
2. Stop Web server processes.

Check IP ports are unused

1. Check that there are no existing active services that use the following IP ports on the server:
   - 900 (bootstrap port)
   - 9000 (Naming Service)
   - 9080 (default application server)

   We suggest using the following command for this task:

   ```
   netstat -an | grep LISTEN
   ```

Stop Web server processes

If during the Red Hat Linux V7.1 installation you chose to install the Apache Web server software instead of the IBM HTTP Server you will need to ensure that you stop the Apache Web server before proceeding. This is because the WebSphere Application Server installation updates the httpd.conf configuration file as part of the Web server plug-in component installation.

1. Log in as root, and start a terminal session.
2. Issue the following commands:

   ```
   cd <apache_server_install_path>/bin
   ./apachectl stop
   ```

IBM HTTP Server

The WebSphere Application Server installation utility is capable of installing the IBM HTTP Server during the WAS installation. As an alternative, many system administrators install the IBM HTTP Server prior to the installation of the WebSphere Application Server.

This is done for the following reasons:

- SSL configuration

  If you plan to enable the IBM HTTP Server for SSL, the entries for SSL configuration are included (commented out) in the httpd.conf.sample file and not in the httpd.conf file. The WAS installation utility by default updates the WAS configuration entries to the httpd.conf. The end result is that the user will need to merge entries for WAS and SSL. By installing and configuring the IBM HTTP Server, including SSL configuration prior to the WAS installation, the WAS installation will update the correct version of the httpd.conf with the WAS plug-in entries.

- Verify before adding prerequisite components.

Many administrators strongly believe that each component should be installed, configured, and verified before installing components that rely on the successful installation of the previous component. In this scenario, the user would install the HTTP Server, configure the Web server (SSL, ServerName, etc.), verify the server, and then install WAS.

2.4.2 Install the WebSphere Application Server

There are two ways to install the WebSphere Application Server:

- Interactive install - GUI
- Automated install - silent

You can choose whichever one you need as appropriate; however, we do recommend that you using the GUI install for new users to WAS V4.

Interactive install - GUI

To install the IBM WebSphere Application Server V4.0.1, Advanced Edition for Linux using the GUI installer interface, complete the following steps on the WAS server machine:

Tip: The WAS installer (install.sh) also provides a non-GUI scripted or “silent” mode of operation (refer to“Automated install - silent” on page 34).

1. Log in as root and start a terminal session.
2. Load the IBM WebSphere Application Server V4.0 CD-ROM into the CD-ROM drive and mount the CD.

   ```
   mount -t iso9660 -r /dev/cdrom /cdrom
   ```

3. Change directory to the installation root.
4. Ensure the DISPLAY and TERM environment variables are properly set.
5. Run the install.sh installation script:

   ```
   ./install.sh
   ```

6. In the Welcome window, click **Next**.

Important: If prerequisite checking is enabled for UNIX, then an alert may be displayed indicating that some of the installation prerequisites have not been met. This can even occur if newer patches/packages than those listed in prereq.properties are installed. If such an alert is displayed, recheck all prerequisites, and if they are met or exceeded, perform the following steps:

- Exit the installation.
- Disable prerequisite checking.
- Restart the installation from the beginning.

7. In the Installation Options window, select **Custom Installation** and click **Next**.
8. In the Choose Application Server Components window, select *all* of the options (see Figure 2-2 on page 32), then click **Next**.

   In the following chapters when installing WebSphere we will uncheck the IBM HTTP Server component in this window for the stand-alone application server installation.

Important: Although not listed in the Application Server Components window, the IBM JDK 1.3.0 is automatically installed under the WAS installation directory. There is no need to separately install a JDK for use by:

- WebSphere Application Server
- Web Server plug-ins

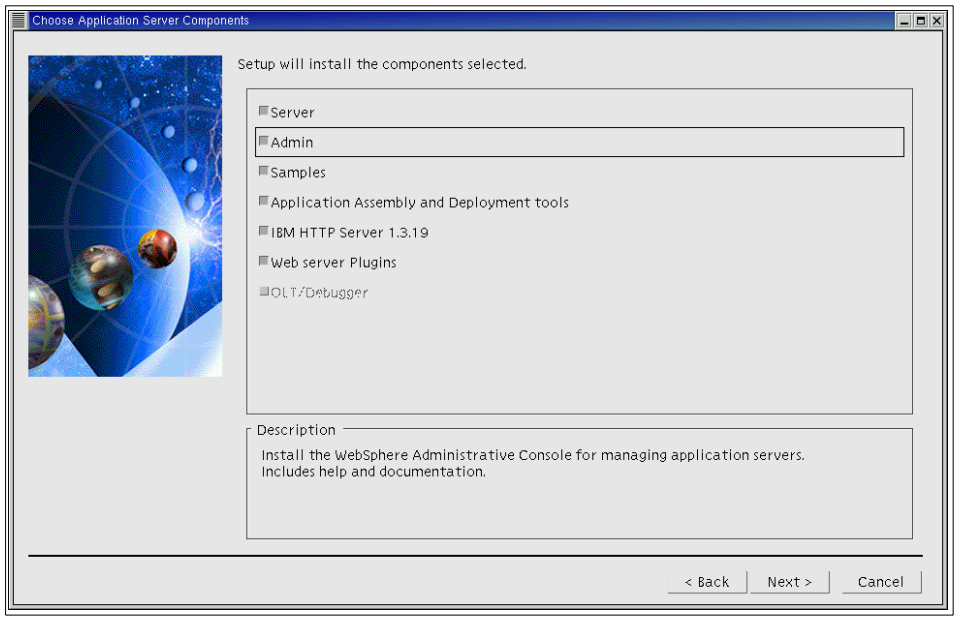


Figure 2-2 WAS V4 for Linux - installation options

9. In the Choose Web server plug-in window, select only the **IBM HTTP Server Plugin**, then click **Next**.
10. In the Database Options window, enter the following (see Figure 2-3) then click **Next**:

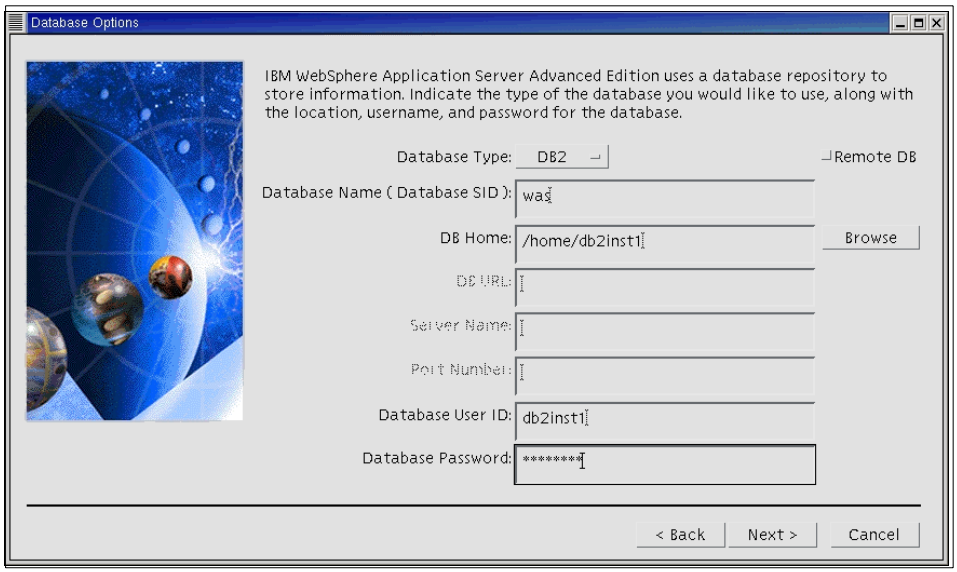


Figure 2-3 Specify DB2 database connection settings

- Database Type: `DB2`
- Database Name (Database SID): `<was_database_alias>`

  Enter the DB2 TCP/IP alias of the WAS administration database created during the configuration of DB2 Server.
- DB Home: `/home/<db2_instance_owner>`

  For example, /home/db2inst1
- Database User ID: `<db2_instance_owner>`

  For example, db2inst1
- Database Password: `<db2_instance_owner_password>`
- Do *not* check Remote DB.

Note: Whether the DB2 WAS database is local or remote, in our test environment we configure the DB2 client to access the database through a catalog alias. Under these conditions, the Remote DB setting should not be selected.

11. In the Select Destination Directory window, accept the default destination directories as follows, and then click **Next** to continue:
    - WebSphere Application Server: `/opt/WebSphere/AppServer`

– IBM HTTP Server: `/opt/IBMHTTPServer`

12. In the Install Options Selected window, review the selected components. If they are correct, click **Install** to start the installation.
13. In the Location of Configuration Files window, enter the path to the IBM HTTP Server configuration file (httpd.conf), then click **Next**.
14. In the Setup Complete window, click **Finish**.

The installation of the WebSphere Application Server is now complete.

Note: Continue to 2.5, “Configure and verify the IBM HTTP Server” on page 37.

Automated install - silent

This section describes how to install WebSphere Application Server V4.0.1, Advanced Edition for Linux using the non-interactive or silent mode. To complete a silent installation, you will use the default response file or create a customized one, and then execute the installation script for WebSphere Application Server, supplying the response file as a command-line parameter.

These instructions assume the following:

- Your machine has sufficient memory and disk space for your installation. See the WebSphere Application Server supported hardware, software, and APIs at the following URL for the proper requirements:

  `http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html`
- You have installed and configured a supported database.
- You do not have a previous version of WebSphere Application Server already installed on this machine.
- If you are using IBM HTTP Server as your Web server, you will install it at the same time and onto the same node as you install WebSphere Application Server. If you are using another supported Web server with WebSphere Application Server, you have already installed it onto the same node as the WebSphere Application Server.

Note: IBM HTTP Server is supplied with WebSphere Application Server. If you plan to use a different Web server, you must purchase it and install it separately. It is recommended that the Web server be installed before WebSphere Application Server.

Using the default response file

A default response file, named install.script, is supplied with WebSphere Application Server. You can use this default response file to install WebSphere Application Server using the default options, or as a template for creating a customized response file.

If you use the default response file to install WebSphere Application Server using the default options, the following software and other resources are installed:

- IBM Java 2 Software Developer's Kit (SDK) 1.3.0
- IBM HTTP Server 1.3.19
- WebSphere Application Server 4.0
- WebSphere Application Server application samples
- Documentation in U.S. English

Note: All products except IBM HTTP Server are installed into the directory /opt/WebSphere/AppServer; IBM HTTP Server is installed into the directory /opt/IBMHTTPServer. In addition, WebSphere Application Server is configured for use with IBM HTTP Server and InstantDB when you use the default response file.

Using a customized response file

The default response file can also be used as a template for creating a customized response file. The default response file can be edited to enable the configuration of WebSphere Application Server with a different supported Web server or database, or to install the products in a different directory. Detailed comments within the default response file guide you through the installation and configuration options available for performing a silent installation.

Note: When performing a silent installation, WebSphere Application Server is installed in a directory that depends on the operating system type. For Linux this is /opt/WebSphere/AppServer; however, if desired this can be changed in the install script.

Performing a silent installation

Perform the following steps to create a customized response file (if desired) to install WebSphere Application Server. These instructions assume that the installation is being performed from the product CD-ROM:

Important: If a Web server is running on your system, stop the Web server. If you plan to install IBM HTTP Server 1.3.19 as part of the WebSphere Application Server installation and you have a level of IBM HTTP Server prior to 1.3.19 on your system, you must uninstall it for the WebSphere Application Server installation program to install IBM HTTP Server 1.3.19.

1. Log in as root, and start a terminal session.
2. Mount the CD-ROM or change to the directory where WebSphere has been downloaded and expanded:
   - Insert the WebSphere CD into the CD-ROM drive.
   - If the CD doesn't automatically mount (as it may if you're running X Windows with either the KDE or Gnome desktops) it can manually be mounted by issuing the following command as root:

     ```
     mount /mnt/cdrom
     ```

     This should work for a stock installation. If it fails, verify the device that represents your CD-ROM drive and issue the following command instead:

     ```
     mount -r /dev/<device> /mnt/cdrom
     ```

     For instance, on our system, the CD-ROM drive is the first device on the second IDE channel. Thus it is (in Linux) referred to as hdc and can be mounted by the following command:

     ```
     mount -r /dev/hdc /mnt/cdrom
     ```
3. The commands in these steps assume the CD-ROM is mounted at /mnt/cdrom. If you mount the CD-ROM at a different location, or you are installing from a file system, use that location when issuing commands.
4. Ensure that the DISPLAY and TERM environment variables are set correctly.
5. Navigate to the /mnt/cdrom directory.
6. Ensure that you are in the /mnt/cdrom directory and create a copy of the default response file by using the **cp** command, as follows:

   ```
   cp install.script /tmp/new_install.script
   ```

   In this command, `new_install.script` represents the full path name of the copy of the default response file you are creating (for example, `/tmp/my_install.script`). The name of your response file must have a .script extension.
7. If you plan to install WebSphere Application Server by using the default options included in the default response file, proceed to Step 9.
8. If you plan to use the default response file as a template for creating a customized response file, perform the following steps:

a. Use a text editor to open your copy of the default response file.

b. Use the detailed comments throughout the file to help you select the appropriate options for your WebSphere Application Server installation.

c. Save the changes that you have made to the customized response file.

9. The install.sh script uses the response file to install the components and options that you have selected. Run the installation script as follows:

```
./install.sh -silent -responseFile /tmp/new_install.script
```

10. If you choose to install the plug-in for IBM HTTP Server, the installation process checks if you have the correct version of the Web server on your machine. If you do not have IBM HTTP Server installed on your machine, the installation process performs one of the following actions based on whether you have indicated in your response file to install IBM HTTP Server:

   – If you indicated in your response file that you do want to have IBM HTTP Server installed, the installation process installs the plug-in for it.

   – If you indicated in your response file that you do not want to have IBM HTTP Server installed, the script exits without installing the plug-in.

11. After installation is complete, refer to the log file named install.log located in the /tmp directory to determine if the silent installation was successful. A copy of this file also exists in the WebSphere Application Server logs directory.

12. Unmount the CD-ROM before removing it from the CD-ROM drive by using the umount command, as follows:

```
umount /mnt/cdrom
```

13. If you installed IBM HTTP Server as part of the WebSphere Application Server silent installation you will need to proceed to the next section, 2.5, “Configure and verify the IBM HTTP Server” on page 37; otherwise, you can proceed to 2.6.1, “Verify the WebSphere Application Server installation” on page 46.

2.5 Configure and verify the IBM HTTP Server

After the installation of IBM HTTP Server, there are a few tasks that need to be performed to complete the Web server setup. Some of these tasks are optional and some should be done in a normal setting.

One choice that will need to be made is whether you will run the IBM Administration Web server on the Web server system. This is a second instance of the IBM HTTP Server that runs by default on port 8008 and allows you to configure most aspects of the regular IBM HTTP Server running on port 80. Some companies do not wish to run this Administration server for security reasons, since it would leave another port open to potential attack.

The following configuration tasks should be completed to configure and verify the IBM HTTP Server:

- Create HTTP Server admin account
- Create an HTTP Administration Server runtime user account
- Create an HTTP Server runtime user account
- Configure the HTTP Server for SSL
- Configuring the ServerName
- Restart the IBM HTTP Server
- Verify the IBM HTTP Server

2.5.1 Create HTTP Server admin account

If you should decide to run the HTTP Administration Server, the instructions in this section should be followed. Otherwise this section can be skipped.

The admin account is used to access the HTTP Administration Server Configuration GUI. To create the account, perform the following steps:

1. Log in as root, and start a terminal session.
2. Change directory to the <http_server_install_path>/bin directory. For example:

   ```
   cd /opt/IBMHTTPServer/bin
   ```

3. Create HTTP admin account <http_admin_account>.

   For example:

   ```
   ./htpasswd -m ../conf/admin.passwd httpadm
   ```

   Where `httpadm` is the admin user account. The **`htpasswd`** command will prompt you for a new password,and then for you to retype the new password.

Syntax: Create HTTP admin account

```
./htpasswd -m ../conf/admin.passwd <http_admin_user>
```

2.5.2 Create an HTTP Administration Server runtime user account

If you should decide to run the HTTP Administration Server the instructions in this section should be followed. Otherwise this section can be skipped.

Although the Admin HTTP Server process is started under the root account, it must be configured to then switch to run under another account. A UNIX account can be created specifically for the purpose.

Perform the following steps to create the UNIX account and configure the Admin HTTP Server:

1. Log in as root, and start a terminal session.
2. Change directory to the <http_server_install_path>/bin directory. For example:

   ```
   cd /opt/IBMHTTPServer/bin
   ```

3. Run the setupadm script:

   ```
   ./setupadm
   ```

   Answer the prompts as follows:

   a. Supply a user ID to run the Administration Server. For example:

      User ID: `httprun`

   b. Supply a Group Name to run the Administration Server. For example:

      Group Name: `httpgrp`

   c. Supply the directory containing the files for which a change in the permissions is necessary. The default is /opt/IBMHTTPServer/conf.

      Press Enter to accept the default.

   d. To perform the change, enter 1. To quit with no changes, enter 2 (default).

      Type `1` and press Enter to perform changes.

   e. The configuration file /opt/IBMHTTPServer/conf/admin.conf will be saved. Do you wish to update the Administration Server Configuration file? To perform the change, enter 1. To exit with no change, enter 2 (default).

      Type `1` and press Enter to update configuration file.

f. Do you wish to run the Admin Server and IHS Server in a language other than English? For a language other than English, enter 1. For English, enter 2 (default).

   Press Enter to accept the default (English).

4. The setupadm program returns to the system prompt.

2.5.3 Create an HTTP Server runtime user account

This section is optional. By default the configuration file (httpd.conf) is set up to run the HTTP Server as user *nobody* and group *nobody*. This should be sufficient for most installations; however, some companies prefer not to do this and, instead, create separate Web server accounts. If the latter is the case for you then you will need to first create the desired user and group on the Linux system using a command such as the following. Otherwise, the user/group setting in the httpd.conf file can be left at defaults:

```
useradd www
```

This will also create a group called *www* automatically.

Edit the <http_server_install_path>/conf/httpd.conf file and update the following settings:

Table 2-9 httpd.conf required settings

Setting	Required value...
User	www
Group	www

2.5.4 Configure the HTTP Server for SSL

Enabling SSL support for the IBM HTTP Server is often required in a production Web server runtime environment. To configure the IBM HTTP Server with SSL support, complete the following steps:

1. Stop the IBM HTTP Server:

```
cd /opt/IBMHTTPServer/bin
./apachectl stop
```

2. Copy the sample HTTPd.conf file:

```
cd /usr/HTTPServer/conf
cp httpd.conf httpd.conf.orig
cp httpd.conf.sample httpd.conf
```

3. Modify the httpd.conf file with a text editor (for example, vi). Uncomment the following lines by removing the # sign:

   ```
   LoadModule ibm_ssl_module libexec/mod_ibm_ssl_<encryption_level>.so
   ```

   Where the <encryption_level> is the appropriate level for your locale (for example, 128 for en_US).

   Remove the comment character from the following lines to configure the SSL module in the HTTP Server:

   ```
   AddModule mod_ibm_ssl.c
   Listen 80
   Listen 443
   <VirtualHost host.domain.com:443>
   ```

   Note: You must substitute your fully qualified host name in this line (for example, <VirtualHost ganci5.itso.ral.ibm.com:443>.

   Uncomment the following:

   ```
   SSLEnable
   </VirtualHost>
   SSLDisable
   Keyfile "/usr/HTTPServer/keys/keyfile.kdb"
   ```

   Note: Replace the word keys with ssl.

   Uncomment the SSL timeout values:

   ```
   SSLV2Timeout 100
   SSLV3Timeout 1000
   ```

4. Save your changes.
5. Set the Java path by typing the following:

   ```
   cd /opt/IBMHTTPServer/ssl
   export JAVA_HOME=/opt/WebSphere/AppServer/java
   ```

6. Start IBM Key Management utility by typing the following command:

   ```
   ikeyman
   ```

7. When the IBM Key Management window appears, click the **Key Database File** menu and select **New.**
8. When the New window appears, enter the following and then click **OK**:
   - File name: keyfile.kdb
   - File location: /usr/HTTPServer/ssl
9. When the Password Prompt window appears, enter your password and then click **OK**. This password is used to access the keyfile.

- Check **set expiration time** and change it to the desired number of days (for example, 360).
- Enable **Stash the password to a file.**

10. When the Information window appears, confirming the password has been saved and encrypted in the keyfile.sth file, click **OK.**

Note: For production SSL enablement, you will need to get a real SSL certificate from an organization such as VeriSign at:

`http://www.verisign.com/`

For the purposes of this Redpaper and testing, we created a self-signed certificate.

11. Click **Create** and select **New Self-Signed Certificate.**
12. When the Create New Self-Signed Certificate window appears, enter the following (required fields) and then click **OK**.
    - Key Label: `ITSO test SSL certificate` (user-defined name)
    - Organization: `IBM` (your company name)
13. Close the Key Management utility.

Changes will not take effect until the HTTP Server is restarted.

2.5.5 Configuring the ServerName

To configure the IBM HTTP Server, ServerName to include a fully qualified hostname, complete the following steps:

1. Stop the IBM HTTP Server:

```
cd /opt/IBMHTTPServer/bin
./apachectl stop
```

2. Change to the directory of the httpd.conf:

```
cd /opt/IBMHTTPServer/conf
```

3. Modify the httpd.conf file to update the value of the ServerName keyword as the fully qualified hostname.

   For example:

```
ServerName ganci5.itso.ral.ibm.com
```

4. Start the HTTP Server:

```
cd /opt/IBMHTTPServer/bin
./apachectl start
```

2.5.6 Restart the IBM HTTP Server

The IBM HTTP Server can be restarted in one of the following methods:

1. For an IBM HTTP Server (Apache) style startup, log in as root and type the following command:

   ```
   /opt/IBMHTTPServer/bin/apachectl restart
   ```

2. For a Red Hat specific startup, log in as root and type the following command:

   ```
   /etc/rc.d/init.d/ibmhttpd restart
   ```

2.5.7 Verify the IBM HTTP Server

In order to verify the IBM HTTP Server installation, perform the following checks on the Web server machine:

- Check process status
- Check request handling

Check process status

1. Check that the HTTP Server processes are running by issuing the following command:

   ```
   ps -ef | grep httpd
   ```

   The output should list a number of httpd processes. The output should look similar to the output in Figure 2-4.

```
root     27585     1  0 Aug30 ?        00:00:00 /opt/IBMHTTPServer/bin/httpd
www      27586 27585  0 Aug30 ?        00:00:00 /opt/IBMHTTPServer/bin/httpd
www      27587 27585  0 Aug30 ?        00:00:00 /opt/IBMHTTPServer/bin/httpd
www      27588 27585  0 Aug30 ?        00:00:00 /opt/IBMHTTPServer/bin/httpd
www      27589 27585  0 Aug30 ?        00:00:00 /opt/IBMHTTPServer/bin/httpd
www      27590 27585  0 Aug30 ?        00:00:00 /opt/IBMHTTPServer/bin/httpd
www      28887 27585  0 Aug31 ?        00:00:00 /opt/IBMHTTPServer/bin/httpd
www      28888 27585  0 Aug31 ?        00:00:00 /opt/IBMHTTPServer/bin/httpd
www      28889 27585  0 Aug31 ?        00:00:00 /opt/IBMHTTPServer/bin/httpd
root     13372 13341  0 12:58 pts/5    00:00:00 grep httpd
```

Figure 2-4 Web server sample process listing

2. Check that the IBM HTTP Server is registered to listen on port 80 and is therefore ready to handle requests:

   ```
   netstat -an | grep LISTEN | grep :80
   ```

   The output should look similar to that in Figure 2-5.

```
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
```

Figure 2-5 Webserver sample socket details

3. If you configured the Web server for SSL, check that the IBM HTTP Server is registered to listen on port 443 and is therefore ready to handle requests:

   ```
   netstat -an | grep LISTEN | grep :443
   ```

   The output should look similar to that in Figure 2-6.

```
tcp        0      0 0.0.0.0:443 0.0.0.0:*               LISTEN
```

Figure 2-6 Web server sample socket details

Check request handling

Using a Web browser, request the following URL representing the IBM HTTP Server Web root for the home page:

```
http://<http_server_hostname>/
```

You should see the IBM HTTP Server home page displayed, as seen in Figure 2-7.

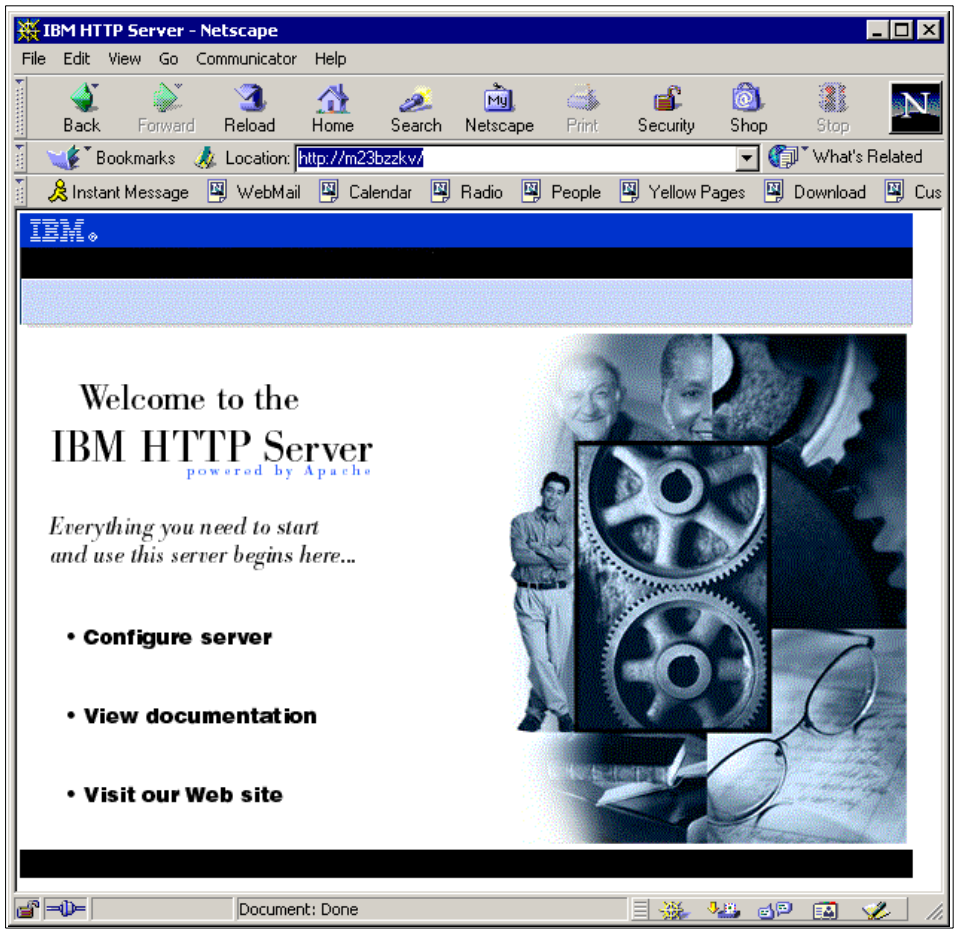


Figure 2-7 IBM HTTP Server home page verification

If your Web server is configured for SSL, repeat the previous step using the following URL:

```
https://<http_server_hostname>/
```

2.6 Configure and verify WAS

This section provides procedures to verify and configure the WebSphere Application Server V4.0.1 after the installation for the following topics:

- Verify the WebSphere Application Server installation
- Configure the WebSphere Application Server

- Configure WAS HTTP transport for SSL

2.6.1 Verify the WebSphere Application Server installation

In order to verify the installation of IBM WebSphere Application Server V4.0.1, Advanced Edition for Linux the following tasks should be completed:

- Check installation log
- Check admin.config settings
- Check Web server configuration file changes

Check the installation log

Check that the installation log <was_install_path>/logs/install.log does not contain any errors.

Check the admin.config settings

1. Check that the repository database settings are correct for the database type (DB2), instance (was) and user ID (<db2_instance_owner>) used in our test environment:

```
com.ibm.ejs.sm.adminServer.dbdataSourceClassName=COM.ibm.db2.jdbc.DB2Connec
tionPoolDataSource
com.ibm.ejs.sm.adminServer.dbserverName=
com.ibm.ejs.sm.adminServer.dbportNumber=
com.ibm.ejs.sm.adminServer.dbdatabaseName=<was database alias>
com.ibm.ejs.sm.adminServer.dbuser=<db2_instance_owner>
com.ibm.ejs.sm.adminServer.dbpassword=<db2owner_password>
com.ibm.ejs.sm.adminServer.dbdisable2Phase=true
```

2. Check that the following path-related parameters are set correctly:

Table 2-10 DB2-related paths required in admin.config

Parameter	Must contain path...
com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs	<db2_install_path>/sqllib/java12/db2java.zip

3. Check that the WAS schema and initial configuration (for example, Default Server) will be written to the repository database on startup:

Table 2-11 Required schema creation and initial configuration flags

Parameter	Required value...
com.ibm.ejs.sm.adminServer.createTables	true
install.initial.config	true

Check Web server configuration file changes

Verify that the Web server configuration file has been updated by completing the following steps:

1. Check that following required settings have been added to the IBM HTTP Server configuration file (httpd.conf) as a result of the WAS installation:

```
LoadModule ibm_app_server_http_module
/opt/WebSphere/AppServer/bin/mod_ibm_app_server_http.so
WebSpherePluginConfig /opt/WebSphere/AppServer/config/plugin-cfg.xml
AddModule mod_app_server_http.c
```

Important: If you have configured SSL as described in the procedure in 2.5.4, "Configure the HTTP Server for SSL" on page 40, then you will need to add the above configuration statements to the end of the httpd.conf file. This is because we copied the httpd.conf.sample containing the SSL specific statements to httpd.conf, which contained the above statements (copy from httpd.conf.org).

2. If not, manually add the above lines to the end of the httpd.conf file and save the changes.

2.6.2 Configure the WebSphere Application Server

This section describes how to configure the WebSphere Application Server.

Startup IBM HTTP Server processes

The IBM HTTP Server processes need to be started in order to serve content. There are two ways to start the Web server. The first way is Red Hat specific and the second is more generic. The purpose in showing the Red Hat specific way is that this is how the system starts the server at boot time. With other UNIXs, the server is started in other ways.

Start the Web server using one of the following two methods:

1. For a Red Hat specific startup:
   - Log in as root and run the following command:

```
/etc/rc.d/init.d/ibmhttpd start
```

2. For an IBM HTTP Server (Apache) style startup:
   - Log in as root and run the following command:

```
<http_install_path>/bin/apachectl start
```

Startup WAS Admin Server processes

The WAS Admin Server needs to be started in order to test the installation as well as the connectivity between WAS Admin Server, WAS repository database, and Administrator Console.

1. Log in as root, and start a terminal session.
2. Ensure that DB2 is started for the DB2 instance containing the WAS repository database. For example:

```
su - db2inst1
db2start
```

3. To start the WebSphere Admin Server, type the following commands:

```
cd /opt/WebSphere/AppServer/bin
./startupServer.sh
```

4. Verify that the startup of WAS Admin Server is successful, by reviewing the the following conditions:
   a. There are no Admin Server error logs in <was_install_path>/logs with names that start with __adminServer.
   b. The last line of the `<was_install_path>/logs/tracefile` file is similar to the following:

```
[01.07.05 11:28:01:591 EDT] 160042d2 Server I WSVR0023I: Server
__adminServer open for e-business
```

Check WAS process status

Check that the WebSphere Admin Server process is running by issuing the following command:

```
ps -ef --forest | grep java
```

The output should list a number of Java processes.

Note: In the event that more than one Java process is running unintentionally, you can stop the process by invoking the **`kill`** command.

```
kill <process_id>
```

Configure WAS virtual hosts

Many runtime environments include the Web server being configured to support SSL connections. In this case, we need to configure virtual hosts for the WebSphere Application Server.

1. Start the WebSphere Administrator Console as follows:

```
cd /opt/WebSphere/AppServer
./adminclient.sh
```

2. Select the **Virtual Hosts** folder.
3. Ensure that the virtual hosts are listed as seen in Table 2-12. If not, click **Add** to add them, and then click **Apply**.

Table 2-12 WAS virtual hosts

Virtual host	Example
<ip_address>	9.24.105.134
<hostname>	ganci5
<fully_qualified_hostname>	ganci5.itso.ral.ibm.com
<ip_address>:443	9.24.105.134:443
<hostname>:443	ganci5:443
<fully_qualified_hostname>:443	ganci5.itso.ral.ibm.com:443

Startup WAS Default Server

The WAS installation sets up a default application server (Default Server) in the WAS administrative domain. This Application Server and its servlets are used to verify that the WAS installation is working correctly.

To start up the Default Server, perform the following steps:

1. Start the WebSphere Administrative Console by issuing the following commands:

```
cd <was_install_path>/bin
./adminclient.sh
```

2. Select and expand **WebSphere Administrative Domain -> Nodes -> <your_hostname> -> Application Servers**.
3. Select **Default Host**, and then right-click **Start**, if not already started.
4. The startup of Default Server is successful if the following conditions are met:
   a. The AdminConsole event messages pane shows the lines:

```
Command "Default Server.start" completed successfully.
Transport http listening on port 9080.
```

5. Verify that the Default Server Web container has been properly installed and configured by accessing its servlets through the Web server "embedded" within the WAS V4 Web container:
   a. Using a Web browser, request the following URL:

```
http://<hostname>:9080/servlet/snoop
```

   A page similar to the one shown in Figure 2-8 should be displayed.

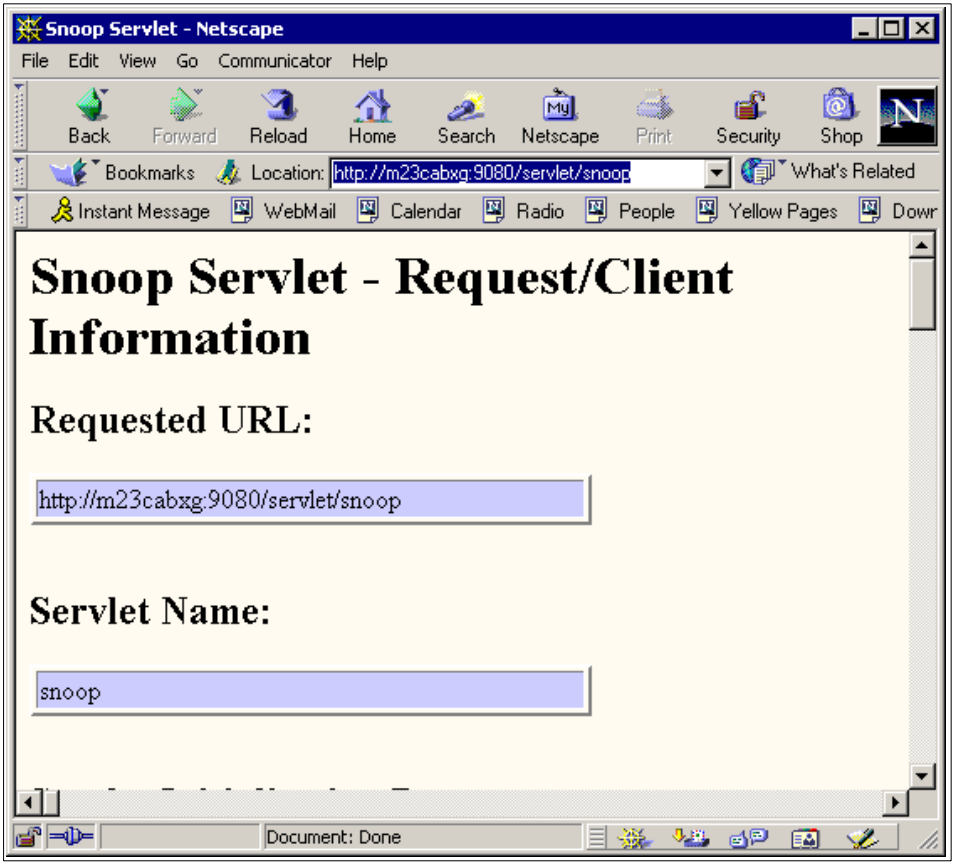


Figure 2-8 Snoop servlet accessed through embedded Web server

b. Using a Web browser, request the following URL:

```
http://<was_server_hostname>:9080/webapp/examples/showCfg
```

Note: The embedded Web server is a new feature introduced with WAS V4.0. In previous releases, a stand-alone Web server was required in order to access any resource hosted in WAS.

Regenerate Web server plug-in settings

Before the Default Server can be accessed from a stand-alone Web server (for example, the IBM HTTP Server) the Web server plug-in settings file <was_install_path>/config/plugin-cfg.xml must be regenerated to reflect the following settings used by the Web server plug-in:

- Virtual host settings

- Application Server transports
- Web Container URIs

Perform the following steps:

1. If you exited the WebSphere Administration Console:
   a. Log in as root, and start a terminal session.
   b. Run the WAS GUI AdminConsole by issuing the following command:

```
cd /opt/WebSphere/AppServer/bin
./adminclient.sh
```

2. Select and expand **WebSphere Administrative Domain -> Nodes**.
3. Select **<your_hostname>**, right-click and select **Regen Webserver Plugin** (see Figure 2-9).

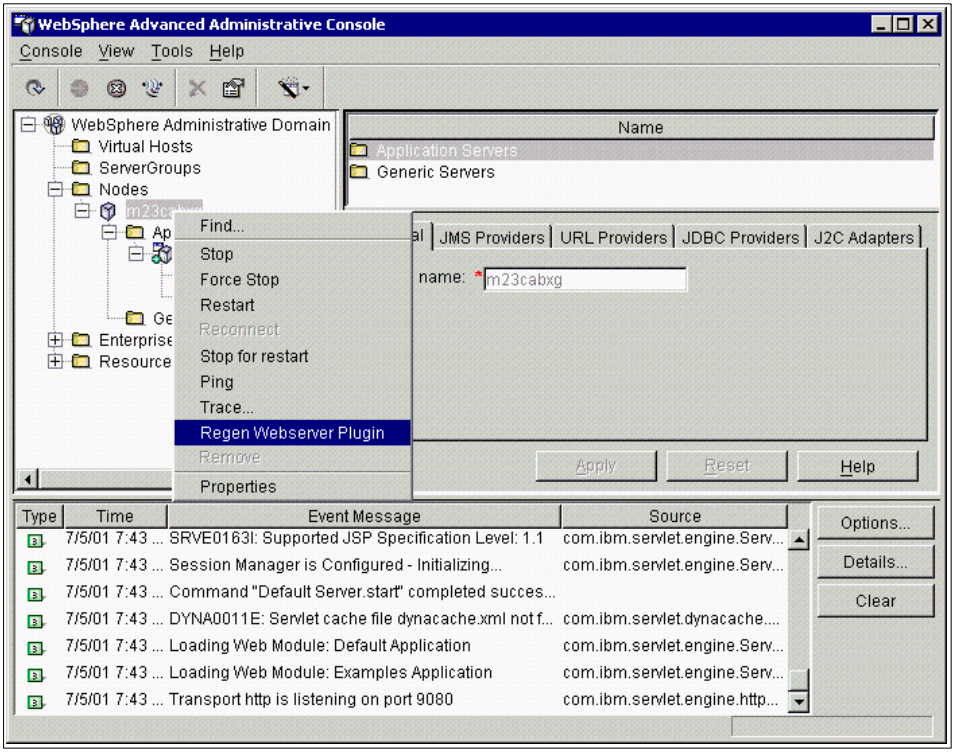


Figure 2-9 Regenerate Web server plug-in settings

Tip: WAS provides a command-line tool that can be used to regenerate the Web server plug-in configuration without having to run the GUI AdminConsole:

```
/opt/WebSphere/AppServer/bin/GenPluginCfg.sh -adminNodeName <hostname>
```

4. Check that the content of the <was_install_path>/config/plugin-cfg.xml file has been updated to include the URIs of servlets contained within Default Server.

Tip: The plug-in regeneration command generates a <Server> element for Default Server that contains a CloneID attribute:

```
<Server CloneID="stsul7n0" Name="Default Server">
   <Transport Hostname="m23wpn18" Port="9080" Protocol="http"/>
</Server>
```

In a non-cloned environment this attribute can be removed, resulting in performance improvements by the Web server plug-in.

Restart Web server processes

The IBM HTTP Server process must be restarted before the Web server plug-in configuration can be tested.

1. Log in as root, and start a terminal session.
2. Issue the following command:

   ```
   /etc/rc.d/init.d/ibmhttpd restart
   ```

   Or you could also use the generic form:

   ```
   /opt/IBMHTTPServer/bin/apachectl restart
   ```

Verify Web server plug-in and WAS configuration

The Web server plug-in configuration can be verified by requesting a servlet through the Web server that has already been successfully requested through the Web container's embedded Web server:

1. Using a Web browser, request the following URL:

   ```
   http://<web_server_hostname>/webapp/examples/showCfg
   ```

2.6.3 Configure WAS HTTP transport for SSL

This section describes a specific example of setting up a client authentication enabled SSL transport for the WebSphere Application Server. In this setup both the server (WAS) and the client (plug-in) are required to send a certificate to each other to authenticate themselves during SSL handshaking.

Note: For more detailed information, refer to “Configure WAS HTTP transport for SSL” section in the *WebSphere V4 Advanced Edition Handbook*, SG24-6176 (currently a redpiece).

The configuration involves the following tasks:

1. Create a WebSphere Application Server key store
2. Create a server self-signed certificate
3. Export the server certificate
4. Create a plug-in key store
5. Create a client self-signed certificate
6. Export the client certificate
7. Import the server’s certificate as a trusted CA
8. Import the clients’s certificate as a trusted CA
9. Configure a new transport to use SSL encryption
10. Regenerate Web server plug-in configuration file
11. Restart the Web server
12. Verify WAS

Note: The commands in the following steps all assume that you are logged in to the server as the root user and have started up a command shell.

Create a WebSphere Application Server key store

The WebSphere Application Server key store is a JKS formatted key database file. The following outlines how to create this file using the ikeyman.sh script/utility:

1. Run the ikeyman.sh script:

```
cd /opt/WebSphere/AppServer/etc
../bin/ikeyman.sh
```

2. Select **Key Database File** from the menu bar. Select **New**, enter the following, and then click **OK**:

- Key Database Type: `JKS`
- File Name: `server.jks`
- Location: `/opt/WebSphere/AppServer/etc/`

3. When the Password Prompt window appears, enter the following and then click **OK**:
   - Password: `<your key file password here>`
   - Confirm Password: `<your key file password here>`
   - Expiration Time: `<optional if on a production server)`

Create a server self-signed certificate

In a production environment, you would likely want to create a certificate request and obtain a valid certificate through a well-known Certificate Authority. For the purposes of our example, the following steps can be taken to generate a self-signed certificate for the server (WebSphere Application Server) to authenticate itself with the client (plug-in):

1. If you exited the server key store, reopen it as follows:

   ```
   cd <was_install_path>/etc
   ../bin/ikeyman.sh
   ```

   You will need to open the server.jks file you created in the previous section.

2. From the IBM Key Management utility menu bar, select **Create -> New Self-Signed Certificate** and enter the following (or fill in your own information) and then click **OK**:
   - Key Label: `SampleServerCertificate`
   - Version: `X509 V3`
   - Key Size: `1024`
   - Common Name: `ganci5.itso.ral.ibm.com`
   - Organization: `IBM`
   - Organizational Unit: `ITSO`

The created certificate will be listed in the Personal Certificates section of the ikeyman utility.

Export the server certificate

In order to permit authentication between the plug-in and the WebSphere Application Server, each key store needs to recognize the other's certificate as a signing authority or certificate authority. To do this the certificates need to be extracted.

To export the server certificate, complete the following steps:

1. If you exited the server key store, reopen it as follows:

   ```
   cd /opt/WebSphere/AppServer/etc
   ../bin/ikeyman.sh
   ```

   You will need to open the server.jks file you created in previous sections.

2. Select the **SampleServerCertificate** created in the previous section and then select **Extract Certificate**.
3. When the Extract Certificate to a File window appears, enter the following and then click **OK**.
   - Data Type: `Base64-encoded ASCII data`
   - Certificate file name: `server-cert.arm`
   - Location: `/opt/WebSphere/AppServer/etc/`
4. Close the IBM Key Management utility.

Create a plug-in key store

The plug-in (client) key store is a CMS formatted key database file. The following outlines how to create this file using the ikeyman utility:

1. Run the ikeyman script:

   ```
   cd /opt/WebSphere/AppServer/etc
   export JAVA_HOME=/opt/WebSphere/AppServer/java
   /usr/bin/ikeyman
   ```

   Note: This is a different version of the ikeyman utility, which stores the key in a CMS formatted database file.

2. Select **Key Database File** from the menu bar. Select **New**, enter the following, and then click **OK**.
   - Key Database Type: `CMS key database file`
   - File Name: `plugin-key.kdb`
   - Location: `/opt/WebSphere/AppServer/etc/`
3. When the Password Prompt window appears, enter the following and then click OK.
   - Password: `<your key file password here>`
   - Confirm Password: `<your key file password here>`
   - Expiration Time: (optional if on a production server)
   - Stash Password: **YES** - be sure to check this option.

Create a client self-signed certificate

In a production environment, you would likely want to create a certificate request and obtain a valid certificate through a well known Certificate Authority. For the purposes of our example, the following steps can be taken to generate a self-signed certificate for the client (plug-in) to authenticate itself with the server (WebSphere Application Server):

1. If you exited the client key store, reopen it as follows:

   ```
   cd <was_install_path>/etc
   export JAVA_HOME=<was_install_path>/java
   /usr/bin/ikeyman
   ```

   You will need to open the plugin-key.kdb file you created in the previous section.

2. From the IBM Key Management utility menu bar, select **Create -> New Self-Signed Certificate**, enter the following (or fill in your own information) and then click **OK**.

   – Key Label: `SampleClientCertificate`

   – Version: `X509 V3`

   – Key Size: `1024`

   – Common Name: `ganci5.itso.ral.ibm.com`

   – Organization: `IBM`

   – Organizational Unit: `ITSO`

The created certificate will be listed in the Personal Certificates section of the ikeyman utility.

Export the client certificate

In order to permit authentication between the plug-in and the WebSphere Application Server, each key store needs to recognize the other's certificate as a Signing Authority or Certificate Authority. To do this the certificates need to be extracted. Follow the following procedures to export the client certificate:

1. If you exited the client key store, reopen it as follows:

   ```
   cd <was_install_path>/etc
   export JAVA_HOME=<was_install_path>/java
   /usr/bin/ikeyman
   ```

   You will need to open the plugin-key.kdb file you created in previous sections.

2. Select the **SampleClientCertificate** created in the previous section and click **Extract Certificate**.

3. When the Extract Certificate window appears, enter the following and then click **OK**:
   - Data Type: `Base64-encoded ASCII data`
   - Certificate file name: `plugin-cert.arm`
   - Location: `/opt/WebSphere/AppServer/etc/`

Import the server's certificate as a trusted CA

Import the server's certificate into the plug-in (client) key store by following these steps:

1. If you exited the client key store, reopen it as follows:

   ```
   cd <was_install_path>/etc
   export JAVA_HOME=/opt/WebSphere/AppServer/java
   /usr/bin/ikeyman
   ```

   You will need to open the plugin-key.kdb file you created in previous sections.
2. From the Key database content pull-down (default Personal Certificates) select **Signer Certificates**, and then click the **Add**.
3. When the Add CA's Certificates from a File window appears, enter the following and then click **OK**.
   - Data Type: `Base64-encoded ASCII data`
   - Certificate file name: `server-cert.arm`
   - Location: `/opt/WebSphere/AppServer/etc/`
4. When prompted, enter the label for the certificate, type `SampleServerCertificate` and then click **OK**.

   The server's certificate should be listed with the other signer certificates.
5. Close the current IBM Key Management utility.

Import the clients's certificate as a trusted CA

Import the plug-in's (client) certificate into the server (WebSphere Application Server) key store by following these steps:

1. If you exited the server key store, reopen it as follows:

   ```
   cd /opt/WebSphere/AppServer/etc/
   ../bin/ikeyman.sh
   ```
2. Open the server.jks file you created in previous section, by selecting **Key Database File** from the menu bar.
3. When the Open window appears, enter the file name `server.jks` and then click **OK**. When prompted for the password, enter the certficate password for

the `SampleServerCertificate` created in “Create a WebSphere Application Server key store” on page 53.

4. From the Key database content pull-down (default Personal Certificates) select **Signer Certificates**, and then click the **Add**.
5. When the Add CA’s Certificates from a File window appears, enter the following and then click **OK**.
   - Data Type: `Base64-encoded ASCII data`
   - Certificate file name: `plugin-cert.arm`
   - Location: `/opt/WebSphere/AppServer/etc/`
6. When prompted, enter the label for the plug-in certificate. For example, `SampleClientCertificate`.

   The client’s certificate should now be listed with the other signer certificates.
7. Close the IBM Key Management utility.

Configure a new transport to use SSL encryption

In order to use the new certificates and set up SSL encryption on a WebSphere Application Server transport, the following steps need to be completed:

Note: For more detailed information, refer to the *WebSphere V4 Advanced Edition Handbook*, SG24-6176 redbook.

1. Start up the WebSphere Administrative Console as follows:

```
cd /opt/WebSphere/AppServer/bin
./adminclient.sh
```

2. Ensure that the Default Server is stopped.
3. Select the **Services** tab of the Default Server configuration page.
4. Select the **Web Container Service**, and then click **Edit Properties**.
5. When the Web Container Service window appears, select the **Transport** tab and click **Add.**
6. When the HTTP Transport Properties window appears, enter the following and then click **OK**.
   - Transport Host: `*`
   - Transport Port: `9081`
   - Check **Enable SSL**
   - Do not check Use global SSL (default)
   - Key file name: `/opt/WebSphere/AppServer/etc/server.jks`

- Key file password: `<your password>`
- Confirm password: `<your password>`
- Key file format: `JKS`
- Trust file name: `<was_install_path>/etc/server.jks`
- Trust file password: `<your password>`
- Confirm password: `<your password>`
- Trust file format: `JKS`
- Check **Enable client authentication**
- Security level: `HIGH`

7. Click **OK**.
8. Once the above entry has been added, be sure to click **Apply** to save the changes to the Default Server.
9. Restart the Default Server by selecting the **Default Server** and right-click **Start**.

Regenerate Web server plug-in configuration file

Follow the steps in“Regenerate Web server plug-in settings” on page 50 to regenerate the Web server plug-in file.

Restart the Web server

Follow the steps in“Restart Web server processes” on page 52 to restart the IBM HTTP Server.

Verify WAS

Follow the steps in 2.6.1, “Verify the WebSphere Application Server installation” on page 46.

In summary, make sure the following URLs entered in a Web browser work correctly:

- `http://<hostname>/servlet/snoop`
- `https://<hostname>/servlet/snoop`
- `http://<hostname>/webapp/examples/showCfg`
- `https://<hostname>/webapp/examples/showCfg`

3

WAS V4 for Linux 2-tier runtime environment

This chapter provides detailed instructions for implementing a WebSphere Application Server V4.0.1, Advanced Edition for Linux in a 2-tier runtime environment.

In the 2-tier architecture, there are two possible scenarios for the location of the required software components. Figure 3-1 on page 62 depicts the following two scenarios to implement a 2-tier runtime environment:

1. Remote Web Server: [IHS] and [DB2 + WAS]

   In this scenario the Web Server is installed on machine A, and the Database Server and the WebSphere Application Server are installed on a separate system, machine B.

2. Remote Database Server: [IHS + WAS + DB2 Client] and [DB2 Server]

   In this scenario the Web Server, WebSphere Application Server, and database client are installed on machine A, and the database server is installed on a separate system, machine B.

3.1 Planning for a WAS V4 for Linux 2-tier runtime

This section defines the hardware and software used within the IBM WebSphere Application Server V3.5, Advanced Edition 2-tier scenarios (Remote Web Server, Remote Database Server).

This section includes the following topics:

- Overview
- Hardware and software prerequisites
- Hardware used within our test environment
- Software used within our test environment
- Install Red Hat Linux

3.1.1 Overview

The 2-tier environment can be built in one of two ways depending on the security and business requirements you may have. The first arrangement puts Web server on one machine, known as a remote Web server, and application and database server on a separate system from the Web server. This allows for a firewall to be placed in between and keeps all application code and data behind the firewall. The potential downside to this configuration is that if the application is database intensive CPU performance may not be what it could be if the database and application server are running on separate servers.

The second scenario puts the Web server and application server on one system and the database on its own server, known as a remote database server. This still allows for a firewall to be implemented to protect application data; however, the application code is more vulnerable to intruders.

It is up to the reader to decide which configuration best suits their needs. The choice will require looking at factors such as those given above, size and capacity of the available systems, and the characteristics of the Web application. If it makes heavy use of databases it might make sense to isolate the database server to give it as much power as possible. If the application is computation intensive with a lot of static content, it might be better to separate the Web server and application server.

The following sections provide instructions on how to implement both runtime environment scenarios, as well as the hardware and software we used in our test environment.

3.1.2 Hardware and software prerequisites

The prerequisites for the 2-tier environment are the same as for the single-tier environment in “Hardware and software prerequisites” on page 6.

The only significant difference between the hardware and software prerequisites for 1-tier and 2-tier environments are that the disk space requirements will be divided up accordingly between the two servers.

3.1.3 Hardware used within our test environment

We used the following systems within our 2-tier environments:

- Web server:
  - Hostname: ganci6
  - IP Address: 9.24.105.101
  - IBM Netfinity 3000 (8476-21U)
    - 300 MHz CPU (Pentium II)
    - 512 MB RAM
    - 4 GB hard disk
    - 1 IBM Ethernet Adapter
    - 1 Integrated S3 Trio-3D Video Adapter
- Database server:
  - Hostname: ganci5
  - IP Address: 9.24.105.134
  - IBM NetVista (6579-A4U)
    - 866 MHz CPU
    - 512 MB RAM
    - 30 GB hard disk
    - 1 IBM Ethernet Adapter
    - 1 Integrated Intel 815e Solano Video Adapter

3.1.4 Software used within our test environment

The software used in the 2-tier environments is the same as that used in the single-tier environment as shown in “Software used in our test environment” on page 8.

3.1.5 Install Red Hat Linux

Both systems, database server and Web server, should be installed with Red Hat Linux V7.1 following the instructions in “Red Hat Linux installation” on page 11. In our setup we configured the same file systems on both servers so that we could facilitate both 2-tier scenarios without having to reinstall/reconfigure the Linux file systems. In a real-world application one scenario would be selected and the appropriate file systems created on each server and not necessarily on both.

3.2 Implementing the Remote Web Server 2-tier

This scenario is built as shown in Figure 3-2 with the Web server on machine A, and the database and application servers on machine B.

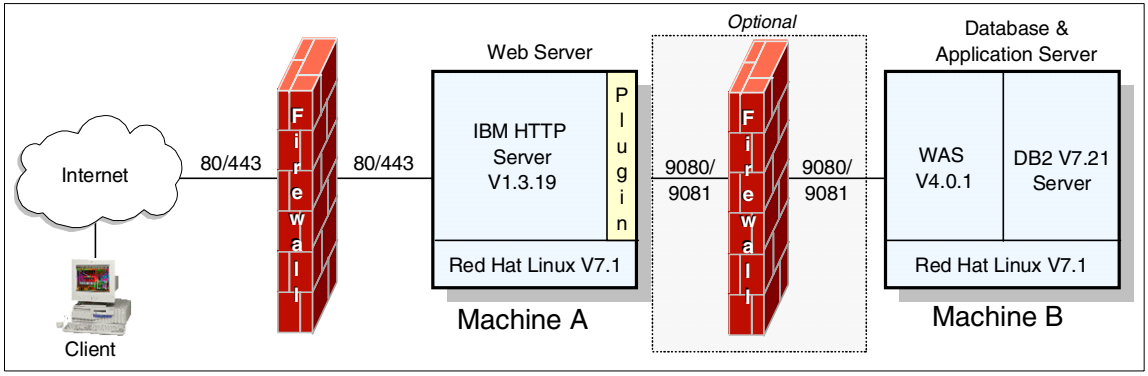


Figure 3-2 Remote Web Server 2-tier

This section includes the following topics:

- Install the DB2 Server
- Install IBM WebSphere Application Server V4.0.1
- Install IBM HTTP Server V1.3.19
- Update the WebSphere plug-in file
- Verify WebSphere plug-in configuration

3.2.1 Install the DB2 Server

Refer to the steps in “Install the DB2 Server” on page 13 to install the DB2 server software on the database server system (machine B).

3.2.2 Install IBM WebSphere Application Server V4.0.1

In this scenario, the installation of the application server is the same as in the single-tier installation (see “Install the WebSphere Application Server” on page 28) except that the IBM HTTP Server and the WebSphere Plug-in need to be deselected during the custom installation step.

Perform the above installation on machine B, as seen in Figure 3-2 on page 65.

3.2.3 Install IBM HTTP Server V1.3.19

This section provides detailed instructions for installing, configuring and verifying IBM HTTP Server V1.3.19 for Linux on the Web server machine (machine A).

The section includes the following tasks:

- Pre-installation tasks
- Install IBM HTTP Server
- Configure IBM HTTP Server
- Verify IBM HTTP Server

Pre-installation tasks

Prior to installing the IBM HTTP Server, the following checks and tasks must be completed:

- Check that there are no existing services on the server that use the following IP ports:
  - 80 (standard HTTP port)
  - 443 (standard HTTPS port)
  - 8008 (IBM HTTP Server Administration port)

  We suggest using the following command for this task:

```
netstat -an | grep LISTEN
```

Install IBM HTTP Server

To install the IBM HTTP Server, complete the following steps:

1. Log in as root, and start a terminal session.
2. Mount the CD-ROM or change to the directory where WebSphere has been downloaded and expanded:
   a. Insert the WebSphere CD into CD-ROM drive.

b. If the CD doesn't automatically mount (as it may if you're running X Windows with either the KDE or Gnome desktops) it can manually be mounted by issuing the following command as root:

```
mount /mnt/cdrom
```

This should work for a stock installation. If it fails, verify the device that represents your CD-ROM drive and issue the following command instead:

```
mount -r /dev/<device> /mnt/cdrom
```

For instance, on our system, the CD-ROM drive is the first device on the second IDE channel. Thus it is (in Linux) referred to as hdc and can be mounted by the following command:

```
mount -r /dev/hdc /mnt/cdrom
```

3. Change to the IBM HTTP Server installation directory of the CD:

```
cd /mnt/cdrom/ihs_128
```

Note: If you're installing from a file system, simply substitute the root of that file system/directory for /mnt/cdrom.

4. Install each of the packages listed in Table 3-1 as follows:

```
rpm -U --nodeps <Package>
```

Where <Package> is exactly as listed in Table 3-1. For example:

```
rpm -U --nodeps gsk5bas-5.0.3.61.i386.rpm
```

Note: The **rpm** command will not return any output if it was successful, but will if there are any problems. If greater verbosity is desired , the following command could be used instead:

```
rpm -Uvh --nodeps <Package>
```

Table 3-1 Required packages for the IBM HTTP Server

Package	Description
gsk5bas-5.0-3.61.i386.rpm	GSK certificate and security libraries
IBM_ADMIN_EN-1.3.19-0.i386.rpm[a]	IBM Administration Server documentation
IBM_ADMIN_Server-1.3.19-0.i386.rpm[a]	IBM Administration Server program files
IBM_FastCGI-1.3.19-0.i386.rpm	Implementation of FastCGI standard - increases CGI performance
IBM_HTTP_Server-1.3.19-0.i386.rpm	IBM HTTP Server program files

Package	Description
IBM_MAN_ENU-1.3.19-0.i386.rpm	HTTP Server manual (man) pages
IBM_MSG_EN-1.3.19-0.i386.rpm	Language message files for IBM HTTP Server
IBM_SSL_128-1.3.19-0.i386.rpm	IBM SSL (Secure Sockets Layer) 128bit library
IBM_SSL_Base-1.3.19-0.i386.rpm	IBM SSL (Secure Sockets Layer) program files
IBM_SSL_EN-1.3.19-0.i386.rpm	Language message files for IBM SSL
a. Optional RPM. This is part of the IHS HTML administration tool which is not used very much in production environments.	

Important: Double-check that you have selected the correct packages. Due to the number of selections, it is easy to make a mistake that will result in the IBM HTTP Server not working properly.

5. Unmount the CD-ROM:

```
cd /
umount /mnt/cdrom
```

Configure IBM HTTP Server

After the installation of IBM HTTP Server there are a few tasks that need to be completed to configure the Web server. Refer to “Configure and verify the IBM HTTP Server” on page 37 for details on configuring the Web server.

Verify IBM HTTP Server

Follow the steps in “Verify the IBM HTTP Server” on page 43 to verify the Web server configuration.

3.2.4 Update the WebSphere plug-in file

One main requirement when separating the WebSphere Application Server from the IBM HTTP Server on separate machines is that the WebSphere plug-in has to be transferred to the Web server after being (re)generated on the application server.

1. If not already done during the WebSphere Application Server installation, regenerate the plug-in as in “Regenerate Web server plug-in settings” on page 50.

2. Log in as root on the Database server, ganci5, and run the following commands:

```
export WASCFG=/opt/WebSphere/AppServer/config
scp $WASCFG/plugin-cfg.xml <webserver>:$WASCFG/plugin-cfg.xml
```

Note: You can substitute your favorite file transfer mechanism/program for `scp` in the above command. The advantage of using `scp` is security and the ability to pass through firewalls. The **`scp`** command uses the SSH protocol which is an encrypted protocol. Most corporate firewalls will block FTP access to their public Web servers, but will allow SSH traffic.

For example, on our 2-tier system we used the following commands:

```
export WASCFG=/opt/WebSphere/AppServer/config
scp $WASCFG/plugin-cfg.xml ganci6:$WASCFG/plugin-cfg.xml
```

Tip: In the above commands we set up a variable, WASCFG, to temporarily store the path to the plugin-cfg.xml file. This was purely to make the above commands look more readable in this document. There is no reason you couldn't substitute the actual path for the `$WASCFG` portion in the **`scp`** commands above and drop the **`export`** command altogether.

3. Log in as root on the Web server, ganci6, and restart the IBM HTTP Server to ensure the plug-in is re-read:

```
/etc/rc.d/init.d/ibmhttpd restart
```

3.2.5 Verify WebSphere plug-in configuration

The Web server plug-in configuration can be verified by requesting a servlet through the Web server that will be served by the WebSphere Application Server on the Database server:

1. Using a Web browser, enter the following URL:

```
http://<Webserver>/webapp/examples/showCfg
```

For example, in our case we used:

```
http://ganci6/webapp/examples/showCfg
```

3.3 Implementing the Remote Database Server 2-tier

In this scenario, as seen in Figure 3-3, the Web and application server are installed on machine A, and the database server installed on a separate system, machine B.

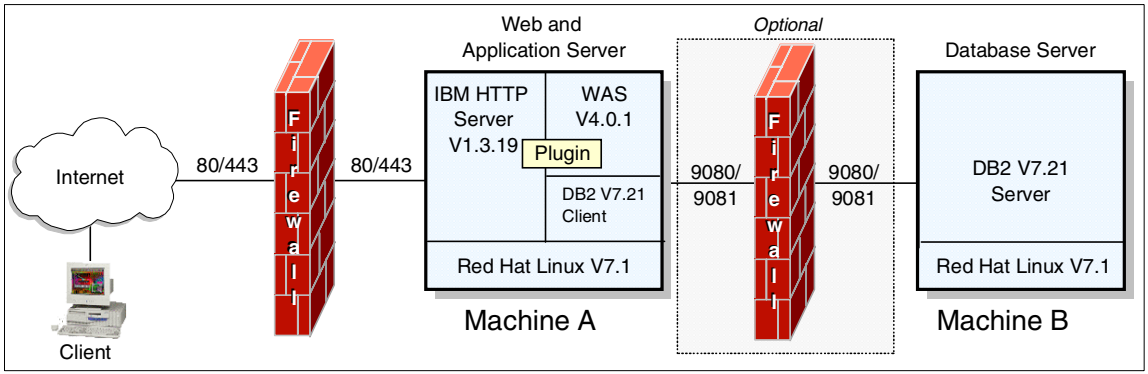


Figure 3-3 Remote Database Server 2-tier

This section includes the following tasks:

- Install the DB2 Server
- Install the DB2 Client
- Configure the DB2 Client
- Install and configure the WebSphere Application Server
- Configure and verify IBM HTTP Server

3.3.1 Install the DB2 Server

On the Database Server, machine B, install the DB2 Server. Refer to “Install the DB2 Server” on page 13 for detailed instructions.

3.3.2 Install the DB2 Client

The steps for installing the DB2 Client software are very similar to the DB2 Server installation, but the steps are outlined here for clarity.

The section is organized into the following tasks:

- Prerequisite Linux packages for DB2
- Pre-installation tasks
- Install the DB2 Client
- Verify DB2 Client installation

Prerequisite Linux packages for DB2

The prerequisites for the DB2 Client software are the same as for the DB2 Server software. Please see “Pre-requisite Linux packages for DB2” on page 14.

Pre-installation tasks

Prior to installing the DB2 Client the following check needs to be completed:

- Verify that there are no existing active services that use the same DB2 TCP/IP ports on the server:
  - 523 (DB2 Server)
  - 50000 (DB2 instance connection port)
  - 50001 (DB2 instance interrupt port)
  - 50002 (DB2 Control Server)

  We suggest using the following command for this task:

  ```
  netstat -an | grep LISTEN
  ```

Install the DB2 Client

In order to install IBM DB2 Universal Database V7.2.1, Enterprise Edition for Linux, perform the following steps:

1. Log in as root.
2. Start a terminal session.
3. Mount the DB2 V7.2.1 CD-ROM:

   ```
   mount /mnt/cdrom
   ```

Note: If you are running X Windows with either the KDE or Gnome window managers the CD-ROM may automatically be mounted for you. To verify this, simply do a **`mount | grep /mnt/cdrom`** and if you get any output then the CD-ROM has already been mounted.

4. Start the DB2 installer program:

   ```
   cd /mnt/cdrom
   ./db2setup
   ```

5. In the Install DB2 V7.2.1 window, select only the following options:
   - DB2 Administration Client

Tip: Navigation tips

- Press Tab to move between available options and fields.
- Press Ctrl+L to refresh the window at any point.
- Highlight and press Enter to select an option.

6. Highlight the DB2 Product Library's **Customize** option and press Enter.

7. In the DB2 Product Library window, highlight the appropriate option for your locale under the DB2 Product Library (HTML) section, then highlight **OK** and press Enter.
8. Highlight **OK** and press Enter.
9. In the Create DB2 Services window, select the **Create a DB2 Instance** option, highlight **OK** and press Enter.
10. In the DB2 instance authentication window appears, enter the following:
    - User Name: `<db2_instance_owner>`
    - User ID: `<use default UID>`
    - Group Name: `db2iadm1`
    - Group ID: `<use default GID>`
    - Home Directory: `/home/<db2_instance_owner>`
    - Password: `<user_password>`
    - Verify Password: `<user_password>`

Important: The DB2 installer uses the above information to automatically perform the following operations:

- Create a group `db2iadm1`
- Create a user `<db2_instance_owner>` with primary group `db2iadm1`
- Sets `<db2_instance_owner>` password to `<user_password>` value. The password used must meet DB2 requirements: eight characters or less and not containing the characters “<“ or “>”.

Changes the ownership (owner:group) of the `/home/<db2_instance_owner>` directory to be `<db2_instance_owner>:db2iadm1`

11. Highlight **OK** and press Enter.
12. Again, highlight **OK** and press Enter.
13. The Summary Report window is displayed, listing the product components to be installed. Highlight **Continue** and press Enter.
14. A warning window appears indicating this is your last chance to stop. Highlight **OK** and press Enter.

    The db2setup program installs the selected components. Depending on the speed of your processor, this can take up to 15 minutes.
15. You may be prompted to register the product. Complete the registration then go back to the install window.

16. When the install completes, a notice window informs you whether the installation was successful. Highlight **OK** and press Enter.
17. Scan the Status Report to ensure that all components were installed successfully. Highlight **OK** and press Enter.
18. In the DB2 Installer window, highlight **Close** and press Enter.
19. A window appears asking `Do you want to exit the DB2 Installer?` Highlight **OK** and press Enter.
20. The DB2 installation is now complete.
21. Unmount the CD-ROM:

```
cd /
umount /mnt/cdrom
```

Verify DB2 Client installation

To verify the DB2 Server installation, complete the following tasks:

- Check home directory permissions.
- Check DB2 instance owner profile.
- Check DB2 instance symbolic links.
- Check DB2 release level.

Check home directory permissions

Check that the home directory ownership has been correctly set up by the db2setup program:

Table 3-2 DB2 home directory required permissions

Home directory path	Owner	Group	Permissions[a]
/home/<db2_instance_owner>	<db2_instance_owner>	db2iadm1	drwxr-xr-x

a. The permissions need to be such that the Owner can read, write, and execute files and directories within the path. Group members' and other users' access rights are up to each company's security policies and business needs

If a DB2 related home directory has not been correctly configured, perform the following steps:

1. Log in as root, and start a terminal session.
2. Issue the command, substituting values from Table 3-2:

```
chown -fR <owner>:<group> <home_directory_path>
```

Check DB2 instance owner profile

The DB2 Server installation should set up the .bashrc environment file of the <db2_instance_owner> so that the DB2 environment is set up when the user logs in.

1. The following content should have been added to the file:

```
if [ -f ~/sqllib/db2profile ] ; then
. ~/sqllib/db2profile
fi
```

2. If not present, manually edit the file to add the above content.

Check DB2 instance symbolic links

The DB2 Server installation automatically creates a DB2 instance <db2_instance_owner> under the /home/<db2_instance_owner> directory. As part of the instance creation, db2setup should create symbolic links in the /home/<db2_instance_owner>/sqllib directory to files under /usr/IBMdb2/V7.1.

Perform the following steps to check whether the symbolic links have been created:

1. Log in as root, and start a terminal session.
2. Change directory to /home/<db2_instance_owner>/sqllib.
3. Check whether a number of symbolic links exist pointing to files under /usr/IBMdb2/V7.1.
4. If not, issue the following commands:

```
cd /usr/IBMdb2/V7.1/cfg
./db2ln
```

Check DB2 release level

Check that DB2 has the correct internal release level to meet WAS requirements:

1. Change to user <db2_instance_owner>:

```
su - <db2_instance_owner>
```

2. Enter the following command:

```
db2level
```

   This should generate output similar to the following:

```
DB21085I  Instance "db2inst1" uses DB2 code release "SQL07021" with level
identifier "03020105" and informational tokens "DB2 v7.1.0.43", "s010504"
and "U475381a"
```

   An internal release level of 7.1.0.43 should be indicated.

3.3.3 Configure the DB2 Client

After the DB2 Client installation, a number of configuration tasks must be performed so that WAS is able to use it as the repository for its administration database:

1. Update root administrative groups
2. Update JDBC level
3. Verify DB2 environment
4. Update root environment file
5. Configure the DB2 Client and server connectivity
6. Verify the DB2 Client and server connectivity

Update root administrative groups

The DB2 Server installation should add the db2asgrp Administrative group to the root user.

Perform the following steps to check whether the root account's administrative groups have been amended:

1. Log in as root.
2. Start a terminal session.
3. Issue the following command:

   ```
   groups
   ```

4. If db2asgrp is not listed as one of the groups assigned to root, you can use the Red Hat GUI tools to reconfigure the root user or you can do the following:

   ```
   vigr
   ```

   This will bring you into a vi editing session with a locked version of the /etc/group file. Find the line in the file that starts with 'db2asgrp:' and add the user, root, to the end of it (separated by a comma from other user names). The line should look similar to this after editing:

   ```
   db2asgrp:db2as,root
   ```

   Once you have added that, press ESC+:wq to save and quit vi. The system will then prompt you to edit the shadow group file. Choose Yes and do the same thing to that file - saving and quitting at the end of your editing.

Update JDBC level

IBM WebSphere Application Server 4.0 requires the use of JDBC2.0, whereas the default installation of IBM DB2 V7.2.1 uses JDBC1.2. To update the DB2 JDBC level, complete the following steps:

1. Change to user <db2_instance_owner>:

   ```
   su - <db2_instance_owner>
   ```

2. Add the following content to the end of the <db2_instance_owner> .bashrc environment file:

   ```
   if [ -f ~/sqllib/java12/usejdbc2 ] ; then
   . ~/sqllib/java12/usejdbc2
   fi
   ```

   Note: We found that the usejdbc2 file on our system was missing a semicolon, ';', on lines 32 and 36 after the right brace, ']', and before the word 'then'. The corrected lines look like:

   ```
   Line 32: if [[ `uname` = "AIX" ]]; then

   Line 36: elif [[ `uname` = "HP-UX" ]]; then
   ```

Verify DB2 environment

After the above configuration steps, we need to check that the environment being set up by the db2profile and usejdbc2 scripts is correct:

1. Change to user <db2_instance_owner>:

   ```
   su - <db2_instance_owner>
   ```

2. Issue the following command:

   ```
   set | grep [Dd][Bb]2
   ```

3. Check that the environment variables in this output match the values in the table below:

Table 3-3 DB2 Server required environment variables

Environment variable	Required value
DB2COMM	tcpip
DB2DIR	/usr/IBMdb2/V7.1
DB2INSTANCE	<db2_instance_owner>
INSTHOME	/home/<db2_instance_owner>
LD_LIBRARY_PATH	:/home/<db2_instance_owner>/sqllib/lib
LIBPATH	.....:/home/<db2_instance_owner>/sqllib/java12:/home/<db2_instance_owner>/sqllib/lib
CLASSPATH	/home/<db2_instance_owner>/sqllib/function:/home/<db2_instance_owner>/sqllib/java12/db2java.zip:/home/<db2_instance_owner>/sqllib/java/runtime.zip

Environment variable	Required value
PATH	.....:/home/<db2_instance_owner>/sqllib/java12:/home/<db2_instance_owner>/sqllib/bin:/home/<db2_instance_owner>/sqllib/adm:/home/<db2_instance_owner>/sqllib/misc

Update root environment file

The WebSphere Application Server will be run under root and will require access to the DB2 environment so that it can access the WAS administration database. This requires that the root account's environment .bashrc file be edited to add the following content at the end of the file:

```
# Setup DB2 environment for root user.
if [ -f /home/db2inst1/sqllib/db2profile ] ; then
. /home/db2inst1/sqllib/db2profile
fi

# Force DB2 to use JDBC 2.0.
if [ -f /home/<db2_instance_owner>/sqllib/java12/usejdbc2 ] ; then
. /home/<db2_instance_owner>/sqllib/java12/usejdbc2
fi
```

Configure the DB2 Client and server connectivity

To configure the DB2 Client, complete the following steps on the Web server (ganci6):

1. Catalog the Database Server TCP/IP node as follows:

   ```
   su - db2inst1
   db2 catalog tcpip node dbsrvlnx remote ganci5 server 50000
   ```

   Syntax:

   ```
   > db2 catalog tcpip node <node_name> remote <db_server_hostname> server
   <service/port#>
   ```

2. Catalog the WAS database as follows:

   ```
   db2 catalog db was at node ganci5
   ```

Syntax:

```
> db2 catalog db <database_name> at node <node_name>
```

Verify the DB2 Client and server connectivity

To verify the DB2 Client and server connectivity, complete the following steps on the Web server (ganci6):

1. Verify the attach to the TCP/IP node as follows:

   ```
   db2 attach to ganci5 user db2inst1 using <your_password>
   ```

   Syntax:

   ```
   > db2 attach to <node_name> user <db2inst_ID> using <db2inst_password>
   ```

2. Verify the connection to the database from the Web server (ganci6) as follows:

   ```
   db2 connect to was user db2inst1 using <your_password>
   ```

   Syntax:

   ```
   > db2 connect to <db_name> user <db2inst_ID> using <db2inst_password>
   ```

3.3.4 Install and configure the WebSphere Application Server

On the Web and application server, machine A, install the WebSphere Application Server. Refer to “Install the WebSphere Application Server” on page 28, being sure to include the IBM HTTP Server during the installation.

3.3.5 Configure and verify IBM HTTP Server

Finally, proceed to “Configure and verify the IBM HTTP Server” on page 37 and “Verify the IBM HTTP Server” on page 43 to complete the Web server configuration and test the environment.

4

WAS V4 for Linux 3-tier runtime environment

This chapter provides detailed instructions for implementing a WebSphere Application Server V4.0.1, Advanced Edition for Linux in a 3-tier runtime environment. Figure 4-1 on page 80 shows the layout of the environment for the 3-tier setup.

The chapter is organized into the following sections:

- Planning for a WAS V4 for Linux 3-tier runtime
- Install the DB2 Server
- Install the DB2 Client
- Install the WebSphere Application Server
- Install the IBM HTTP Server

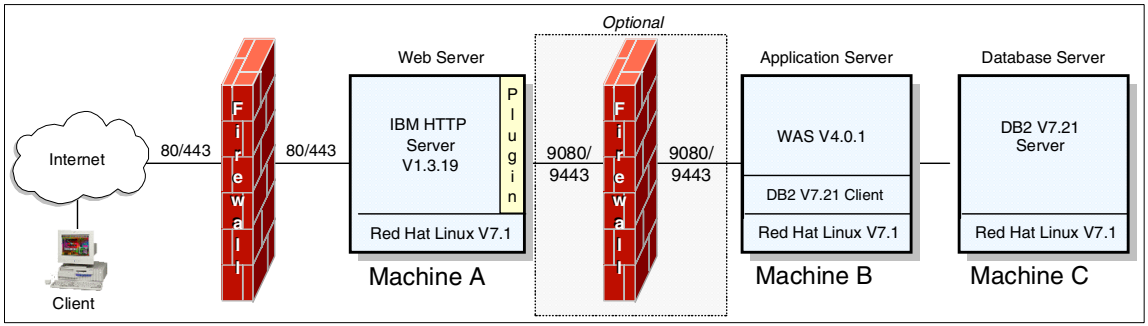


Figure 4-1 WAS V4 for Linux 3-tier runtime environment

4.1 Planning for a WAS V4 for Linux 3-tier runtime

This section defines the hardware and software used within the 3-tier WebSphere Application Server Linux environment.

This section includes the following topics:

- Overview
- Hardware and software prerequisites
- Hardware used within our test environment
- Software used within our test environment
- Install Red Hat Linux

4.1.1 Overview

The 3-tier environment is set up for performance, scalability, and security reasons. Having the three separate servers, as seen in Figure 4-1 on page 80, allows for the placement of firewalls between critical servers and also allows for the addition of servers to either of the three tiers with the least amount of hassle.

The following sections show the hardware and software we used in setting up and testing the 3-tier environment.

4.1.2 Hardware and software prerequisites

The prerequisites for the 3-tier environment are the same as for the single-tier environment in “Hardware and software prerequisites” on page 6. The only note would be that in the 3-tier environments the disk space requirements will be divided up accordingly between the two servers.

4.1.3 Hardware used within our test environment

We used the following systems within our two tier environments:

- Database server:
  - Hostname: ganci5
  - IP Address: 9.24.105.134
  - IBM NetVista (6579-A4U)
    - 866 MHz CPU
    - 512 MB RAM
    - 30 GB hard disk
    - 1 IBM Ethernet Adapter
    - 1 Integrated Intel 815e Solano Video Adapter
- Application server:
  - Hostname: ganci6
  - IP Address: 9.24.105.101
  - IBM Netfinity 3000 (8476-21U)
    - 300 MHz CPU (Pentium II)
    - 512 MB RAM
    - 4 GB hard disk
    - 1 IBM Ethernet Adapter
    - 1 Integrated S3 Trio-3D Video Adapter
- Web server:
  - Hostname: ganci4
  - IP Address: 9.24.105.133
  - IBM PC300PL (6565-3BU)
    - 667 MHz CPU (Pentium III)
    - 512 MB RAM
    - 20 GB hard disk
    - 1 IBM Ethernet Adapter
    - 1 S3 Inc. Savage4 Video Adapter

4.1.4 Software used within our test environment

The software used in the 3-tier environment is the same as that used in the single tier environment as shown in “Software used in our test environment” on page 8

4.1.5 Install Red Hat Linux

All three systems, Database server, Application server, and Web server, should be installed with Red Hat Linux V7.1 following the instructions in “Red Hat Linux installation” on page 11. In our setup we configured the same file systems on all servers for simplicity. In a real-world application the appropriate file systems would be created on each server. For instance, the database server would primarily need space in /usr/IBMdb2 and /home/<instance_user_names>.

4.2 Install the DB2 Server

Repeat the steps in “Install the DB2 Server” on page 13 to install the DB2 server software on the Database server (ganci5).

4.3 Install the DB2 Client

Repeat the steps in “Install the DB2 Client” on page 70 and “Configure the DB2 Client” on page 75 to install and configure the DB2 client software on the Application server (ganci6).

Important: In the referenced sections, ganci6 is referred to as the Web server. For the purposes of setting up the 3-tier environment, ganci6 has become the Application server. When completing the DB2 Client install and configuration steps, please make this substitution. In the 3-tier environment, the Web server, ganci4, will have no DB2 components at all.

4.4 Install the WebSphere Application Server

On the Application Server (ganci6) install the WebSphere Application Server as in “Install the WebSphere Application Server” on page 28 being sure *not* to include the IBM HTTP Server or plug-in during the installation.

4.5 Install the IBM HTTP Server

There are three ways that the IBM HTTP Server can be installed on the Web server:

- Manual, RPM, install
  - See “Install IBM HTTP Server V1.3.19” on page 66
- WebSphere Silent install
  - See “Automated install - silent” on page 34
- WebSphere GUI install
  - See “Interactive install - GUI” on page 30

Both the silent and GUI install options will require you to alter the selected products to just install the IBM HTTP Server and the plug-in.

If a number of Web servers were to be rolled out it might be quickest to use the WebSphere Silent install option to install both the IBM HTTP Server as well as the WebSphere plug-in. In our case we used the WebSphere GUI install to install both IBM HTTP Server and the plug-in.

After the installation of IBM HTTP Server refer to 2.5, “Configure and verify the IBM HTTP Server” on page 37 for details on configuring the Web server.

5

Deploy applications to a WAS V4 for Linux runtime

Throughout this chapter, we will utilize the PiggyBank sample application developed as part of the *WebSphere Version 4 Application Development Handbook*, SG24-6134 redbook. We will use this sample application to demonstrate the deployment and installation of an enterprise application in a WebSphere Application Server V4.0.1, Advanced Edition for Linux runtime environment.

This chapter is organized into the following sections:

- PiggyBank sample application overview
- Deployment considerations
- Set up Linux runtime for deployment
- Deploying VisualAge for Java assets
- Deploying WebSphere Studio assets
- Deploying the enterprise application
- Running the deployed enterprise application

5.1 PiggyBank sample application overview

The PiggyBank application is a simple enterprise application that manages account and customer record information for a fictitious bank. The Web-based application can be accessed either by a Web browser or by a custom developed client application. In this chapter, we will discuss accessing the application from a Web browser.

The PiggyBank sample Web application includes the following types of assets:

- Static Web content (HTML files, GIFs, JPEGs)
- Dynamic Web content (JSPs, servlets)
- Enterprise JavaBeans

Note: The development of Web assets using WebSphere Studio V4.0 and VisualAge for Java V4.0 are not discussed in this chapter. For more detailed information regarding the use of the tools and development of the PiggyBank sample application, refer to the *WebSphere Version 4 Application Development Handbook*, SG24-6134 redbook.

5.1.1 Download the sample application zip file

The PiggyBank application can be found on the following FTP server:

```
ftp://www.redbooks.ibm.com/redbooks/SG246134
```

The following files are included:

- Readme.txt (short instructions)
- SG246134code.zip (All sample code in ZIP format)

Download the ZIP file and save on your Windows development machine. For example, we unzipped the file to the d:\piggybank directory on a Windows development system.

5.1.2 Contents of the sample application zip file

The PiggyBank sample application includes the following directory structure and files:

SG246134\sampcode	Master directory
SG246134\sampcode\piggybank	PiggyBank application deployable modules
SG246134\sampcode\repository	VisualAge for Java repository
SG246134\sampcode\studio	WebSphere Studio archive file

The piggybank subdirectory contains the files listed in Table 5-1.

Table 5-1 Contents of the piggybank subdirectory

File name	Description
piggybank.ear	Enterprise archive ready for installation on WAS.
piggybank.jar	All the Java code
table.ddl	DDL for customer and account tables
earexpanded	Directory with EAR file contents
warexpanded	Directory with WAR file contents (from EAR file)

For development source code, the repository subdirectory contains the exported repository for VisualAge for Java piggybank.dat file.

In addition, the studio subdirectory contains a Studio archive file, piggybank-all.wsr for the HTML and JSP source code.

For the purpose of clarity, we have included a sample table.ddl (see Example 5-1), that contains user data.

Example 5-1 Sample table.ddl

```
CREATE TABLE "CUSTOMER"
 ("ID" INTEGER NOT NULL,
 "NAME" VARCHAR(32));

ALTER TABLE "CUSTOMER"
 ADD CONSTRAINT "CUSTOMERPK" PRIMARY KEY ("ID");

CREATE TABLE "ACCOUNT"
 ("NUMBER" INTEGER NOT NULL,
 "BALANCE" INTEGER,
 "CHECKING" SMALLINT,
 "CUSTOMERID" INTEGER);

ALTER TABLE "ACCOUNT"
 ADD CONSTRAINT "ACCOUNTPK" PRIMARY KEY ("NUMBER");

INSERT INTO CUSTOMER VALUES (123, 'Brian');
INSERT INTO ACCOUNT VALUES (1560, 1000, 0, 123);
INSERT INTO ACCOUNT VALUES (1561, 5000, 1, 123);
```

5.2 Deployment considerations

This section discusses considerations when setting up a WebSphere Application Server V4.0.1, Advanced Edition for Linux runtime environment for deployment of an enterprise application. Issues such as deployment stages, multi-tiered environments, Linux users, file systems, file transfers, and security are discussed.

There are a number of factors to consider when developing assets in a Windows environment and deploying them to a Linux runtime environment. The term *deployment* in this chapter refers to the movement of developed assets from the development environment to a usable form in the runtime environment.

The following issues should be taken into consideration:

- Deployment assets
- Deployment stages
- Multi-tiered environments
- Clustered environments (shared file systems such as DFS)
- Security (user access, file transfer mechanisms)
- File system (location of temporary files, user access)

5.2.1 Deployment assets

The WebSphere Application Server V4.0.1, Advanced Edition for Linux J2EE deployment modules include the following:

- **JAR files** - Java archive consisting of Java classes, Enterprise JavaBeans and associated deployment descriptors, and Java property files.
- **WAR files** - Web application archive consisting of JSPs, servlets, Java code, XML configuration files, and static content such as HTML files and images
- **EAR files** - Enterprise application archive consisting of JAR files (EJB, Java code), WAR files, and XML configuration files.
- **Static content** - HTML, images (GIFs, JPEGs), and Cascading Style Sheets (CSS) should be considered as separate deployable assets even though they can also be packaged in WAR files. Typically, production environments separate the static content (HTTP Server) from the dynamic content (Web application server) when deployed.

5.2.2 General deployment considerations

This section includes considerations for general deployment issues regarding enterprise Web assets.

Separation of static and dynamic content

WAR files provide the flexibility to package the static content (HTML, Images, CSS, etc.) in the same WAR file as dynamic content, such as JSPs and servlets. This requires that the WebSphere Application Server serve the static HTML and image files. The advantage of this is the capability to have a single WAR file contain the entire Web application. This is useful when testing and when quick packaging and deployment are needed. However, this method should not be used in production.

The preferred production method is to separate the static content that can be served by an HTTP server. This allows you to separate the HTTP server node from the Web application server node. Separation of static from dynamic content also provides performance benefits since the HTTP server is tuned for file serving functions. We will demonstrate how to do this with the PiggyBank application.

Creation of the WAR file

On the development machine or application server, we recommend that you create a WAR file. WebSphere Studio V4.0 provides the ability to create a WAR file on the development machine. The advantage of this method is that it allows you to deploy a single WAR file to the application server rather than an entire directory structure. The disadvantage is that your development system must contain all of the JSPs, servlets, and Java code necessary for the Web application.

The alternative method is to publish the entire directory structure of JSPs, servlets, and Java code to the application server, where it is assembled into a WAR file using the Application Assembly Tool (AAT). The advantage of this method is that creation of the WAR file is done on the target system ensuring that all content is available from their various sources. The disadvantage is that you have to manage temporary storage for all the dynamic content files used to create the WAR file. You also may have to manage multiple user access to the file system, since the source content may be coming from multiple sources.

The preferred method is to create the WAR file in the development environment. This helps to ensure that your resulting Web application module is independent of the Web application server being deployed to and keeps the management issues on the Web application server to a minimum.

Package common Java classes in a separate JAR file

Typically in a Web application, there are a number of utility Java classes that are used by various components of the enterprise application. These common classes should be packaged in a separate JAR file and assembled as part of the enterprise application (EAR file), not as part of the EJB deployment JAR or the Web application (WAR file).

5.2.3 Deployment stages

In a typical Web application life cycle, there are multiple stages that the application goes through. A typical scenario includes, but is not limited to:

- Development stage (source code development and unit testing)
- Testing stage (may be multiple testing stages)
- Staging stage (production-like environment for final verification before going into production)
- Production stage

Development stage

A typical scenario begins with the development stage where the Web application assets are developed and unit tested for single unit functionality. With the use of the integrated testing facilities of WebSphere Studio and VisualAge for Java, the unit testing is typically done in the development environment.

Deploying from development to testing

The next stage is typically some type of integrated testing stage where multiple functions are brought together for the first time for integration testing. There may be multiple testing stages: one for integration testing, one for multi-tiered testing, etc.

Integration testing is typically done on the target runtime platform so this is the first time where developed assets must be deployed to a separate runtime environment. Since the test environment is typically in a very controlled environment, the deployment considerations are not as rigorous as they are when deploying to a staging or production environment. This means that your choice of file transfer mechanism and user access security can typically be focused on what's easier for the development and test teams.

Deploying from testing to staging

After successfully completing the development and test cycle, the Web application is ready to be deployed to a staging environment. A staging environment is a near-production environment where Web applications can go through final verification before going “live”. This allows all involved parties to ensure that the staged Web application is indeed functioning as required.

It is at this point where security and file system issues become more important. The staging and production environment are typically in environments that are exposed to many other groups outside your immediate development team.

Deploying from staging to production

Typical staging and production environments are identical in every way except that the production environment is accessible by the intended users of the enterprise application, while the staging environment is only accessible by select individuals or teams.

Deployment from staging to production is a simple, but secure, file propagation from the staging server file system to the production server file system.

5.2.4 Deployment nodes and tiers

Deployment of an enterprise application typically involves multiple nodes and tiers. Typical separation of server nodes is as follows for a 3-tier runtime environment:

- HTTP Server node (for example, IBM HTTP Server)
- Web Application Server node (for example, IBM WebSphere Application Server)
- Database Server node (for example, IBM DB2 UDB)

VisualAge for Java assets are deployed to the Web Application Server node. WebSphere Studio assets, however, are deployed to both the HTTP Server node and the Web Application Server node. This should be taken into consideration when configuring your publishing stages in the WebSphere Studio environment. This will be demonstrated with the PiggyBank application later in this chapter.

5.2.5 Clustered environments

WebSphere production environments may include a cluster of application servers for load balancing and scalability. Typically, clustered environments use some type of shared file system such as DFS (Distributed File System) for synchronized file access by multiple servers. If this exists in your environment, then you will need to take the appropriate steps to conform to the file system structure and user access requirements of your shared file system.

Discussion of shared file systems and deployment is out of the scope of this book, however it is a deployment consideration you must be aware of.

5.2.6 Security considerations

Security and user access considerations vary from stage to stage. When deploying assets from development to a test environment, security issues are minimal since the test environment is typically in a very controlled environment and only accessible by the development and test teams. In this environment, FTP or a Samba server file share are suitable mechanisms for transferring files.

WebSphere Studio's publishing stages are designed to publish files using FTP or publish to a file system. In either case, the Linux server must have either an FTP server or Samba server installed and configured to allow files to be published from WebSphere Studio. We will demonstrate the FTP method in the PiggyBank sample application.

When deploying to a staging or production environment, extra security measures must be taken to ensure that the Linux server is secure from unauthorized access.

5.2.7 File transfer methods for deployment

Depending on the stage of deployment security requirements, you may use one of the following methods to transfer files (Web assets) to the runtime environment for deployment.

- File sharing (Samba, NFS)

  Direct file system sharing can be set up in non-production environments to facilitate simple deployment of assets. Linux systems have the ability to share file and print resources directly with PCs through the Samba package using the NetBIOS protocol. More information can be found at:

  http://www.samba.org/

- FTP

  FTP (File Transfer Protocol) can also be used as a transfer mechanism. Linux comes with an FTP server that should be enabled by default depending on the firewall settings chosen at install time.

- SCP

  SCP (Secure CoPy) uses an SSL (Secure Socket Layer) connection to safely transfer files from one server to another. Typically this is used in a production environment to enhance security. Production servers are usually hardened and will not have an FTP server enabled. Red Hat Linux V7.1 comes bundled with Open SSH, which includes an SCP client and server. Normally, these files would be transferred from a staging server to production servers; however, if needed there are a few Windows-based SCP clients. One example is WinSCP, found at:

  http://winscp.vse.cz/eng/

5.2.8 File system considerations

This section provides considerations for the target file system in various stages of deployment.

WebSphere installable and installed applications directory

The organization of deployable and installed applications is important. WebSphere Application Server V4.0.1, Advanced Edition for Linux creates two directories for this purpose:

Deployable applications: /opt/WebSphere/AppServer/installableApps

Installed applications: /opt/WebSphere/AppServer/installedApps

When an enterprise application is installed to the WebSphere Application Server, an associated application directory is created in the following directory containing enterprise application assets:

/opt/WebSphere/AppServer/installedApps

HTTP Server directories for static content

When separating the static content from the dynamic content, the static content will be deployed somewhere within the HTTP server document root. In the PiggyBank sample application, we will use the following directory for storing the Web application static content (default):

/opt/IBMHTTPServer/htdocs/en_US/piggybank

Temporary directories for deployed assets

Unless you are deploying a fully created EAR file, you will need a temporary location to store your JAR and WAR files on the Linux file system while you use the Application Assembly Tool to create the final EAR file.

Whatever directory you choose to use, you must ensure that you have the appropriate user access rights to create directories and files from the Windows development environment.

In the PiggyBank sample application, we will use a temporary working directory for storing our transient WAR and JAR files deployed from the Windows development environment as follows:

```
/usr/piggybank
```

Note: This temporary working directory should not be stored in the /tmp directory. The /tmp directory is meant to be used as a VERY temporary storage space. It is usually deleted often (every reboot, or sometimes every few hours) to keep the system clean.

You may want to create a file system specifically for holding the temporary files used in creating WebSphere applications. For example:

```
/opt/WebSphere/tempfiles
```

You could then create subdirectories for each application you are working with:

```
/opt/WebSphere/tempfiles/piggybank
```

These temporary files and directories can be cleaned out as necessary.

5.3 Set up Linux runtime for deployment

In our application deployment demonstration, we will deploy both VisualAge for Java and WebSphere Studio assets to a single-tier Linux runtime test environment utilizing FTP as the file transfer mechanism. We will also demonstrate how to separate the static content (HTML, GIF, CSS) from the dynamic content (JSP, servlets) and deploy from WebSphere Studio.

1. Install and configure an FTP server on the WebSphere Application Server V4.0.1, Advanced Edition for Linux server if one is not already installed.

   For example, do the following within the test runtime environment:

   a. Change to the xinetd.d directory

      ```
      cd /etc/xinetd.d
      ```

b. Rename the file wu-ftpd to ftpd.

```
mv wu-ftpd ftpd
```

c. Modify the ftpd file. By default Red Hat 7.1 installs with the default disable = yes. Set disable to no.

```
disable = no
```

d. Restart the service.

```
/etc/rc.d/init.d/xinetd restart
```

2. Create the piggybank directories.

   Create a temporary directory for deploying files from the development to the Linux runtime.

```
mkdir /usr/piggybank
mkdir /opt/IBMHTTPServer/htdocs/en_US/piggybank
```

3. Create the waslinux user account.

   a. Create a user named `waslinux` on the Linux system with a password of `waslinux`.

```
adduser waslinux
```

   b. Change the password of the `waslinux` user:

```
passwd waslinux
New password: waslinux
Retype password: waslinux
```

   c. Make sure that the waslinux user has access to write to both piggybank directories:

```
chown waslinux /usr/piggybank
chown waslinux /opt/IBMHTTPServer/htdocs/en_US/piggybank
```

4. Create the was4ad application database.

   a. First change users to the database instance owner:

```
su - db2inst1
```

   b. Create the was4ad database used by the PiggyBank sample application:

```
db2 create db was4ad
```

   c. Connect to the was4ad database:

```
db2 connect to was4ad
```

5. Create the was4ad database tables.

   a. At the DB2 command prompt, type the following:

```
db2
```

b. Type the following commands at the DB2 command prompt to create the tables and data:

```
CREATE TABLE “CUSTOMER” (“ID” INTEGER NOT NULL, “NAME” VARCHAR(32))
ALTER TABLE “CUSTOMER” ADD CONSTRAINT “CUSTOMERPK” PRIMARY KEY (“ID”)
CREATE TABLE “ACCOUNT” (“NUMBER” INTEGER NOT NULL, “BALANCE” INTEGER,
“CHECKING” SMALLINT, “CUSTOMERID” INTEGER)
ALTER TABLE “ACCOUNT” ADD CONSTRAINT “ACCOUNTPK” PRIMARY KEY (“NUMBER”)
INSERT INTO CUSTOMER VALUES (123, ‘Brian’)
INSERT INTO ACCOUNT VALUES (1560, 1000, 0, 123)
INSERT INTO ACCOUNT VALUES (1561, 5000, 1, 123)
```

6. To verify the values inserted to the ACCOUNT table, type the following commands:

```
db2
select * from ACCOUNT
```

You should see the values inserted in the previous step.

7. Disconnect from the database.

```
quit
db2 disconnect current
```

5.4 Deploying VisualAge for Java assets

This section describes how to deploy assets developed in VisualAge for Java from a Windows development environment to a WebSphere Application Server V4.0.1, Advanced Edition for Linux runtime environment.

Note: For detailed information about development using VisualAge for Java, refer to the redbook *WebSphere Version 4 Application Development Handbook*, SG24-6134 (Chapter 11).

5.4.1 VisualAge for Java assets

Web applications usually consist of HTML, JSPs, servlets, and EJBs. VisualAge for Java does not provide editing facilities for HTML or JSP code. WebSphere Studio provides the Page Designer for editing Web pages. VisualAge for Java is used to develop:

- Servlets
- EJBs
- Java classes required by the Web application

The deployable form of all of these asset types is a JAR file that can be generated within VisualAge for Java.

5.4.2 VisualAge for Java deployment considerations

VisualAge for Java code can be exported in a number of ways for deployment to WebSphere Application Server:

- **Export to a JAR file** - Java source code and compiled class files can be exported into a JAR file.
- **Export to a directory** - Java source code and compiled class files can be exported to a directory structure.
- **Export a deployable EJB JAR file** - VisualAge for Java can create an EJB JAR file that is ready to be deployed in WebSphere. In the PiggyBank sample application, we will generate the deployable EJB JAR file using the Application Assembly Tool (AAT) on the WebSphere Application Server.

Note: All JAR files should have all lowercase names. For example use the name:

piggybank-ejb.jar

instead of:

PiggybankEJB.jar

This not only follows standard Java naming convention for JAR files, but also ensures that your JAR files are found in the classpath. You may have problems with the WebSphere runtime finding JAR files if the filename is not all lowercase.

5.4.3 Deploying PiggyBank VAJ assets

VisualAge for Java (VAJ) creates the servlet, EJB, and helper classes needed by a Web application. The development steps for the PiggyBank application are covered in the *WebSphere Version 4 Application Development Handbook*, SG24-6134 (Chapter 4). The VAJ-created classes are packaged in one or many JAR files. The JAR files we will use for the PiggyBank application are located in the following directory:

d:\piggybank\sg246134\sampcode\piggybank\EARexpanded

The directory includes the following:

- piggybank-ejb.jar - EJB classes
- piggybank-usecases.jar - command pattern classes
- piggybank-common.jar - helper classes required by the EJBs

Note: These deployable assets can be created using any development tool, as long as the resulting classes follow the J2EE 1.2 specification. This includes Servlet 2.2 and EJB 1.1.

Deploying VisualAge for Java assets from Windows to Linux

We will not deploy the VisualAge for Java assets directly to the Linux runtime environment. Instead, we will create the JAR files (EJBs, servlets, helper classes, etc.) on the local file system and then FTP the files to the Linux environment. This section does not discuss the use of VisualAge for Java to create the JAR files. For detailed information regarding the use of VisualAge for Java, please refer to *WebSphere Version 4 Application Development Handbook*, SG24-6134.

The first step is to move the VAJ-created JAR files to the Linux environment via FTP.

1. On your Windows development machine, change to the directory with the JAR files as follows:

   ```
   cd d:\piggybank\sg246134\sampcode\piggybank\EARexpanded
   ```

2. Make an FTP connection to the Linux WAS machine:

   ```
   ftp ganci5
   ```

   user: `waslinux`

   pw: `waslinux`

3. Ensure that the files are transferred in binary mode:

   ```
   binary
   ```

4. Change to piggybank directory:

   ```
   cd /usr/piggybank
   ```

5. Put the files into piggybank directory:

   ```
   mput p*.jar (answer y to each request)
   ```

6. Close the FTP session:

   ```
   quit
   ```

Creating the EJB module

The next step is to create the EJB module using the Application Assembly Tool (AAT) on the WebSphere Application Server V4.0.1, Advanced Edition for Linux.

1. Start the AAT on the Linux WebSphere Application Server as follows:

   ```
   cd /opt/WebSphere/AppServer/bin
   ./assembly.sh
   ```

2. Open the **New** tab of the AAT, select **EJB Module**, and then click **OK**.
3. Select the **General** tab, and enter the following (see Figure 5-1):
   a. Display Name: `Piggybank EJBs`
   b. Classpath: `piggybank-common.jar`

      The classes required by the EJBs are in this JAR file.

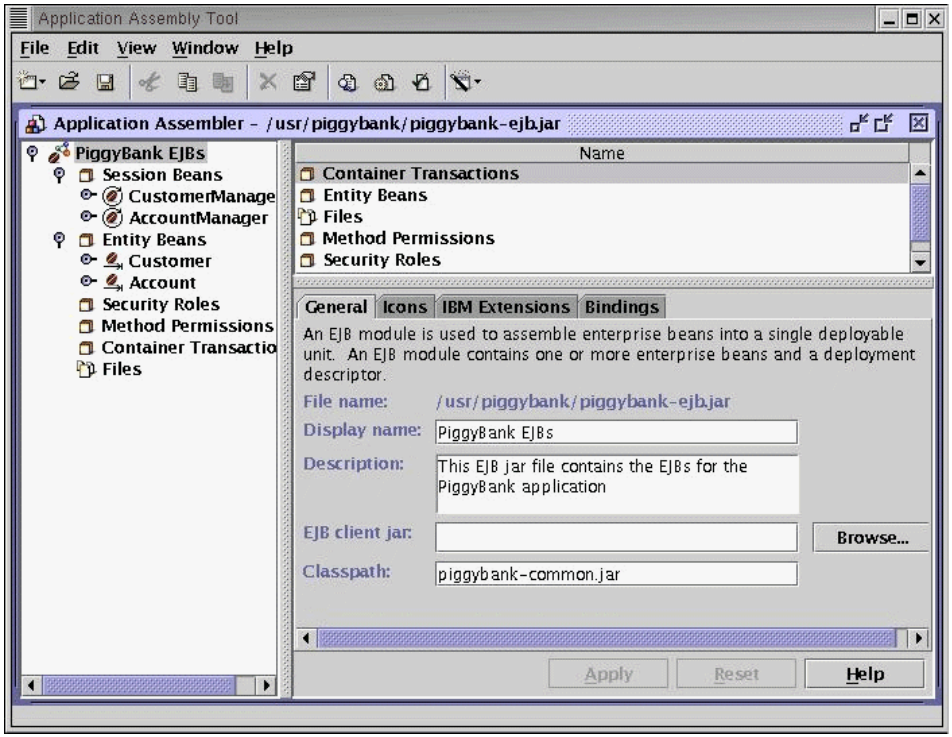


Figure 5-1 EJB module - piggybank-ejb.jar

4. Open the **Bindings** tab, enter the following, and then click **Apply**:

   JNDI name: `jdbc/WAS4AD`

   User ID: `db2inst1`

   Password: `db2inst1`
5. Import Entity Beans as follows:
   a. Right-click **Entity Beans** in the left-hand pane, and select **Import**.
   b. Specify the EJB JAR file as the archive:

      `/usr/piggybank/piggybank-ejb.jar`

c. Select both the **Customer** and **Account** beans, click **Add**, and then click **OK**.

6. Import Session Beans as follows:
   a. Right-click **Session Beans** in the left-hand pane, and then select **Import**.
   b. Specify the EJB JAR file as the archive as follows and press Enter:

      `/usr/piggybank/piggybank-ejb.jar`

   c. Select both the **CustomerManager** and **AccountManager** beans, click **Add**, and then click **OK**.
7. Update the Container Transaction properties.

   The EJB container manages the transaction scope for EJB method invocation.
   a. Right-click **Container Transactions** in the left-hand pane, and click **New**.
   b. In the New Container Transaction window, specify the following and then click **Add**:

      Name: `Account`

      Transaction attribute: `Required`

   c. When the Add Methods window appears, expand the **+** and select the **Account (*)** folder, and then click **OK**.
   d. Click **OK** in the New Container Transaction window.
8. Repeat the previous procedure to update container transaction properties for the following:
   – AccountManager
   – Customer
   – Customer Manager
9. Add Security Roles.
   a. Right-click **Security Roles** in the left-hand pane, and then click **New**.
   b. When the New Security Role window appears, specify the following and click **OK**:

      Name: `DenyAllRole`

      Description: `Deny all access role`

10. Save the current EJB JAR file by clicking **File -> Save As**. In the Save window, enter the following and then click **Save**:

    File name: `/usr/piggybank/piggy-ejb.jar`

Note: The file name is piggy-ejb.jar, not piggybank-ejb.jar.

11. You should get a message stating the EJB Module /usr/piggybank/piggy-ejb.jar is now complete!

We will generate the deployed EJB code when we create the Piggybank EAR file. The next step is to create the Web Application Module (WAR file).

5.5 Deploying WebSphere Studio assets

This section describes how to deploy assets developed in WebSphere Studio on a Windows development platform to a WebSphere Application Server V4.0.1, Advanced Edition for Linux runtime environment.

Note: For detailed information about development using WebSphere Studio refer to *WebSphere Version 4 Application Development Handbook*, SG24-6134 (Chapter 10).

5.5.1 WebSphere Studio assets

Enterprise Web applications usually consist of HTML, JSPs, servlets, and EJBs. WebSphere Studio provides for the following:

- Developing the Web presentation content: HTML, JSP, CSS
- Assembling the Web application: organizing the flow, creating and publishing WAR files and other assets to deployment servers.

5.5.2 WebSphere Studio deployment considerations

WebSphere Studio allows you significant flexibility in how you deploy WebSphere Studio assets.

Features include the following:

- **Publishing Stages** - allows you to define stages that define the content that gets published and to what server. For example, you may have a test stage and a production stage that deploy different content to different servers.
- **Multiple server definitions in a publishing stage** - allows you to define multiple servers in a single publishing stage. This allows you to specify that certain content should get published to different servers.

- **File transfer mechanisms** - You can configure the publishing information to publish either using FTP or to publish to a file system.
- **WAR file creation** - Version 4.0 now allows you to create a Web application archive file on the development system instead of on the WebSphere Application Server.

In the PiggyBank sample application, we will configure WebSphere Studio to deploy the application to a multi-tiered environment. This will consist of two server definitions, one for the HTTP Server and one for the WebSphere Application Server. We will assemble the dynamic JSP and servlet content into a WAR file in WebSphere Studio before deploying to the Linux WebSphere Application Server.

5.5.3 Deploying PiggyBank WebSphere Studio assets

In this section, we will create a WebSphere Studio project called PiggyBank, define the publishing stages and servers, create a WAR file, and deploy the assets to a Linux runtime environment.

1. First, start WebSphere Studio and create a new project called PiggyBank.
   a. Start WebSphere Studio.
   b. Select **File -> New Project**.
   c. When the New Project window appears, specify the following and the click **OK**:

      Project Name: `PiggyBank`
2. Delete the theme/Master.css file. If prompted by a delete window, select **Delete from disk**.
3. Insert and modify WebSphere Studio PiggyBank assets (HTML, JSPs, and image files) to the project as follows:
   a. Right-click the **PiggyBank** project folder, and select **Insert -> Folder**.
   b. Select the **Use Existing** tab.
   c. Select the following folder and click **OK**:

      `D:\piggybank\SG246134\sampcode\piggybank\WARexpanded\images`
   d. Click **Add**.
   e. Click **OK**.
   f. Right-click the **PiggyBank** folder, and select **Insert -> File**.
   g. Select the **Use Existing** tab.
   h. Select all JSPs and HTML files in the following folder, and click **Open**:

```
D:\piggybank\SG246134\sampcode\piggybank\WARexpanded
```

Tip: Hold down the shift key while selecting to select more than one file.

i. Click **OK**.

4. Insert the Master.css file as follows:
   a. Right-click the **theme** folder, and select **Insert** -> **File**.
   b. In the Insert File window, select the **Use Existing** tab.
   c. Select the following file and click **Open**:

      ```
      d:\piggybank\SG246134\sampcode\piggybank\WARexpanded\theme\Master.css
      file
      ```

5. Insert the Java classes and servlets as follows:
   a. Right-click the **servlet** folder, and then select **Insert -> Folder**.
   b. In the Insert folder window, select the **Use Existing** tab.
   c. Select the following directory:

      ```
      D:\piggybank\SG246134\sampcode\piggybank\WARexpanded\WEB-INF\classes\its
      o
      ```

   d. Click **Add**.
   e. Click **OK**.

Setup WebSphere publishing stages

The first step in preparing to deploy WebSphere Studio assets to a Linux environment is to configure the servers and publishing stages in WebSphere Studio.

Since our Web application includes both static (HTML and GIF) and dynamic (JSP, servlets) assets, we will create two publishing servers. We will create one publishing server for the HTTP Server and one for the WebSphere Application Server. This is done to take into account environments which have separate HTTP and application servers (multi-tiered runtimes).

1. Create a deployment publishing stage as follows:
   a. Select **Project -> Customize Publishing Stages**.
   b. Set the stage name to **Deployment**.
   c. Click **Add**.
   d. Click **OK**.
2. Adding publishing server definitions to the deployment publishing stage.

a. Change to the Deployment stage and select **Project -> Publishing Stage -> Deployment**.

b. Open the Publish view and select **View -> Publishing.**

3. Add an HTTP Server definition to the deployment publishing stage.

   a. Right-click **Deployment** from the Publishing View.

   b. Select **Insert** -> **Server**...

   c. Set the following values:

      Server name: `LinuxHTTPServer`

      Server address: `http://<hostname or IP of your Linux HTTP server>`

   d. Click **OK**.

4. Add a WebSphere Application Server definition to the Deployment publishing stage.

   a. Right-click **Deployment** from the Publishing View.

   b. Select **Insert -> Server**...

   c. Set the following values:

      Server name: `LinuxWebSphereServer`

      Server address: `http://<hostname or IP of your Linux WAS server>`

   d. Click **OK**.

5. Modify the LinuxHTTPServer server properties.

   a. Right-click **LinuxHTTPServer** in the Publishing view.

   b. Select **Properties**.

   c. Select the **Publishing** tab.

   d. Select **Publish using FTP**.

   e. Enter the following values:

      - Login: `waslinux`
      - Password: `waslinux`
      - Check **Save password**.

   f. Modify the targets of this server by clicking **Targets**.

   g. When the Publishing Targets window appears, specify the following and then click **OK**:

      - html: `/opt/IBMHTTPServer/htdocs/en_US/piggybank`
      - servlet: `/usr/piggybank`
      - Remove the resources target folder

- Remove the rules target folder

6. Modify the LinuxWebSphereServer server properties:
   a. Right-click **LinuxWebSphereServer** in the Publishing view.
   b. Select **Properties**.
   c. Select the **Publishing** tab.
   d. Select **Publish using FTP**.
   e. Set the following values:
      - Login: `waslinux`
      - Password: `waslinux`
      - Check **Save password**.
   f. Modify the targets of this server (see Figure 5-2 on page 106) by clicking **Targets**.
   g. When the Publishing Targets window appears, specify the following and then click **OK**:
      - Change the HTML path to:

        `/opt/IBMHTTPServer/htdocs/en_US/piggybank`
      - Change the servlet path to: `/usr/piggybank`
      - Remove the resources target folder
      - Remove the rules target folder
      - Add a new target called `war` and set the path to `/usr/piggybank`.
   h. Click **OK**.

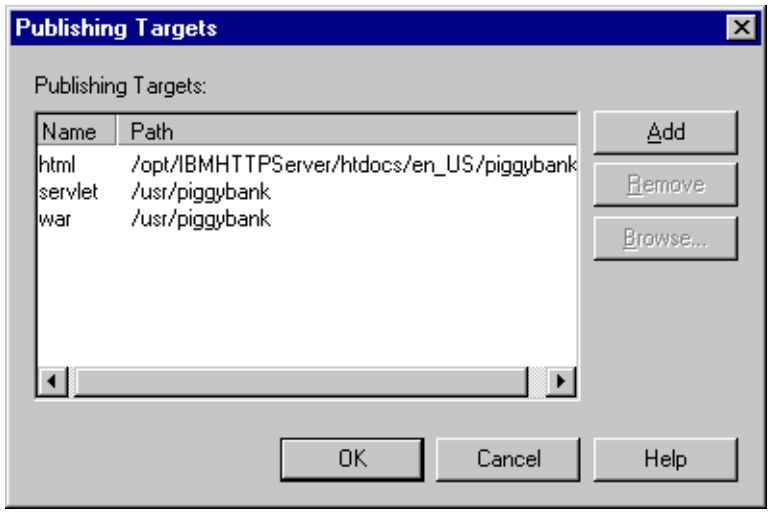


Figure 5-2 WAS V4 for Linux: WebSphere Studio publishing targets

7. Modify the items to publish to the Linux HTTP Server and Linux WebSphere Application Server.

   Move all non-static content from the LinuxHTTPServer tree to the LinuxWebSphereServer tree as shown in Figure 5-3 on page 107.

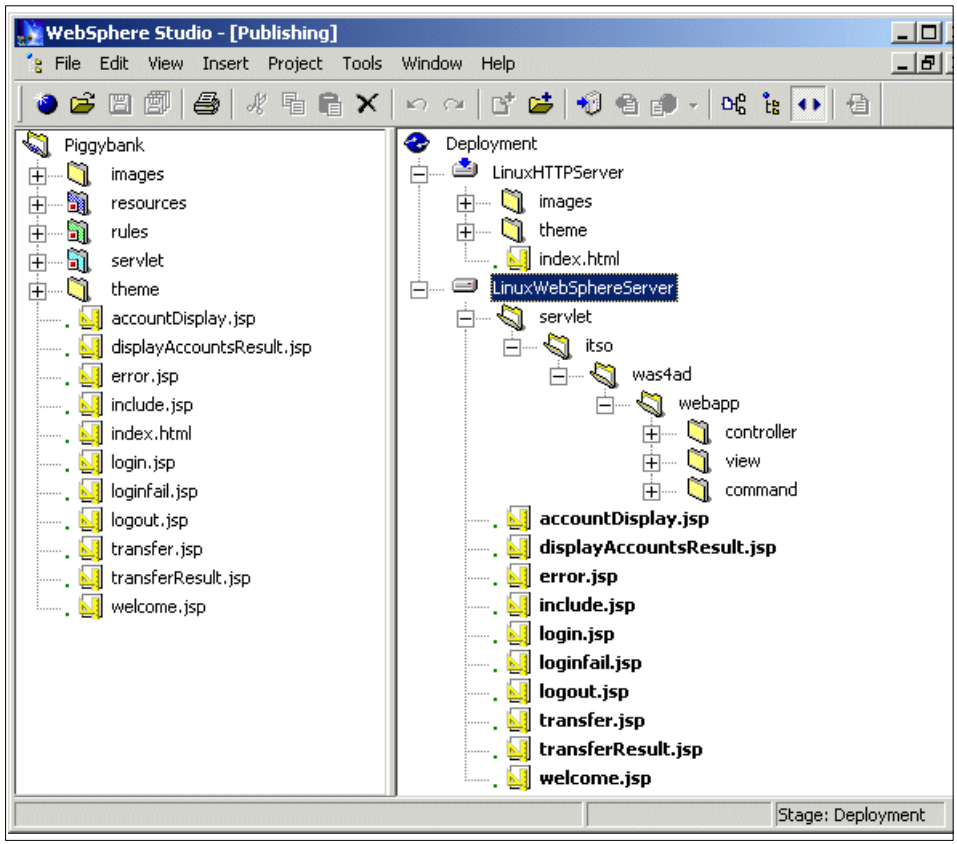


Figure 5-3 Piggybank project publishing trees

Publishing from WebSphere Studio

Once the publishing stages are set up and publishing servers are defined, you are now ready to publish WebSphere Studio content to the targets.

1. Publish the static assets to the HTTP Server:
   a. Right-click the **LinuxHTTPServer**.
   b. Select **Publish this Server**.
   c. Click **OK**.

      The static content will be FTPed to the Linux HTTP Server under the following directory:

      `/opt/IBMHTTPServer/htdocs/en_US/piggybank`
2. Set the LinuxWebSphereServer as the default publishing server.
   a. Right-click the **Deployment** stage, and then select **Properties.**

b. In the Advanced tab, set the default server to LinuxWebSphereServer.

c. Click **OK**.

Create the dynamic Web application WAR file

Before publishing the dynamic content to the WebSphere Application Server, we will generate a WAR file. This is an optional step. The alternative is to publish all the dynamic content (JSPs, servlets, etc) to a file system on the Linux WebSphere Application Server and assemble the WAR file on the server using the Application Assembly Tool. However, to minimize the amount of work performed on the server, we will assemble and publish the WAR file within WebSphere Studio.

1. Generate the servlet mappings for the Controller Servlet.
   a. Right-click the **Controller Servlet** class file and select **Properties**.
   b. Check **Use default publishing path**.
   c. Set Servlet Mapping to `*.pbc`.
   d. Click **OK.**
2. Generate the deployment descriptor file:
   a. Select **Project -> Create Web Configurator Descriptor File.**
   b. In the window, set the following values:
      - Server: `LinuxWebSphereServer`
      - Check the **ControllerServlet** box
      - Click **Create**

   This creates the LinuxWebSphereServer_web.xml descriptor in the WEB-INF directory.
3. Once the deployment descriptor has been generated, we can generate the WAR file.
   a. Select **Project -> Create Web Archive file**.
   b. Specify the server as LinuxWebSphereServer.
   c. Click **OK**.
   d. Specify a location and filename for the WAR file:

      ```
      D:\piggybank\piggybank.war
      ```

   The WAR file has been created and is ready for publishing to the Linux WebSphere Server.
4. Publish the Web Archive (WAR) file:
   a. Select **Project -> Publish Web Archive**

b. In the window, set the following values:
   - Specify the D:\piggybank\piggybank.war file
   - Specify the server as the LinuxWebSphereServer
   - Specify the target as `war`

c. Click **OK.**

The piggybank.war file will be FTPed to the /usr/piggybank directory.

All VAJ and Studio assets have now been deployed to the Linux machine. We are now ready to create and install the enterprise application on the WebSphere Application Server V4.0.1, Advanced Edition for Linux.

5.6 Deploying the enterprise application

After the WAR, JAR, and static content are deployed to the Linux runtime environment, you are ready to deploy the complete enterprise application. The deployment form of a J2EE enterprise application is an EAR file. Here are the steps we will take on the WebSphere Application Server V4.0.1, Advanced Edition for Linux to deploy the enterprise application:

- Configure the WAR file
- Create the EJB references
- Create and assemble the enterprise application
- Install the enterprise application using the Admin Console

5.6.1 Configure the WAR file

The first step in preparing the enterprise application is to configure the WAR file(s) that has been published by WebSphere Studio.

1. Start the Application Assembly Tool (AAT) as follows:

   ```
   /opt/WebSphere/AppServer/bin/assembly.sh
   ```

2. Configure the piggybank.war file. The first step is to configure the piggybank.war file (see Figure 5-4).

   a. Select **File -> Open**.

   b. Open the /usr/piggybank/piggybank.war file.

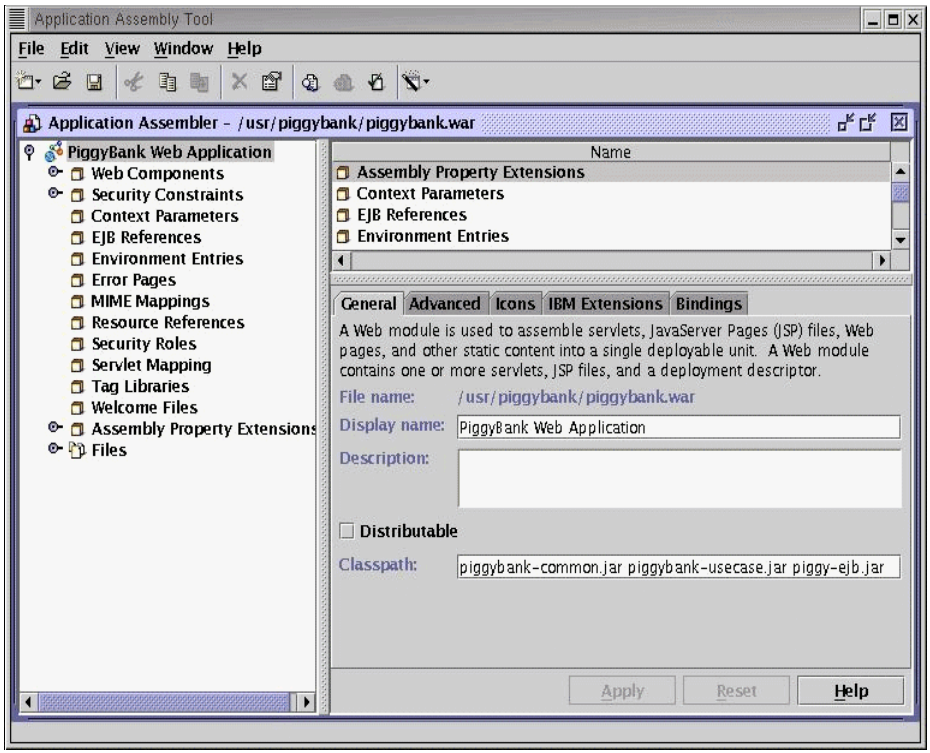


Figure 5-4 WAR Module - piggybank.war

3. Modify the PiggyBank properties.
   a. Select the **General** tab, and update the classpath as follows (see Figure 5-4):

      `piggybank-common.jar piggybank-usecase.jar piggy-ejb.jar`

   b. Select the **IBM Extensions** tab, and update the additional classpath as follows: `common.jar`.
4. Modify the Controller Servlet properties.
   a. Select **Web Components**.
   b. Select the **General** tab, and specify the following:

      Component Name: `ControllerServlet`

      Display Name: `PiggyBank Controller`

c. Configure the initialization parameters by right-clicking **Initialization Parameters**, and then select **New**. The initialization parameters we entered are listed in Table 5-2.

Table 5-2 Controller Servlet initialization parameters

Name	Value
COMMAND_PACKAGE	itso.was4ad.webapp.command
ERROR_PACKAGE	/error.jsp
LOGIN_ATTR	customer
LOGIN_COMMAND	Login
LOGIN_PAGE	/login.jsp

5.6.2 Create the EJB references

Create the EJB references to locate the home interface of the two EJBs used by the Web application.

1. Create an EJB Reference for the AccountManager by right-clicking **EJB References**, and then select **New**.
   a. In the General tab, specify the following:

      Name: `ejb/AccountManager`

      Home: `itso.was4ad.ejb.account.AccountManagerHome`

      Remote: `itso.was4ad.ejb.account.AccountManager`

      Type: `Session`

   b. In the Bindings tab, specify the following:

      JNDI Name: `itso/was4ad/ejb/account/AccountManager`
2. Create an EJB Reference for the CustomerManager by right-clicking on **EJB References**, and selecting **New**.
   a. In the General tab, specify the following:

      Name: `ejb/CustomerManager`

      Home: `itso.was4ad.ejb.customer.CustomerManagerHome`

      Remote: `itso.was4ad.ejb.customer.CustomerManager`

      Type: `Session`

   b. In the Bindings tab, specify the following:

      JNDI Name: `itso/was4ad/ejb/customer/CustomerManager`
3. Add Security Roles.

a. Right-click **Security Roles** and click **New**.

b. Specify the following:

Name: `DenyAllRole`

Description: `Deny all access role`

4. Update Servlet Mappings.

a. Right-click **Servlet Mapping** and select **New**.

b. Specify the following:

URL pattern: `*.pbc`

Servlet: `ControllerServlet`

5. Save the piggybank.war file by selecting **File -> Save**.

5.6.3 Create and assemble the enterprise application

The enterprise application (EAR) is created using the Application Assembly Tool. An EAR file is comprised of:

- WAR files
- EJB Module JAR files
- XML Deployment Descriptors

We have already created and configured the WAR file and EJB JAR file needed by the PiggyBank application. We will now assemble those components into an EAR file that is deployed to the WebSphere Application Server.

To create and assemble the application using the AAT, complete the following steps:

1. Start the Application Assembly Tool (AAT) as follows:

```
/opt/WebSphere/AppServer/bin/assembly.sh
```

2. Create a new enterprise application, by selecting **File -> New -> Application**.
3. Add the EJB Module.

a. Right-click **EJB Modules**, and the select **Import**.

b. Select **/usr/piggybank/piggy-ejb.jar**.

4. Add the Web app module.

a. Right-click **Web Modules**, and select **Import**.

b. Select **/usr/piggybank/piggybank.war**.

c. Specify the following context root: `/`

5. Add the common JAR files to the enterprise application.
   a. Right-click **Files** and select **Add Files**.
   b. Select the following files:

      ```
      piggybank-common.jar
      piggybank-usecase.jar
      ```

   c. Click **Add**.
   d. Click **OK**.
6. Specify EAR properties in the General tab (see Figure 5-5):

   Display Name: `PiggyBank Application`

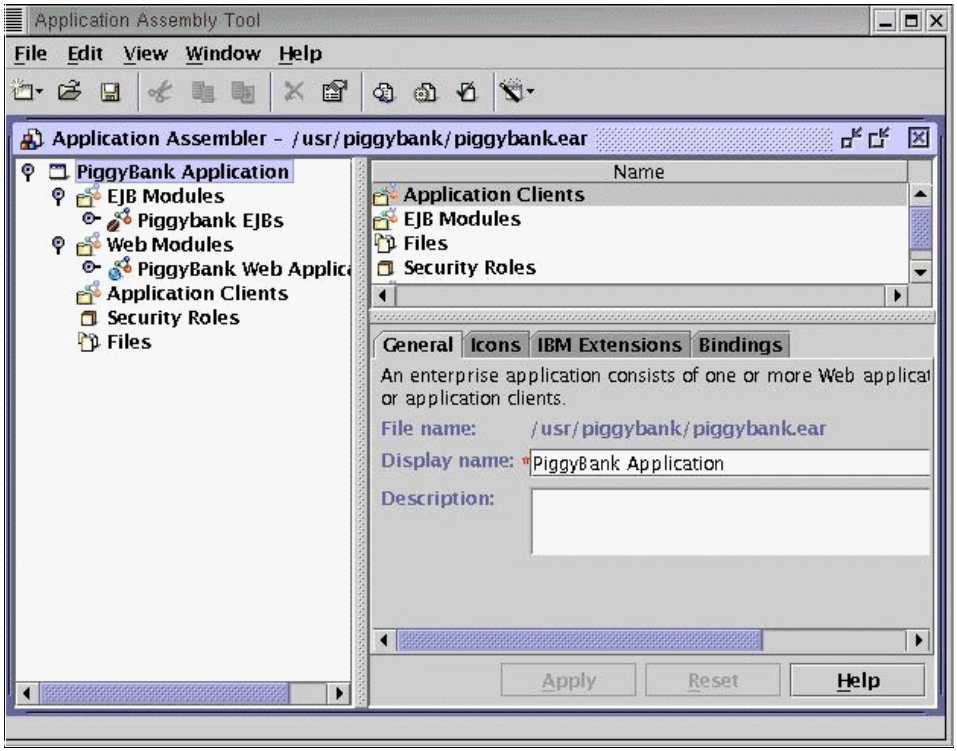


Figure 5-5 EAR Module - piggybank.ear

7. Save the EAR file by clicking **File -> Save As.** Specify the file as:

   `/usr/piggybank/piggybank.ear`
8. Generate the deployable EAR file.
   a. Right-click the **PiggyBank Application**.

b. Select **Generate Code for Deployment**.

c. Specify the Deployed module location:

`/opt/WebSphere/AppServer/installableApps/Deployed_piggybank.ear`

d. Specify the Dependent classpath:

`/usr/piggybank/piggybank-common.jar`

e. Specify the Database name:

`was4ad`

f. Check the **Verify Archive** check box, and then select **Generate Now**.

The Deployed_piggybank.ear file will be generated.

You are now ready to install the PiggyBank enterprise application to the WebSphere Application Server.

5.6.4 Install the enterprise application using the Admin Console

The WebSphere Application Server provides two directories for managing enterprise application (EAR) files:

- Installable Applications: `/opt/WebSphere/AppServer/installableApps`
- Installed Applications: `/opt/WebSphere/AppServer/installedApps`

To install a deployable EAR file, complete the following steps:

1. Start the WebSphere Administrative Console as follows:

   `/opt/WebSphere/AppServer/bin/adminclient.sh`

2. Create the JDBC Driver and DataSource for the PiggyBank CMP EJBs.

   a. Select **Console -> New -> JDBC Provider**.

   b. Specify the following:

   Name: `jdbc`

   Implementation class: `COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource`

   c. Select **Console -> New -> Data Source**.

   d. Specify the following:

   Name: `WAS4AD`

   User ID: `db2inst1`

   Password: `db2inst1`

   Confirm password: `db2inst1`

3. Install the new JDBC Driver on the server node.

Note: On Linux (Intel), the JDBC provider implementations are located in the following locations:

```
DB2: $DB2_HOME/sqllib/java12/db2java.zip
Oracle: $ORACLE_HOME/jdbc/lib/classes12.zip
```

a. Under Resources, select the **jdbc** provider.

b. Select the **Nodes** tab.

c. Click **Install New**.

d. Select the server node to install the driver on.

e. Specify the following driver:

```
/home/db2inst1/sqllib/java12/db2java.zip
```

f. Select **Install**.

4. Install the PiggyBank enterprise application.

Note: The Piggybank application uses a root context of `'/'`. Before installing the Piggybank application, remove or disable any existing Web application that uses the root context `'/'` and shares the same virtual host as the PiggyBank Web application. If you intend to use the default_host virtual host, you have to remove or disable the WebSphere samples that also use the root context `/` in order to install the PiggyBank Web application.

a. Select **Console -> Wizards -> Install Enterprise Application**.

b. Specify the following:

Path:
`/opt/WebSphere/AppServer/installableApps/Deployed_piggybank.ear`

Application Name: `PiggyBank`

c. Click **Next.**

d. Continue selecting **Next** until you get to the last window, and then click **Finish**.

e. When prompted, click **No** to regenerate code since it has already been generated.

5. Regenerate the Webserver Plug-in.

a. Right-click **<your_server_node>**.

b. Select **Regen Webserver Plugin**.

The PiggyBank enterprise application is now installed.

5.7 Running the deployed enterprise application

Update the hpptd.conf with the Piggybank directory as the new document root. This is necessary because the HTML and JSP pages expect the Piggybank images and home page to be in the document root.

1. Edit the /opt/IBMHTTPServer/conf/httpd.conf file to specify the following:

   ```
   DocumentRoot "/opt/IBMHTTPServer/htdocs/en_US/piggybank
   ```

2. Restart the IBM HTTP Server as follows:

   ```
   /opt/IBMHTTPServer/bin/apachectl restart
   ```

3. Start the PiggyBank application from the WebSphere Admin Console by right-clicking the **PiggyBank** application and selecting **Start**.

4. Launch a Web browser, and enter the following URL to access the application:

   ```
   http://<your_http_server>/index.html
   ```

5. In the Login page, specify the following:

   Customer ID: `123`

   Password: `<any password>`

6

The future of WebSphere development tools for Linux

A typical Web development environment includes a wide range of tools and development roles. Developers and specialists have a great need for an open development environment that allows components to be integrated into a common work environment or workbench based on the roles of the user.

To address this need, IBM has adopted a new open WebSphere development tools strategy. The new development tools provide an open and more easily integrated solution for external development tools (design, source control, etc.) to be added into the development workbench. In contrast with the current development tools (VAJ and Studio), which only provides support for Windows, the new WebSphere tools strategy includes support for Windows 98, ME, NT, 2000, Red Hat Linux 7.1.

The new IBM WebSphere development tools strategy includes the following:

- Eclipse Project

  The Eclipse Project is an initiative for the independent development of tools that provide the user with a seamless integrated experience when they are brought together in the Eclipse Workbench.

  The primary initiatives are the common framework called the Eclipse Platform, the Java development tools, and the Plug-in Development Environment (PDE).

At the core of the Eclipse Project is the Eclipse Platform, which is a common development framework or Integrated Development Environment (IDE). The Eclipse Platform provides the core frameworks and services upon which all plug-in extensions are created. It also provides the runtime in which plug-ins are loaded, integrated, and executed. This open architecture enables other tool developers to build and deliver integrated tools for the Eclipse Platform. In addition, the Eclipse Platform provides non-developers an IDE for such tasks as deployment.

The Java development tools provide tool plug-ins that implement a Java IDE that supports the development of Java applications including Eclipse plug-ins. The Java development tools allows the Eclipse Platform to be a development environment for itself.

The Plug-in Development Environment (PDE) extends the Eclipse Platform and the JDT to provide views and editors that make it easier to build plug-ins for Eclipse. The PDE helps you figure out what extension are available and how to plug into them, and helps you put together your code in a plug-in format. The PDE makes integrating plug-ins easy.

Additional information on Eclipse Projects can be found at:

http://www.eclipse.org/

- IBM WebSphere Studio Workbench

  The IBM WebSphere Studio Workbench is based upon a release level of the Eclipse Project that is supported by IBM and is the base for IBM application development products (WebSphere Studio family). New version of the WebSphere Studio Workbench will be created from the Eclipse Project at appropriate intervals.

  The WebSphere Studio Workbench provides a common platform for tools development and additional functionality. The workbench is the base for accessing common services for the WebSphere Studio development products. The IBM WebSphere Studio Workbench is intended for ISVs that create development tools and plug-ins.

- IBM WebSphere Studio Site Developer

  This product is based on the IBM WebSphere Studio Workbench and adds the functionality for Web or site developers to create such assets as static HTML, XML, servlets, and JSPs.

- IBM WebSphere Studio Application Developer

  This product is based on the IBM WebSphere Studio Workbench, includes the features of the WebSphere Studio Site Developer, and additional functionality for J2EE development (EJBs, database tools, etc.).

6.1 WebSphere Studio Workbench

The IBM WebSphere Studio Workbench, announced in May 2001, is the foundation for the next generation of application development tools. It provides a common platform for both tool development and tool functionality by providing the base for accessing common services and frameworks such as:

- Editor Frameworks
- User Interface Frameworks
- Team Programming Models
- Resource Management
- Debugging
- Java Development

The platform provides an open and flexible model for adding new tool functionality as plug-in components, as seen in Figure 6-1 on page 120. WebSphere Studio Site Developer and WebSphere Studio Application Developer are the first two IBM development products to be based on the WebSphere Studio Workbench.

For detailed information about the IBM WebSphere Studio Workbench, refer to the following PDF file:

`http://www.ibm.com/software/ad/workbench/pdf/workbench5_25.pdf`

or the IBM Web site:

`http://www.ibm.com/software/ad/workbench/`

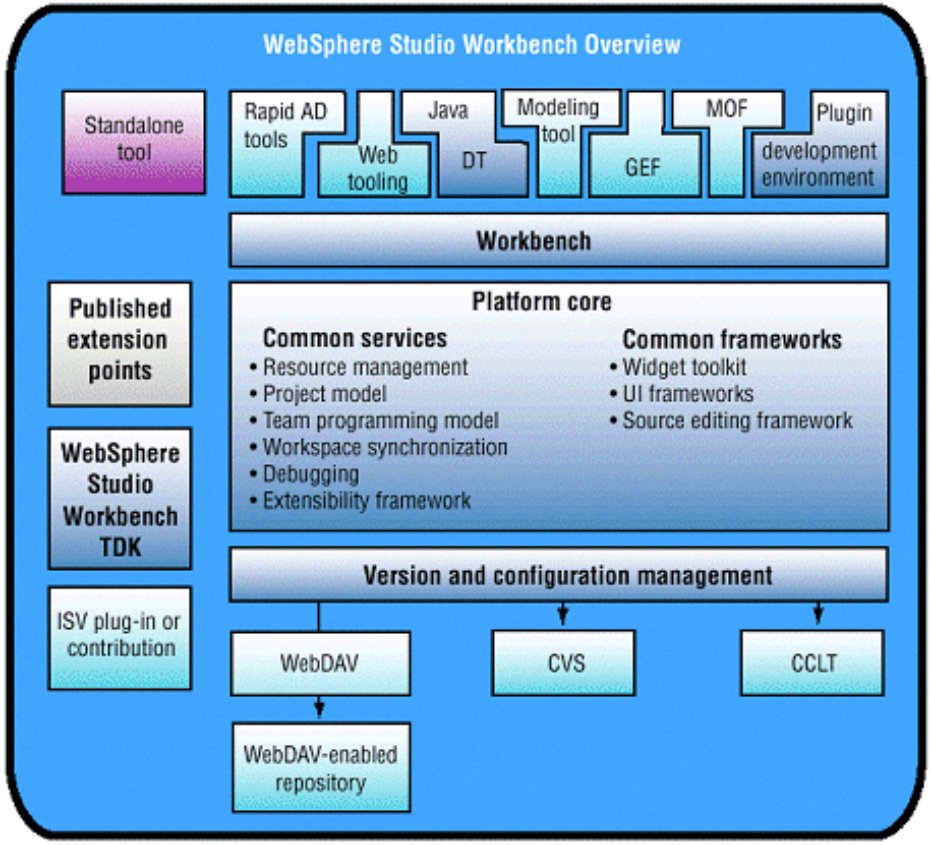


Figure 6-1 WebSphere Studio Workbench overview

6.1.1 Why a new development platform?

The WebSphere Studio Workbench is for tool builders. It provides the base platform for the new WebSphere Studio family of products and ISV tools. It provides services and frameworks that enable tool builders to focus on tool building, not on building tool infrastructure.

Application development requirements for e-business environments have evolved dramatically over the past few years. The shift from a business-to-consumer focus only to an inclusion of business-to-business processes has created a significant shift in complexity. Until now, software

vendors have provided vertical tools (modeling, development, debugging, versioning, etc.) forcing customers to integrate them. IBM has created the WebSphere Studio Workbench platform to provide the base for integration of all tools needed in an e-business development environment.

6.1.2 WebSphere Studio Site and Application Developer

The first two IBM offerings built on top of the WebSphere Studio Workbench are (see Figure 6-2):

- WebSphere Studio Site Developer
- WebSphere Studio Application Developer

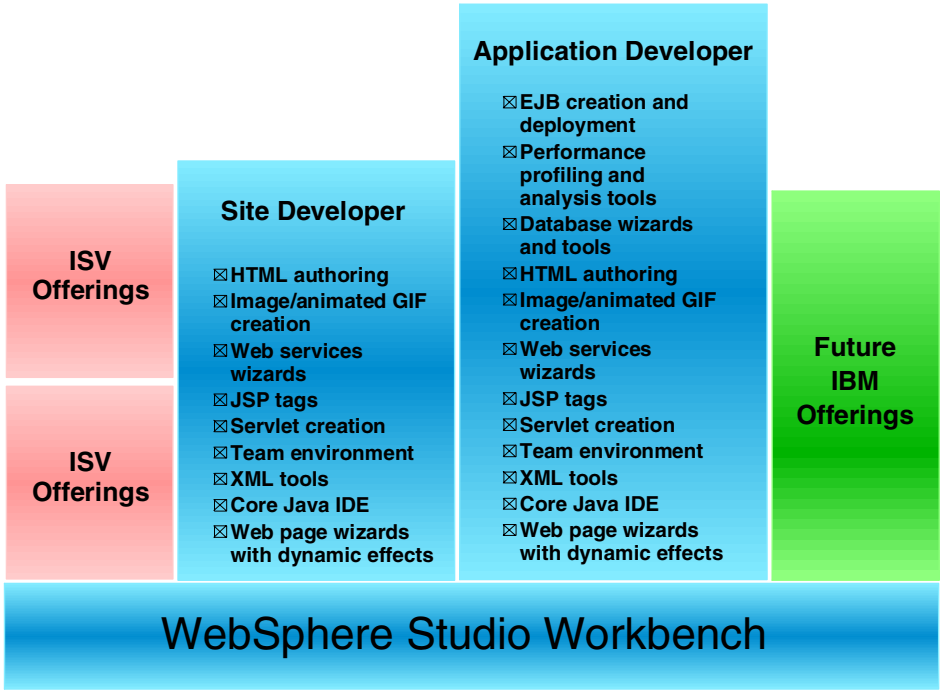


Figure 6-2 WebSphere tool offerings built on the WebSphere Studio Workbench

WebSphere Studio Site Developer (WSSD) is the follow-on to the WebSphere Studio Professional and Advanced edition products. WebSphere Studio Application Developer (WSAD) includes all of the functionality of WebSphere Studio Site Developer and adds support for Enterprise JavaBeans (EJBs), advanced monitoring and profiling tools, and relational database tools, and will become the follow-on product to VisualAge for Java.

Both of these tools are built on open standards and generate code complying with open standards. This includes support for:

- J2EE 1.2
  - EJB 1.1
  - Servlet 2.2
  - JSP 1.2
  - JRE 1.3
  - JCA (J2EE Connector Architecture)
- Web Services
  - WSDL 1.1 (Web Services Definition Language)
  - SOAP 2.1 (Simple Object Access Protocol)
  - UDDI
  - XML
- HTML 4.01
- CSS2

Consistent interface for multiple tools

One of the design goals of the WebSphere Studio Workbench is to provide for a consistent look and feel while maintaining role-based development. Previous WebSphere Studio products have provided support for the Web Page Designer focusing on the flow of an application and the user interface while VisualAge for Java products have supported the programmer developing business logic.

The new WebSphere Studio Site Developer and WebSphere Studio Application Developer products combine the features of WebSphere Studio and VisualAge for Java into a single integrated environment with a consistent user interface of the WebSphere Studio Workbench.

WebSphere Studio Site Developer is intended for Web developers who develop, organize, and manage complete Web sites. WebSphere Studio Application Developer includes all the functionality of WebSphere Studio Site Developer and adds support for developers working on business logic, such as EJBs. As IBM delivers future extensions to the WebSphere Studio Workbench, it will expand the range of options, matching products to user roles and requirements.

Common tools for testing and deployment

WebSphere Studio Site Developer and WebSphere Studio Application Developer provide a unit-test environment where you can test JSP files, servlets, and HTML files. They also provide the capability to configure other local or remote servers, for integrated testing and debugging of Web and EJB applications (EJB testing support is only available in WebSphere Studio Application Developer). The following server tools are included:

- A lightweight runtime environment that loads quickly
- Stand-alone unit testing
- Ability to debug live server-side code using the integrated debugger
- Support for configuring multiple Web applications

The testing and deployment tools support the following runtime environments, which can be installed locally or remotely:

- WebSphere Application Server V4.0.1, Single Server Edition supports testing of both EJB and Web applications
- Apache Tomcat - supports only Web applications

Team collaboration

Application development teams are becoming even more distributed, more diverse, and under more pressure to deliver solutions quickly. It is critical to have a development environment that can support these needs while also addressing personalized requirements. The team development environment for WebSphere Studio Site Developer and WebSphere Studio Application Developer includes open pluggable repositories instead of a proprietary repository. The move away from a proprietary repository to a file-based system offers the following improvements:

- Easier integration of favorite tools since you can work directly on the file system.
- More flexibility in source management and team development.
- Minimizes the duplication of data on the file system and the Workbench.

In WebSphere Studio Site Developer and WebSphere Studio Application Developer, team developers do all of their work in their individual workbenches, and then periodically make changes to a *team stream*. This model allows individual developers to work on a team project, share their work with others as changes are made, and access the work of other developers as the project evolves. At any time, developers can update their workbenches by retrieving the changes that have been made to the team stream or by submitting changes to the team stream.

The supported repository of the WebSphere Studio Workbench is the Concurrent Versions System (CVS). CVS uses an unreserved check-out model for version control to avoid artificial conflicts common with the exclusive check-out model. This allows more than one developer to work on the same files at the same time. In addition, any SCM (Software Configuration Management) provider can add an adapter to integrate their SCM system into the Workbench. Several industry-leading vendors have announced that they will do so.

6.1.3 Planned platform support - Linux and Windows

IBM has plans to provide the WebSphere Studio Workbench and WebSphere Studio products on the following platforms:

- Red Hat Linux 7.1 (Intel)
- Microsoft Windows 2000, NT 4.0, ME, 98, beta on XP

This statement of direction is significant as it positions Linux as an alternative development environment to Windows for creating the next generation of e-business applications.

6.2 WebSphere Studio Site Developer

WebSphere Studio Site Developer is the evolution of the WebSphere Studio products designed to J2SE and J2EE specifications. It supports the development of:

- JSPs
- Servlets
- HTML
- JavaScript
- DHTML
- XML files, schemas, and DTDs
- Web Services - self-contained modular applications that can be described, published, located, and invoked over the Internet
- Images and animated GIFs

WebSphere Studio Site Developer also allows Web developers to use their favorite content creation tools in conjunction with its built-in local and remote publishing capabilities

6.2.1 Web development environment

The development role WebSphere Studio Site Developer is tailored to is the Web site development team. This includes:

- Content authors
- Graphic artists
- JSP/Servlet programmers
- Web masters

The Web development environment features in WebSphere Studio Site Developer include:

- Support for latest Web technology with intuitive user interface
- Advanced scripting support to create client-side dynamic applications with Visual Basic or JavaScript
- WebArt Designer to create graphic titles, logos, buttons, and photo frames with professional-looking touches
- AnimatedGif Designer to create life-like animation from still pictures, graphics, and animated banners
- Over 2,000 images and sounds in a built-in library
- Integrated easy-to-use visual layout tool for JSP and HTML file creation and editing
- Web project creation, using the J2EE container structure
- XML file creation and editing
- Automatic update of links as resources are moved or renamed
- Servlet creation by means of a wizard
- Generation of Web applications from database queries and beans
- J2EE WAR/EAR deployment support

6.2.2 XML tools

XML (eXtensible Markup Language) has become one of the most important technologies enabling e-business. WebSphere Studio Site Developer provides a comprehensive visual XML development environment providing the following features:

- Create, view, and validate DTDs and XML schemas
- Create XML documents from a DTD
- Generate Java beans that navigate documents from the DTD or XML schema

- Define mappings between XML documents and generate XSLT scripts that transform documents
- Create an HTML or XML document by applying an XSL stylesheet against an XML document, using the Xalan processor
- Produce XML-related artifacts from an SQL query
- Define mappings between relational tables and DTD files
- Generate a document access definition (DAD) script, used by IBM DB2 XML Extender, to either compose XML documents from existing DB2 data or decompose XML documents into DB2 data
- Generate XML and related artifacts from SQL statements and use these files to implement your query in other applications

6.2.3 Web Services development environment

Web Services are module, standards-based components that businesses use to dynamically advertise and discover application functionality that can be assembled to solve business problems. Web Services represent the next level of e-business functionality and efficiency.

IBM's Web Services development tools are built upon the following open, cross-platform standards:

- **Universal Description Discovery and Integration for Java (UDDI4J)**, which enables businesses to describe themselves, publish technical specs on how they want to conduct e-business with other companies, and search for other businesses that provide goods and services they need, all via online UDDI registries.
- **Simple Object Access Protocol (SOAP)**, which is a standard for reliably transporting electronic business messages from one business application to another over the Internet.
- **Web Services Description Language (WSDL)**, which describes programs accessible via the Internet (or other networks), and the message formats and protocols used to communicate with them.

WebSphere Studio Site Developer provides functionality for the following Web Services development tasks:

- **Discover** - Browse the UDDI business registry to locate existing Web Services for integration.
- **Create or transform** - Create Web Services from existing artifacts, such as Java beans, URLs that take and return data, DB2 XML Extender calls, DB2 stored procedures, and SQL queries.

- **Develop** - Generate a sample application to assist you in creating your own Web service client application.
- **Build** - Wrap existing artifacts as SOAP and HTTP GET/POST-accessible services and describe them in WSDL. The tools also assist you both in generating SOAP and HTTP GET/POST proxies to Web Services described in WSDL and in generating Java bean skeletons from WSDL.
- **Deploy** - Deploy the Web service into the WebSphere Application Server or Tomcat test environments.
- **Test** - Test the Web service as it runs locally or remotely.
- **Publish** - Publish your Web Services to the UDDI business registry advertising your Web Services so that other businesses can access them.

6.2.4 The evolution of WebSphere Studio

WebSphere Studio Professional and Advanced Edition are the precursors to WebSphere Studio Site Developer. These robust Web development tools are widely used by teams who need powerful JSP and servlet support combined with an easy-to-use visual page designer. But to continue to provide customers with state-of-the-art Web development tools and to answer their requests for tighter integration with other technologies, IBM has moved to a new base that will be tightly integrated with VisualAge for Java technologies and other vendor tools.

WebSphere Studio Site Developer is the planned follow-on toolset for WebSphere Studio 4.0 and serves a broader spectrum of Web developer needs. Site Developer's new base includes an integrated Java development environment and a source configuration management support system. It also includes a fully integrated set of tools to support XML source development, XSL transformations, and Web Services development.

6.3 WebSphere Studio Application Developer

WebSphere Studio Application Developer includes all of the functionality of the WebSphere Studio Site Developer and adds support for the following:

- Enterprise JavaBeans (EJBs)
- Database application development
- Performance, profiling, and monitoring tools

6.3.1 Enterprise JavaBeans development environment

WebSphere Studio Application Developer features full EJB 1.1 support, an updated EJB test client, an enhanced unit test environment for J2EE, and deployment support for Web application archive (WAR) files and enterprise application archive (EAR) files. You can create multiple projects with different unit test configurations and share the instances with other developers. Entity beans can be mapped to databases, and EJB components can be generated to tie into transaction processing systems. XML provides an extended format for deployment descriptors within EJB.

WebSphere Studio Application Developer provides the following leading-edge EJB development tools:

- Tools for import/export, creation and code generation, and editing, as well as support for standard deployment descriptors and extensions and bindings specific to WebSphere Application Server.

6.3.3 Monitoring and profiling tools

WebSphere Studio Application Developer provides monitoring and profiling tools that feature customizable views and logs for recognizing, isolating, and fixing performance problems. The tools feature a customizable set of plug-ins for viewing logs. The performance analyzer feature enables you to test your application's performance early in the cycle. This gives software architects enough time to make architectural changes with the knowledge that developers will also have sufficient time to implement those changes. This reduces risk early in the cycle, and avoids problems in the final performance tests.

The performance analyzer feature helps you to visualize your program execution easily, and explore different patterns within the program. This tool is useful for performance analysis, debugging, and for gaining a deeper understanding of your Java program. You can use it to view object creation and garbage collection, execution sequences, thread interaction, and object references. It also displays which operations take the most time, and helps you to find and solve memory leaks. You can easily identify repetitive execution behavior and eliminate redundancy, while focusing on the highlights of an execution.

6.3.4 The evolution of VisualAge for Java

The IBM VisualAge for Java and WebSphere Studio development products provide a rich set of functions. VisualAge for Java has been very successful and is used by thousands of enterprises worldwide that are developing some of the most advanced Java applications in production. However, to continue to provide the leading development environment, IBM has created a new technology base that encompasses Web application development as well as Java development. This technology base is the WebSphere Studio Workbench and it provides, for the first time, for the Web application development team and the business logic development team to work together on a common platform.

WebSphere Studio Application Developer is the follow-on technology for VisualAge for Java Enterprise Edition. It is designed from the ground up to meet the requirements for all new types of applications. These requirements include:

- Open standards
- Java
- XML
- HTML
- Web Services
- Testing
- Integration with other components and ISV products

- Pluggability of new components
- Expandability
- Role-based development
- Increased usability for all users
- Enhanced team collaboration
- Increased speed to market

WebSphere Studio Application Developer will focus on J2EE server-side (EJB and servlet) development, deployment, and profiling. VisualAge for Java has a much wider scope, including client development using Swing via the Visual Composition Editor, Enterprise Access Builder, Domino AgentRunner, and Tivoli Connection. During 2001 and 2002 developers can use the products together to accomplish specific tasks, depending on their project requirements. VisualAge for Java will be bundled with WebSphere Studio Application Developer. When WebSphere Studio Application Developer becomes generally available, it will include VisualAge for Java Enterprise Edition Version 4.0 in the box. In future releases, more of the rich functionality of VisualAge for Java will be migrated to WebSphere Studio Application Developer.

6.3.5 PiggyBank in WebSphere Studio Application Developer

In this section, we import the Piggybank application into WebSphere Studio Application Developer to get a look at this new development environment.

1. Start WebSphere Studio Application Developer
2. Select **File** -> **Import**.
3. In the **Import** window, select **EAR File**, then select **Next**.
4. In the EAR Import window, specify the following:

   EAR File: `D:\piggybank\SG246134\sampcode\piggybank\piggybank.ear`

   EAR Project: `Piggybank`
5. Select **Next**.
6. In the EAR Selected Import window, select **All Files**, then click **Finish**.

You can view an application from different perspectives. Let's view the application from the J2EE perspective.

1. From the menu bar, select **Perspective -> Open -> Other...**.
2. Select the **J2EE** perspective and select **OK**.

The J2EE view, as seen in Figure 6-3, looks very similar to the views we have seen in the WebSphere Application Assembly Tool (AAT).

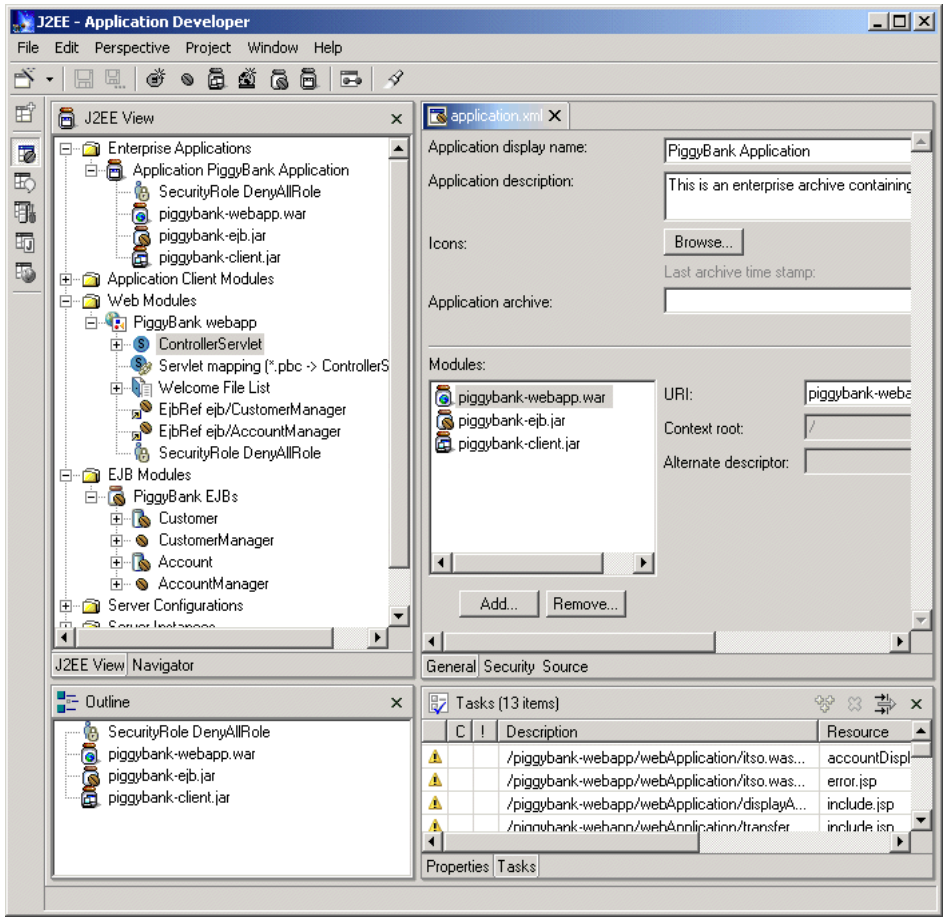


Figure 6-3 WebSphere Studio Application Developer - J2EE view

Like the AAT in WebSphere Studio, you can create all of your deployable J2EE modules from the WebSphere Studio Application Developer environment:

- WAR files
- EJB JAR files
- EAR files

WebSphere Studio Application Developer has an integrated WebSphere Test environment and an EJB Test Client allowing you to test your deployable assets within the development environment. You can also specify that you want to test your modules on a remote WebSphere application server.

Finally, let's take a look at the integrated Web and Java development environments of WSAD as seen in Figure 6-4 on page 132 and Figure 6-5 on page 133 on the Windows platform.

.

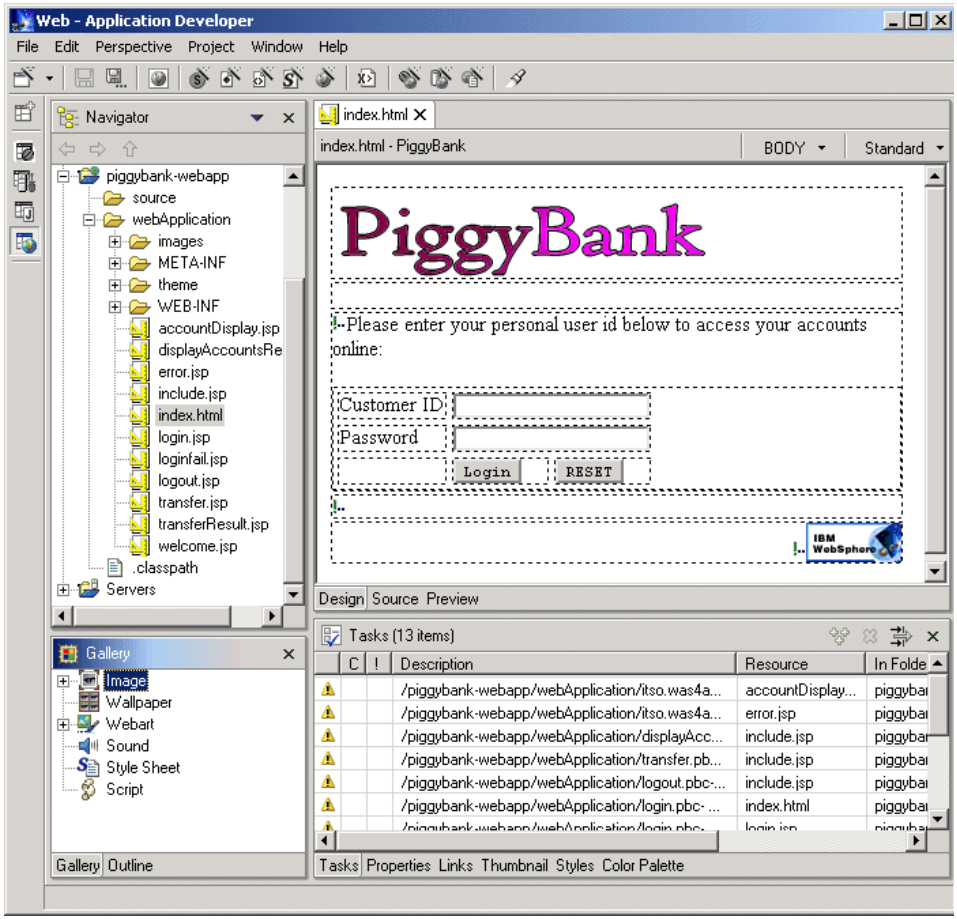


Figure 6-4 WebSphere Studio Application Developer - Web view

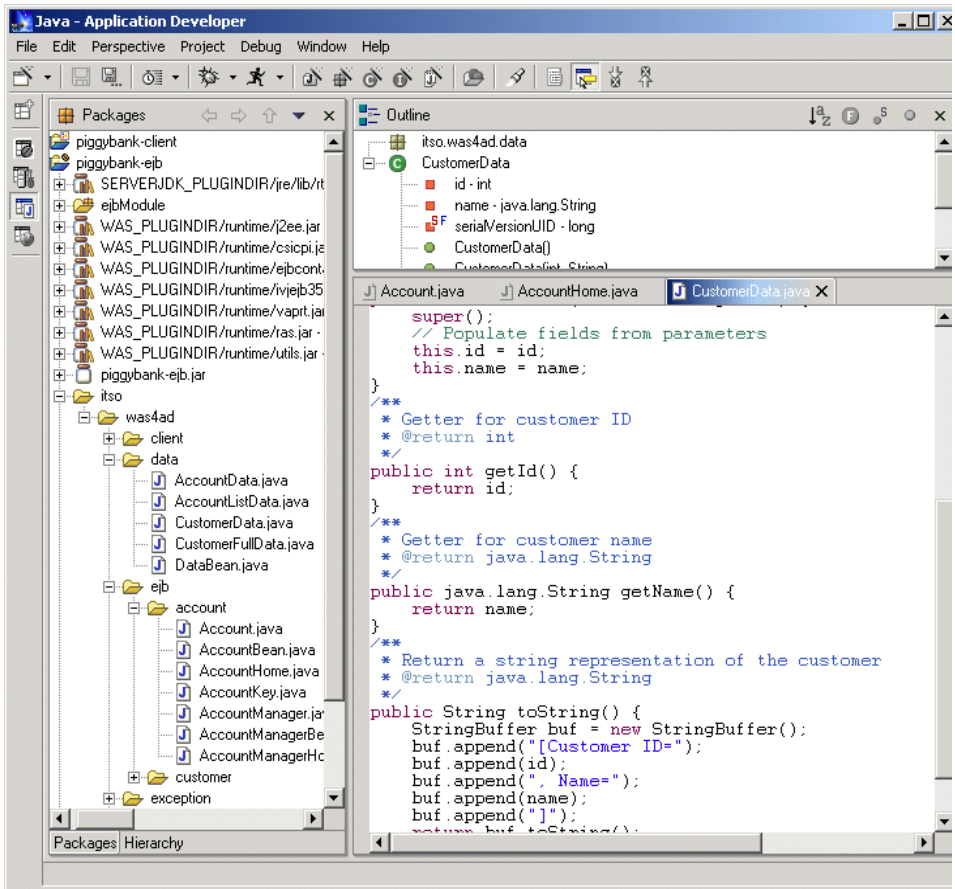


Figure 6-5 WebSphere Studio Application Developer - Java view

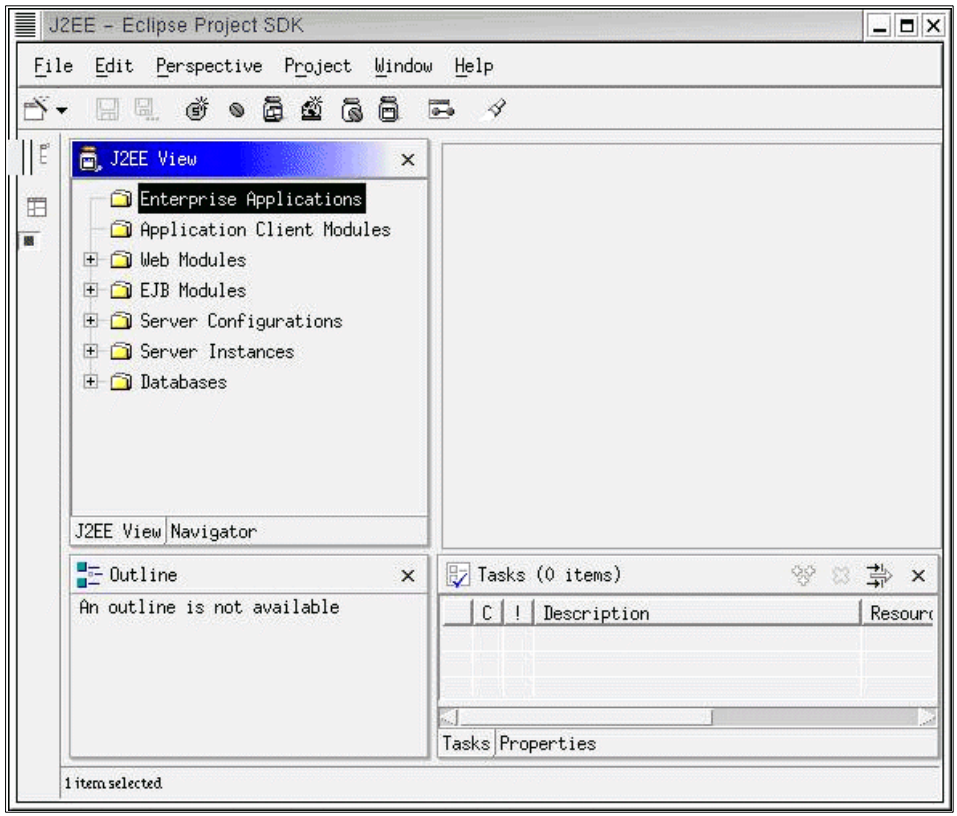


Figure 6-6 WebSphere Studio Application Developer for Linux

Figure 6-6 displays the WebSphere Studio Application Developer running on Red Hat Linux.

6.3.6 Summary

The WebSphere Studio Workbench platform combined with the new WebSphere Studio Site and Application Developer products integrates the best features of IBM's award-winning VisualAge for Java development environment and WebSphere Studio. The open and pluggable base of the Workbench will allow for unprecedented integration of development, modeling, team, testing, and performance tools for the next generation of e-business applications. Figure 6-7 provides a summary of the evolution and features offered in the WebSphere Studio Application Developer.

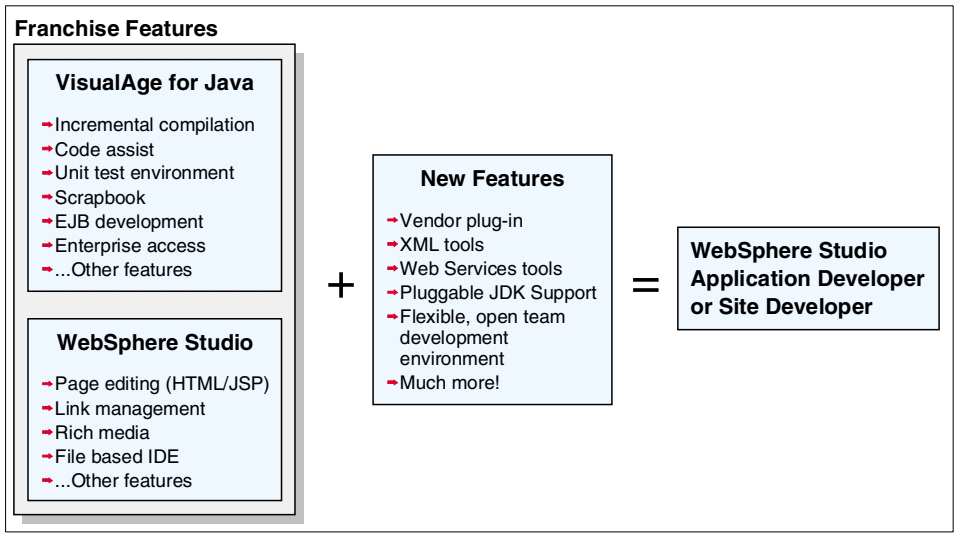


Figure 6-7 WebSphere Studio products - combined features of VAJ and Studio

Special notices

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

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

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

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

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

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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