

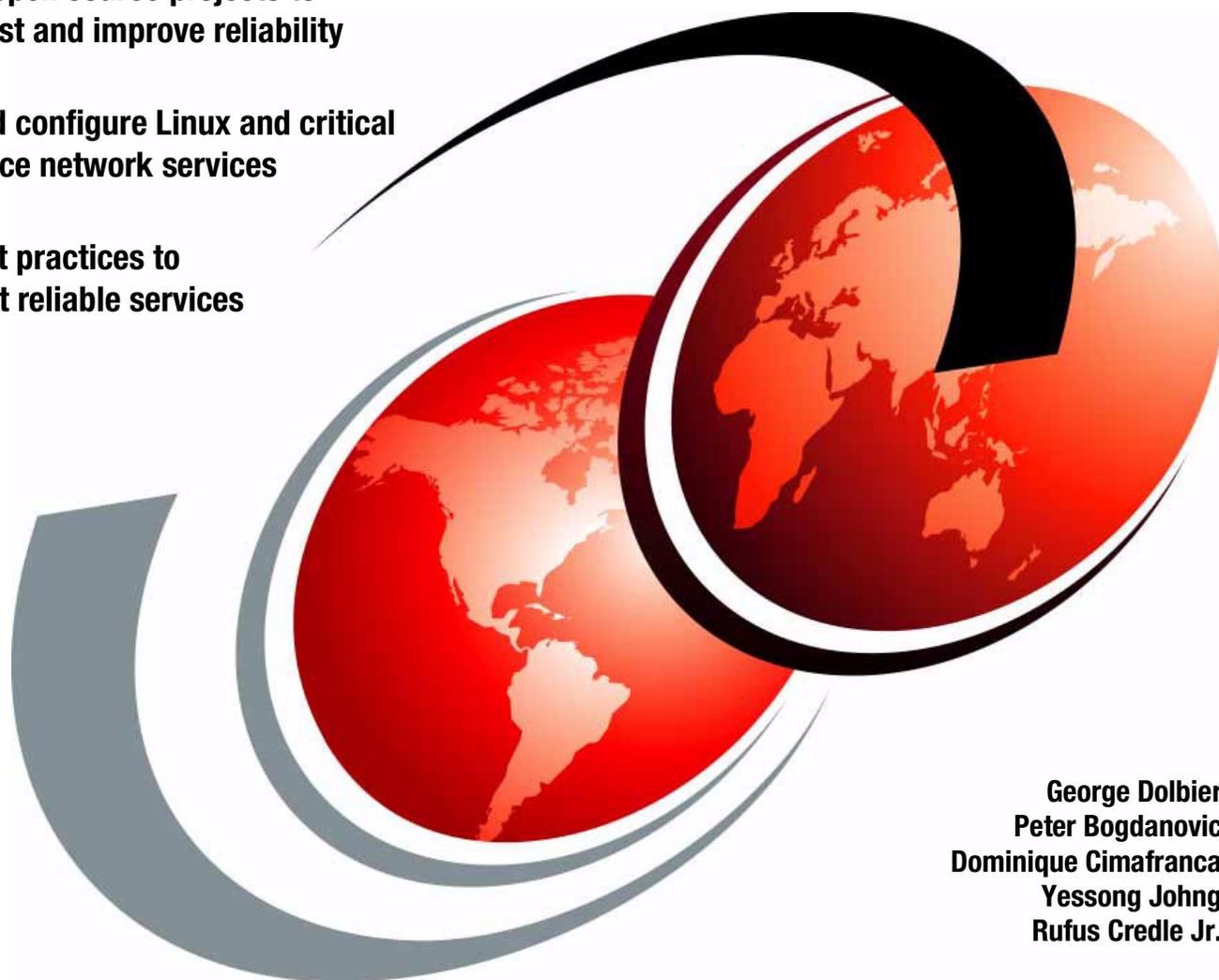
IBM *e*server BladeCenter, Linux, and Open Source

Blueprint for e-business on demand

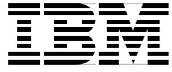
Discover open source projects to
reduce cost and improve reliability

Install and configure Linux and critical
open source network services

Learn best practices to
implement reliable services



George Dolbier
Peter Bogdanovic
Dominique Cimafranca
Yessong Johng
Rufus Credle Jr.



International Technical Support Organization

**IBM @server BladeCenter, Linux, and Open Source:
Blueprint for e-business on demand**

July 2003

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (July 2003)

This edition applies to Red Hat Advanced Server 2.1.

© Copyright International Business Machines Corporation 2003. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	vii
Trademarks	viii
Preface	ix
The team that wrote this redbook	ix
Become a published author	xi
Comments welcome	xi
Chapter 1. About the book: Blueprint for building an e-business application for BladeCenter	1
1.1 Building an e-business infrastructure	2
1.1.1 Materials	2
1.1.2 Objectives	3
1.2 IBM eServer™ BladeCenter	3
1.3 FAStT SAN storage	3
1.4 BladeCenter business value	4
1.5 Linux business value	4
1.6 Open source business value	4
1.7 Other references	5
Chapter 2. Architecture: Solution overview	7
2.1 Open source e-business infrastructure a modular approach	8
2.2 All construction projects start with a pattern	8
2.2.1 Industry standard e-business pattern: A three-tier infrastructure	8
2.3 Blade servers	8
2.3.1 The next evolutionary step in computing: Blade-based computing	9
2.3.2 IBM eServer BladeCenter	9
2.3.3 BladeCenter value	9
2.3.4 When BladeCenter is not the right platform	10
2.4 SAN storage	10
2.5 Software stack	10
2.5.1 High-level architecture	10
2.5.2 Open source e-business software components	11
2.5.3 Functional aspects	12
2.5.4 Non-functional requirements	13
2.5.5 Non-functional aspects	13
2.5.6 Detailed software stack	13
Chapter 3. Foundation	17
3.1 Hardware	18
3.1.1 Single CD-ROM, floppy drive, keyboard, video, and mouse	18
3.2 Installing operating system instances	18
3.2.1 PXE	19
3.2.2 Red Hat Kickstart	20
3.2.3 Sample Kickstart configuration for BladeCenter	22
Chapter 4. Plumbing: Network infrastructure	25
4.1 DHCP	26
4.1.1 Background	26

4.1.2	Building in fault tolerance	26
4.1.3	Security concerns	28
4.1.4	Conclusion	29
4.2	DNS	29
4.2.1	History	30
4.2.2	Building a highly available DNS	32
4.2.3	Conclusion	34
4.3	LDAP	34
4.3.1	LDAP servers	35
4.3.2	LDAP concepts	35
4.3.3	Working with OpenLDAP	39
4.3.4	gq: A graphical LDAP browser	45
4.3.5	Server authentication with LDAP	52
4.3.6	Apache authentication with LDAP	58
Chapter 5. Wiring: File services with Samba and NFS		61
5.1	Working with Samba	62
5.1.1	Required Samba packages	62
5.1.2	Configuring Samba as a basic file server	62
5.1.3	Adding Samba users	63
5.1.4	Samba passwords	63
5.1.5	Connecting to the Samba server using smbclient	64
5.1.6	Connecting to the Samba server using smbmount	64
5.1.7	Connecting to the Samba server from a Windows machine	64
5.1.8	Automatically mounting a Samba directory at boot time	64
5.1.9	Sharing additional directories	64
5.1.10	For more information on Samba	65
5.2	Working with NFS	65
5.2.1	Required NFS packages	65
5.2.2	Configuring NFS	65
Chapter 6. Doorways: Web serving and messaging		67
6.1	Web serving	68
6.1.1	The Apache Web server	68
6.1.2	Installing Apache HTTP Server Version 2.0	68
6.1.3	Installing Apache HTTP Server and the SSL module	68
6.1.4	Installing the Perl module	69
6.1.5	Installing the PHP module	70
6.1.6	Configuring and testing Apache	71
6.1.7	Load balancing and Linux Virtual Server (LVS)	73
6.1.8	Installing the Web cluster	74
6.1.9	Configuring the Web cluster	75
6.2	E-mail	83
6.2.1	How Internet e-mail systems fit together	83
6.2.2	Building an e-mail server with Sendmail and UW-IMAP	85
6.2.3	Replacing Sendmail with Postfix	90
6.2.4	Replacing UW-IMAP with Courier	92
6.2.5	Virtual users and domains with Courier and Postfix	94
6.2.6	Virtual mail servers with Postfix, OpenLDAP, and Courier	99
6.2.7	Dealing with spam and viruses	105
6.2.8	Sendmail clusters on Linux	113
6.3	Instant messaging	123
6.3.1	Instant messaging's value to modern companies	123

6.3.2	Jabber	123
6.3.3	Running a Jabber server	125
6.3.4	Using Jabber clients	128
6.3.5	Considerations for using jabberd for an intranet	135
6.3.6	Extending Jabber	137
Chapter 7. Living spaces: Applications and portal server		139
7.1	Web applications	140
7.1.1	Servlets	140
7.1.2	JavaBeans	140
7.1.3	JavaServer Pages	140
7.1.4	Containers	141
7.2	Tomcat	141
7.2.1	A brief history of Tomcat	141
7.2.2	Diving into Tomcat	142
7.2.3	Java Web applications	147
7.2.4	A Quick example: Jetspeed	149
7.2.5	The deployment descriptor: web.xml	150
7.2.6	Understanding Tomcat's configuration file	152
7.2.7	Using the Tomcat Web Application Manager	159
7.2.8	SSL with Tomcat	164
7.2.9	Integrating Tomcat and Apache	167
7.3	Portals	171
7.3.1	Jetspeed	171
Chapter 8. Cabinetry: Open source databases		183
8.1	PostgreSQL, MySQL, and others	184
8.1.1	PostgreSQL	184
8.1.2	MySQL	184
8.1.3	PostgreSQL versus MySQL	185
8.1.4	Other open source databases	185
8.2	Working with MySQL	185
8.2.1	Required MySQL RPM packages	186
8.2.2	Starting MySQL the first time	186
8.2.3	Securing MySQL	187
8.3	MySQL replication	188
8.3.1	Uses of replication	188
8.3.2	Setting up replication	188
8.4	Using MySQL replication	191
8.4.1	Load balancing MySQL queries with a workload manager	191
8.4.2	Application logic versus cluster logic	192
8.4.3	Example: Using application logic	192
8.4.4	Horizontal scaling and MySQL replication	193
8.4.5	High availability	194
8.5	What if the master fails?	195
8.5.1	Setting up a mutual master-slave relationship	195
8.5.2	Chaining servers	196
8.5.3	How far do we go?	197
Chapter 9. Security		199
9.1	Good practices	200
9.2	OpenSSH	201
9.3	Segregate networks	202
9.4	IPChains	203

9.4.1	Creating rules	203
Chapter 10. Household maintenance: System management and application development		
10.1	Simple Network Management Protocol (SNMP)	206
10.1.1	Configuring snmpd	206
10.1.2	Using snmp utilities	207
10.2	MRTG	207
10.2.1	Installing MRTG	208
10.3	Mon	209
10.3.1	Installing Mon	209
10.3.2	Configuring Mon	211
10.4	Eclipse	212
10.4.1	Getting started with Eclipse	213
10.4.2	Working with Eclipse	217
10.4.3	Tomcat plug-in for Eclipse	223
10.4.4	For more information	230
Related publications		
	IBM Redbooks	231
	Other publications	231
	Online resources	231
	How to get IBM Redbooks	234
	Help from IBM	234
Index		
		235

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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:

IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law. INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

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

@server™
@server™
eServer™
e-business on demand™
ibm.com®
iSeries™

xSeries®
AS/400®
BladeCenter™
DB2®
Informix®
IBM®

Redbooks(logos) ™
Redbooks™
Sequent®
Tivoli Enterprise™
Tivoli®
WebSphere®

The following terms are trademarks of other companies:

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

UNIX is a registered trademark of The Open Group in the United States and other countries.

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.

Preface

Every construction project relies on a few critical components. This is true whether you are building a house or an e-business on demand™ infrastructure. When building a house, the critical components include the foundation, plumbing, and electrical wiring. When building a computing environment, the critical components include a robust operating system, file, and network services.

Not long ago, building a robust e-business infrastructure in a “do-it-yourself” approach was rather daunting and reserved to a handful of IT enthusiasts. Now those of you who look for alternatives to expensive solutions based on commercial software and supported on large server farms, can benefit from using the techniques and technologies in this redbook.

This IBM® Redbook takes a modular approach to building an e-business on demand infrastructure. It covers many topics including Linux installation on IBM @server™ BladeCenter™ and IBM Fibre Array Storage Technology (FAStT) storage area network (SAN) storage. This redbook explains:

- ▶ How to implement failover for core Internet services such as domain name system (DNS), Dynamic Host Configuration Protocol (DHCP), and Lightweight Directory Access Protocol (LDAP)
- ▶ How to use a single LDAP directory for Linux system accounts, Apache, Samba, Postfix, Sendmail, and Jetspeed
- ▶ An implementation of load balanced services using Linux Virtual Server (LVS), and failover with Linux Heartbeat
- ▶ How to install and configure critical file services using Linux, Network File System (NFS), Samba, and IBM FAStT storage
- ▶ Practices for security, systems management, configuration, and performance

If you are looking to reduce the cost of your computing infrastructure, provide critical IT services, install Linux on BladeCenter blades, and install and configure SAN storage with Linux and BladeCenter, this redbook is for you.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Rochester Center.

George Dolbier is a Senior Consulting IT architect with over 15 years experience in various parts of the high technology industry. He spent most of his career as a software engineer, notably for Oracle, Informix®, and Sequent® Computer Systems. Prior to joining IBM, he was Director of Engineering for a small .COM, directing the development of Web-based collaboration software. He came to IBM via the IBM/Sequent merger in 1999 where he was a member of the IBM Application Service Provider (ASP) and xSP project teams. Since then, he has worked in helping IBM customers incorporate Linux and open source technologies into their operations.

Peter Bogdanovic works in the Linux for Service Providers Lab for IBM in Beaverton, Oregon. Prior to joining IBM two years ago, he worked as a software integrator and system

administrator for over ten years. Among his achievements are building the UNIX® network for a regional telephone carrier.

Dominique Cimafranca is a Linux IT Specialist from IBM Philippines. A long-time advocate of open-source in southeast Asia, Dominique writes a weekly online column on Linux for a national daily and contributes to technical journals. He has worked for IBM for six years.

Yessong Johng is an IBM Certified IT Specialist at the IBM ITSO, Rochester Center. He started his IT career at IBM 20 years ago as a S/38 Systems Engineer in 1982 and has continued his work on the AS/400® and now IBM @server iSeries™. He writes extensively and develops and teaches IBM classes worldwide on the areas of e-business on iSeries. His major responsibilities are Linux and WebSphere® implementation on iSeries.

Rufus Credle Jr. is a Senior I/T Specialist and certified Professional Server Specialist at the IBM ITSO, Raleigh Center. He conducts residencies and develops Redbooks™ about network operating systems, ERP solutions, voice technology, high availability and clustering solutions, Web application servers, pervasive computing, and IBM and OEM e-business applications, all running IBM @server xSeries® and BladeCenter systems. Rufus' various positions during his IBM career have included assignments in administration and asset management, systems engineering, sales and marketing, and IT services. He holds a Bachelor of Science degree in business management from Saint Augustine's College. Rufus has been employed at IBM for 23 years.

This redbook is built on, and collects the works from, a diverse team. Thanks to the following people for their contributions to this project:

Jeff Chui
IBM China (Hong Kong S. A. R.)

Jay Allen
Connie Blauwkamp
Pete Jordan
Robert MacFarlan
Rich McDevitt
Mark Nellen
Norm Patten
IBM Beaverton

Scott Knupp
IBM White Plains

Justyna Nowak
IBM Philadelphia

Larry O'Connell
IBM Piscataway

Cristina Zabeu
IBM Raleigh

Cliff White
Open Source Development Laboratory

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this 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:

IBM Corporation, International Technical Support Organization
Dept. JLU Building 107-2
3605 Highway 52N
Rochester, Minnesota 55901-7829

1.1 Building an e-business infrastructure

Many people can relate to the materials and techniques involved in building a house. The materials of concrete, stone, steel, and wood are all very familiar. However, new high-tech materials may not be as widely known. The process of constructing a house is familiar to many. It includes laying a foundation, building walls, and putting on a roof. Advanced construction techniques may not be as familiar, such as the techniques used to harden homes against earthquakes.

Creating a cost effective, secure computing environment for businesses bares many similarities to construction projects. Both projects should start with a good architectural design pattern, a good working knowledge of the materials, and a good understanding of the techniques used in construction.

For example, modern s computing environments heavily depend on many services. The number of these services can be surprising. They include e-mail, domain name system (DNS), Lightweight Directory Access Protocol (LDAP), Dynamic Host Configuration Protocol (DHCP), file services, print services, Web serving, and application serving. Each service is critical to the computing infrastructure. The classical deployment model for network services is to install and configure each service on their own redundant servers. This deployment model has a very high total cost of ownership (TCO), and not just in capital. Each system consumes network, power and other costly resources, and must be managed.

Modern trends in network computing, namely blade-based servers, and open source software allow for a fundamental change in the deployment model of critical services. High performance blade-based servers allow you to deploy multiple services on a single pair of systems without sacrificing performance, capacity, availability, or security. Open source network services save you capital up front, and if deployed properly, they can have very low maintenance cost.

Another good practice is to build with a modular design in mind. Even though the services we describe are all inter-related, we show you how to deploy and configure them independently. This book allows you to look at the table of contents, find a chapter that discusses a specific topic, and jump right to that section of that chapter without reading the entire book.

If you want to reduce the cost of your critical network infrastructure, or deploy new infrastructure components, such as integrating LDAP into your environment, this redbook is for you.

We meet the goals of the redbook by demonstrating a fully functional solution based on open source software components implemented on IBM @server BladeCenter with IBM Fibre Array Storage Technology (FASTT) storage area network (SAN) storage, using the Linux operating system. In our solution, open source software provides a basic set of business computing services. BladeCenter and FASTT uniquely combine high performance computing, capacity, management ease, and dense form factors. This creates a strong, long lasting base that any computing infrastructure can build.

1.1.1 Materials

The foundation this redbook is made up of BladeCenter and IBM FASTT SAN storage. These two materials, when mixed together, create a strong, long lasting, material that any computing infrastructure can be built on. Open source software, such as Linux, Sendmail, MON, Multi-Router Traffic Grapher (MRTG), are the materials that make up the rest of our construction project.

The materials consist of the following components:

- ▶ Open source software
- ▶ Linux operating system
- ▶ BladeCenter
- ▶ IBM FASTT SAN storage

1.1.2 Objectives

Every construction project is built to meet a set of objectives. Using the tools and techniques in this redbook allows you to build a computing infrastructure with these objectives in mind:

- ▶ Provide critical network services
- ▶ Leverage the capabilities inherent within a state-of-the art computing platform
- ▶ Provide the critical operational characteristics of reliability, high availability, and scalability
- ▶ Minimize licensing and implementation cost
- ▶ Optimize return on investment (ROI)
- ▶ Minimize management and maintenance costs

1.2 IBM eServer™ BladeCenter

Blade servers are a relatively new technology that has captured industry focus because of their high density, high power, and modular design, which can reduce cost. This cost reduction comes with a more efficient use of valuable floor space, reduced network and power infrastructure requirements, and simplified management.

All of these features can reduce the cost of deployment, reprovisioning, updating, and troubleshooting. The cost savings comes from the fact that modern computing environments are often made up of hundreds of servers. With that many systems, even simple infrastructure, such as network cabling, can become very expensive. Blade-based computing reduces the amount of infrastructure required to support large numbers of servers. By integrating resources and sharing key components, costs are reduced and availability is increased.

1.3 FASTT SAN storage

IBM FASTT solutions are designed to support the large and growing data storage requirements of business-critical applications. The FASTT storage server is a Redundant Array of Independent Disks (RAID) controller device that contains Fibre Channel (FC) interfaces to connect the host systems and the disk drive enclosures.

The storage server provides high system availability through the use of hot-swappable and redundant components. The storage server features two RAID controller units, redundant power supplies, and fans. All of these components are hot-swappable, which assures excellent system availability. A fan or power supply failure does not cause downtime, although such faults can be fixed while the system remains operational. The same is true for a disk failure if fault-tolerant RAID levels are used. With two RAID controller units and proper cabling, a RAID controller or path failure does not cause loss of access to data.

The disk enclosures can be connected in a fully redundant manner, which provides a very high level of availability. On the host side FC connections, you can use up to four minihubs. The storage server can support high-end configurations with massive storage capacities (up to 33 terabytes (TB) per FASTT controller) and a large number of heterogeneous host systems. It offers a high level of availability, performance, and expandability.

1.4 BladeCenter business value

BladeCenter has a very concrete and specific business value. When your computing needs call for a dozen or so servers, you become concerned about the real-estate costs and the maintenance costs of those systems. After these issues become a concern, blade-based computing becomes valuable. This is due to its reduced real-estate and maintenance costs when compared to traditional, or even rack-optimized form factors.

As the number of systems you have to manage grows, your plumbing and wiring complexity grows as a multiple of the number of systems you manage. You can almost say that blade servers are to computing environments as brownstone apartments are to urban environments. Both technologies allow for the efficient delivery and management of services to a moderately large community.

1.5 Linux business value

Much has been written about the value of Linux to businesses. This redbook is based on the proposition that Linux is a stable, flexible, and cost-effective operating system that can be used as the foundation on which to build business-oriented information technologies.

Many of the services we document come with Linux distributions. However, we have made an effort to document how you can obtain them independently. This flexibility gives you the ability to easily tailor an environment to suit your own needs.

1.6 Open source business value

Linux is but one open source project. It is arguably the largest open source project and rightfully receives most of the attention of the media and technical community. However, it is still just one component of an overall architecture. Much of this redbook is concerned with the components that make up open source information architecture for business and technical computing.

All of these components, Linux included, share the same fundamental traits that differentiate open source software. That is the source code for the software is openly available, the source code can be modified, and the source code can be redistributed (subject to the terms of the license governing each component). These components include Web and application serving, application development, system security and management, and communications. Each component is developed by a supportive and collaborative development community. The software can often be acquired at no upfront cost. These characteristics help open source software to deliver value to its customers.

1.7 Other references

This redbook builds upon the excellent work of other IBM teams. If you intend to follow the instructions in this redbook, as if it were a blueprint, you must obtain the following Redbooks and Redpapers before proceeding:

- ▶ *Deploying Samba on IBM @server BladeCenter*, REDP3595
- ▶ *The Cutting Edge: IBM @server BladeCenter*, REDP3581
- ▶ *Implementing Linux with IBM Disk Storage*, SG24-6261
- ▶ *Linux Application Development Using WebSphere Studio 5*, SG24-6431
- ▶ *Linux Handbook: A Guide to IBM Linux Solutions and Resources*, SG24-7000



Architecture: Solution overview

Every successful building project must start with a good architecture. The same is true for information technology projects. This chapter introduces our architecture and the major components of our open source infrastructure.

2.1 Open source e-business infrastructure a modular approach

The value of any system is enhanced when the system can be broken down into discreet components that are then replaced or reused elsewhere. We know that the entire system we document in this redbook is not applicable to all situations. Our intent is to document best practices and implementation procedures in a modular fashion. This approach allows you to implement sections of this redbook independently to suit your needs.

2.2 All construction projects start with a pattern

Most suburban American homes are built around basic *architectural patterns*. For example, you can consider the popular ranch style home an architectural pattern. This pattern features a living room and open floor plan. The pattern includes structural features such as a concrete foundation and low-pitched roof.

Similarly, many modern business applications are built on top of a very common architectural pattern. We call that pattern the *three-tier e-business pattern*.

2.2.1 Industry standard e-business pattern: A three-tier infrastructure

The rapid pace of all technology-related industries has driven the use of standards and well-specified components designed for reuse. In the construction of software, these approaches gave rise to object-oriented software development, design patterns, and component-based development. The concept of software design patterns was first published in *Design Patterns: Elements of Reusable Object-Oriented Software* by Eric Gamma, Richard Helm, Ralph Johnson, and John Vlissides.

The software design patterns were inspired by the idea of patterns in the design of buildings, published in *A Pattern Language: Towns, Buildings, Construction* by Christopher Alexander, Sara Ishikawa, and Murray Silverstein. In the software industry, design patterns have gained acceptance by software architects and software engineers alike. The pattern concept has been applied to systems architecture in *Design Patterns: Elements of Reusable Object-Oriented Software*. This book leverages work done by IBM to advance this area.

The *Patterns for e-business* aim to communicate, in a highly accessible fashion, the business pattern, systems architecture (application and runtime topologies), product mappings, and guidelines required for different classes of applications. The patterns themselves are a group of proven, reusable assets that can help speed the process of developing applications.

2.3 Blade servers

Blade servers are a relatively new technology that has captured industry focus because of its modular design. This design can reduce cost with more efficient use of valuable floor space, reduce network infrastructure, and simplify its management. This can help to speed up such tasks as deploying, reprovisioning, updating, and troubleshooting hundreds of blade servers. All this can be done remotely with one graphical console using IBM Director systems management tools. In addition, blade servers provide improved performance by doubling current rack density. Integrating resources and sharing key components reduces costs, while increasing availability.

2.3.1 The next evolutionary step in computing: Blade-based computing

Compared to their predecessors, computers today are smaller and faster. Each generation adds more computing power and reduces the overall physical size of a system. In the relatively short history of computing, specialized computers have gone from the size of warehouses to the size of a matchbox. Generalized servers have also followed this trend.

For the last few years, the 1U rack mount server has been the workhorse of large scale computing. With a predictable pace, the market pressures of cost reduction are driving system vendors to provide ever smaller server platforms. The current state of the art is blade-based server technology. This type of system removes much of the frame around individual systems, while aggregating many of the services and cabling common to a rack of systems.

2.3.2 IBM eServer BladeCenter

There are two basic features of all blade-based computing platforms, the blade and the chassis. The *blade* houses main memory, CPU, and core input/output (I/O) components and peripheral components.

Blades plug into a *chassis*. The chassis provides consolidated electrical power, networking, and other services. In the case of the BladeCenter server platform, the chassis provides redundant power and networking as well as shared peripherals. These may include CD-ROM and floppy disk drives, as well as an integrated Keyboard Video Mouse (KVM) switch. The BladeCenter chassis can support up to 14 blades and is seven standard units (U) high.

2.3.3 BladeCenter value

When implementing typical server-based applications, a major consideration is determining the right “size of the box”. For example, you buy a single box that is large enough to handle the load of your application. If the application’s utilization grows, you need to add more memory, CPU, or I/O resources to your single box. If application utilization continues to grow, eventually you run out of capacity and need to buy a bigger box. This strategy is typically called “scale-up”.

There is another strategy that is common for applications that expect to grow very quickly or unpredictably. This strategy has drawbacks if your application needs to grow very quickly or exponentially, or if its growth is unpredictable.

An alternate strategy is to decompose an application into functions and deploy those functions across many networked systems. This strategy allows an application to grow asymmetrically. That is, you can add resources only to where they are needed, such as in the presentation layer. This strategy often referred to as “scale-out”.

To implement the scale-out strategy, you ideally want a standards-based server platform. This type of platform requires very little to install and configure, is packaged in a small form factor, and is relatively inexpensive. This type of application has driven server platforms to become smaller and more modular.

For many years now, 1U servers have been available in the market. This form factor allows roughly 48 systems to be installed in a standard 19-inch rack. For many applications, this level of density still requires considerable cost in floor space, management, networking, power, and heat.

To provide servers in a higher density requires a new paradigm in server design. This new paradigm is *blade-based servers*. BladeCenter currently doubles the physical server density

of 1U servers. In addition, BladeCenter can provide a 14 to 1 (14:1) reduction in network infrastructure, console cabling, and storage area network (SAN) connectivity.

In summary, BladeCenter allows for a very cost effective scale-out approach to application deployment.

2.3.4 When BladeCenter is not the right platform

BladeCenter is not a panacea for all IT problems. There are some situations where BladeCenter does not fit. Specifically small deployments, that will not grow, do not make sense for blades.

The current rule of thumb (as of publication of this redbook) is for nine systems, which is roughly the break-even point. This break-even point refers to the cost of BladeCenter blades and the chassis, when compared to rack-optimized servers.

Therefore, if you have a system that requires less than nine servers, BladeCenter may not be a cost-effective solution. With all things, there are extenuating circumstances. BladeCenter may make sense for a small deployment if you need the infrastructure to grow very large, very fast, or both.

2.4 SAN storage

The direct-attach storage capacity in blade-based computing solutions is limited by the very small nature of the blades themselves. This drawback has the potential to limit the applicability for blade-based computing.

Fortunately BladeCenter provides an alternative. BladeCenter blades can attach to a gigabit fibre SAN. This ability is critical for implementing high I/O applications, such as database and failover applications, that require access to shared disk.

IBM produces a complete line of fibre attach SAN products. For this redbook, we use the IBM FAST products to provide shared storage. The IBM FAST provides a reliable, manageable, and performing storage solution for both database and clustered applications.

2.5 Software stack

This redbook documents how to implement an infrastructure that can support a wide variety of activities and applications. This framework best supports applications that can be broken into a grid model or a n-tier model. This section provides an overview for the rest of the redbook.

2.5.1 High-level architecture

The bulk of Internet applications is designed, developed, and deployed using this pattern. The majority of this redbook deals with the technical details of implementing an open source framework that supports this architecture.

The architecture is broken down into three basic tiers that roughly match the classic Model-View-Controller (MVC) architectural pattern developed at Xerox PARC for Smalltalk-80. The three tiers are:

- ▶ **Network edge:** Systems in this tier are the most accessible of all three tiers. Users can directly access all the services provided by systems within this tier. For this reason, and many others, this tier is the most susceptible to security breaches and attacks.

Typically, there is only one protocol firewall between the network edge and the outside world. Often a VPN server also provides additional secure access to servers within this tier. When an application is deployed in this pattern, presentation logic is deployed and served from this tier. For Web applications, this tier is where the Web servers go.

- ▶ **Demilitarized zone (DMZ):** This tier is traditionally the domain of application or business logic. For Web applications, this tier is home to the application server. In our model, this tier is also home to the systems management systems and the application development systems. This tier is more secure than the network edge tier because the systems are not directly accessed by any general user community. Most of the services provided by this tier are actually services to the network edge systems.
- ▶ **Data management:** This tier is home to databases. The sole function of systems in this tier is to protect and serve data.

Each of these tiers are implemented on separate hardware and each tier is separated by firewalls. See Figure 2-1.

In keeping with our construction theme, you can think of these tiers as different rooms within a restaurant. If you are serving dinner to customers, they have to come in through the front door (outer firewall). They proceed to the dining room where they are served (presentation tier). The dinner is prepared in the kitchen, which is often behind another door. Customers do not have direct access (logic tier) to the kitchen. Finally all the food is stored in refrigerators, cabinets, and pantries (behind yet more doors), which are only accessible by the kitchen staff.

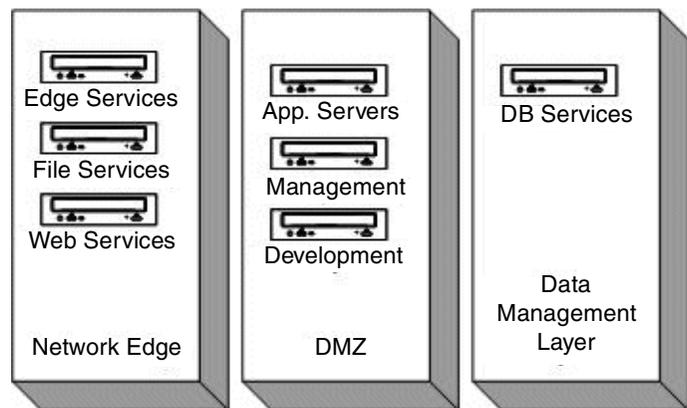


Figure 2-1 High-level architecture

Several services run on clusters that leverage fibre-attached *shared storage*. Shared storage is the fundamental technology that allows us to build clustered services. If you have never dealt with shared SAN storage, the concept is pretty simple. Your disk drive is housed and managed by a separate, very reliable, very fast computer. To your system, the SAN looks like any other disk drive. What your system does not know is that the disk is actually connected to a special switched storage network. Like any resource on a network, the disks can be concurrently shared by multiple systems. This ability is provided by significant intelligence in the SAN storage manager (sometimes called the *switch*).

2.5.2 Open source e-business software components

This section briefly explains the software stack and why each component was chosen. There are a few general rules of thumb used to select the software used in this infrastructure.

These are the criteria we used for selecting the components that make up our solution:

- ▶ The infrastructure component is in open source.
- ▶ The infrastructure component is used in production in customer accounts.
- ▶ The infrastructure component has a utility to a broad application set.

- ▶ The infrastructure component has an active support community.
- ▶ The open source infrastructure component functionality can be performed by commercial products.

2.5.3 Functional aspects

This section details the functional features, or aspects, of the infrastructure:

- ▶ **Network**
 - DNS
 - DHCP
 - BOOTP
 - PXE
 - NFS/CIFS
 - TFTP
 - Kickstart
- ▶ **Authentication**
 - LDAP (SLAPD, GQ)
 - PAM
 - NSS
 - SSH
- ▶ **High availability**
 - LVS
 - Failover File Services
- ▶ **Application services**
 - Apache
 - Tomcat
 - MySQL
 - Jetspeed
 - Mail transfer agent (MTA, selection TBD)
 - OpenConnect
 - Java
 - JSSE
 - Mod_perl
 - Mod_jk
 - Ant
- ▶ **Management**
 - MRTG
 - UCD SNMP
- ▶ **Security**
 - IPChains
 - Mod_ssl
 - SpanAssassin
- ▶ **Messaging**
 - Postfix
 - Jabber
 - Sendmail
 - Postfix
 - WU-IMAP

2.5.4 Non-functional requirements

For this architecture, the choices made in selecting components and deployment were driven by several requirements that are not directly related to the *functionality* of the system. Many of our rules of thumb for component selection are non-functional requirements.

Operational components: These are software and hardware systems that implement specific functionality.

The non-functional requirements are:

- ▶ All software infrastructure components must be open source.
- ▶ The primary hardware platform is BladeCenter.
- ▶ All infrastructure components must appeal to customers outside the xSP segment.
- ▶ The infrastructure component is being used in production in customer accounts.
- ▶ The infrastructure component has utility to a broad application set.
- ▶ The infrastructure component has an active support community.
- ▶ The open source infrastructure component functionality can be performed by commercial products.
- ▶ Modularity: All functional and operational components are documented in a manner that allows a reader to reproduce the implementation of the operational component without implementing the entire infrastructure. The interdependency of functional components is minimized.
- ▶ Robustness: Relevant functional components are implemented in a basic high availability fashion. A secondary goal is to document how to implement critical functional components in a high availability failover situation.
- ▶ Manageability: All functional components exist in a managed environment.

2.5.5 Non-functional aspects

This infrastructure meets several non-functional aspects, but the functional aspects are relatively few:

- ▶ Robust critical network infrastructure
- ▶ Robust Lightweight Directory Access Protocol (LDAP) authentication, directory services
- ▶ Robust file serving for Windows® and UNIX clients
- ▶ Robust messaging infrastructure (e-mail and instant messaging)
- ▶ Robust Web portal infrastructure
- ▶ Management and monitoring

2.5.6 Detailed software stack

The following sections explain the software stack and why each component was chosen.

OpenLDAP

OpenLDAP is an open source implementation of the LDAP RFC standards. The OpenLDAP project is an attempt to produce an open source LDAP implementation that is robust and fully functional. OpenLDAP comes with a complete set of tools that allows the implementation and deployment of an LDAP-based directory.

Postfix

Postfix is a mailer written by Wietse Venema. Postfix was developed to create a fast, secure, and easily administered alternative to Sendmail.

Apache

Apache, according to many sources, is the most popular Web server on the Internet. The Apache HTTP Server project is the Web server's official name. Apache is just one of several projects developed by the Apache Software Foundation.

Open SSL and MOD_SSL

Secure Sockets Layer (SSL) is a security protocol commonly used to secure HTTP transactions and Web sites. OpenSSL is an open source development effort aiming to provide a robust full featured implementation of the SSL V2 and V3 specifications. The OpenSSL project contains a full-strength general purpose cryptography toolkit.

MOD_SSL is an Apache module that allows a Web server to provide secure communications using all open source technologies.

Linux Virtual Server (LVS) (load balancing, high availability)

Linux Virtual Server is a scalability and availability technology that allows applications to leverage the power of cluster or grid computing. LVS provides load balancing and failover functionality in a fashion that is transparent to users running applications on an LVS system.

DNS, BIND, DHCP

These three protocols are core to any network built using Internet technologies. Support for these protocols is provided by services within the Linux operating system itself. All Linux distributions contain *daemons* that support these protocols. This redbook shows how to implement these protocols in a robust fashion.

Domain Name System (DNS)

The DNS is a distributed Internet directory service. DNS is used mostly to translate between domain names and Internet Protocol (IP) addresses, and to control Internet e-mail delivery. Most Internet services rely on DNS to work. If DNS fails, Web sites cannot be located and e-mail delivery stalls.

The DNS directory service consists of DNS data, DNSs, and Internet protocols for fetching data from the servers. The billions of *resource records* in the DNS directory are split into millions of files called *zones*. Zones are kept on *authoritative servers* distributed all over the Internet, which answers queries according to the DNS network protocols. In contrast, *caching servers* simply query the authoritative servers and cache any replies.

Most servers are authoritative for some zones and perform a caching function for all other DNS information. Most DNSs are authoritative for just a few zones, but larger servers are authoritative for tens of thousands of zones.

Dynamic Host Configuration Protocol (DHCP)

DHCP is an Internet protocol for automating the configuration of computers that use TCP/IP. DHCP can be used to:

- ▶ Automatically assign IP addresses
- ▶ Deliver TCP/IP stack configuration parameters, such as the subnet mask and default router

- ▶ Provide other configuration information such as the addresses for printer, time, and news servers

Berkeley Internet Name Domain (BIND)

DNS is actually implemented by a program called BIND. BIND is the Linux standard implementation. It is an implementation of the DNS protocols. It provides an openly redistributable reference implementation of the major components of the DNS, including:

- ▶ A DNS (named)
- ▶ A DNS resolver library
- ▶ Tools to verify the proper operation of the DNS

The BIND DNS is used on the vast majority of name serving machines on the Internet. It provides a robust and stable architecture on top of which an organization's naming architecture can be built. The resolver library included in the BIND distribution provides the standard application programming interfaces (APIs) for translation between domain names and Internet addresses. It is intended to be linked with applications requiring name service.

Systems management tools MON and MRTG

MON is a general-purpose systems management and monitoring tool. It can be used to monitor services and send alerts upon failure detection. MON was designed to be flexible and provides an extension API available to C, Perl, Shell, and other technologies commonly used by UNIX systems administrators.

Multi Router Traffic Grapher (MRTG) is a network traffic monitor. Where SNORT examines network traffic for security breaches, MRTG monitors and displays network utilization. MRTG is especially useful because it generates its reports as HTML pages. Its utility is enhanced by the fact that it runs on Windows operating systems and various UNIX operating systems, as well as Linux.

Tomcat

Tomcat is the reference Java servlet container and JavaServer Page (JSP) engine. The Java Servlet and JSP specifications are developed by Sun under the Java Community Process. Most Apache software development efforts that involve Java technologies fall under the Jakarta project. Tomcat is one of the better known technologies, but is definitely not the only one of these projects.

Samba

Samba is an open source toolkit that creates a bridge between Linux and Windows resources. Samba allows Windows clients to access Linux file systems and printers. It is implemented in such a way that Linux resources appear, to windows clients, just as if they were native Windows services.

This software allows Linux to effectively replace most Windows network services, such as print, file, and authentication. Most Linux distributions include Samba as part of the base file set.

Jetspeed (Web portal)

Jetspeed is another Apache Jakarta Open Source project. It is an open source implementation of Enterprise Information Portal, using Java and XML.

MySQL

MySQL is one of the most popular open source database engines. There are several including the equally capable PostGres SQL. We chose MySQL due to its ease of implementation, ease of integration, and apparent popularity.



Foundation

Building a house requires someone to dig a hole and pour concrete into it. This is just like putting IBM @server BladeCenter into a rack and providing power.

All construction projects require a firm foundation on which to build. The computer server is the foundation upon which we build applications that store and manage information. A strong and sound foundation is key for the longevity and survivability of any structure. Foundations made up of redundant structures can provide timeless support for massive buildings.

BladeCenter allows you to use relatively inexpensive components to create a highly robust and redundant foundation for any application.

3.1 Hardware

BladeCenter is a high-density, rack-mounted server system. The BladeCenter system provides shared resources to all the blades, such as power, cooling, system management, network connections, CD-ROM, floppy, keyboard, video, and mouse. The use of common resources allows the blades to be smaller and reduces the need for cabling.

BladeCenter consists of a rack-mounted chassis. The front of BladeCenter supports 14 blade server slots and has a CD-ROM drive, USB port, and a floppy drive. The back of the chassis has slots for two blower modules, four power modules, four network modules, and a management module.

3.1.1 Single CD-ROM, floppy drive, keyboard, video, and mouse

All the blades share the CD-ROM, floppy drive, keyboard, video and mouse. There are two I/O selection buttons on the front of each blade:

- ▶ Select the CD-ROM and floppy drive
- ▶ Select the keyboard, video, and mouse (KVM)

There is also a power button on each blade that is protected by a hinged plastic flap. After a blade is powered up, you can press the CD-ROM or the KVM button on that blade. On the blade that is currently connected to the CD-ROM or the KVM, the I/O selection button appears in solid green.

Sharing the CD-ROM for all the blades is a limitation to installing the operating systems on multiple blades. Using the CD-ROM, you can serially install operating systems. However, that process is very time consuming if you install more than only one or two blades. We recommend that you install one blade that you configure to be a network installed server. Subsequent operating system installations are then performed from that server. The following section explains how to do this.

3.2 Installing operating system instances

A challenge to installing and maintaining a manageable collection of servers is having a system for consistent and reproducible operating system installations. There are various strategies for producing consistent operating system installations. This redbook demonstrates how to install Red Hat Advanced Server 2.1 using the software package system provided by the Red Hat Linux distribution, the RedHat Package Management System (RPMS), and a feature of the Red Hat installer called Kickstart.

Before the Kickstart installation software loads, a Linux kernel and initial RAM disk, referred to as an *initrd*, must be loaded on the system. You can boot a Linux kernel and *initrd* from the floppy, CD-ROM, or network using the Intel® pre-execution environment (PXE). We recommend using PXE because it provides the most flexibility after it is setup. One PXE server can provide multiple configuration files for various operating systems and relieves the administrator from shuffling CD-ROMs or floppy disks.

We recommend that you install one blade from the Red Hat Advance Server 2.1 CD-ROMs. Then complete the instructions in the following section to configure that machine as a PXE boot and Kickstart installation server.

3.2.1 PXE

PXE is an Intel i386 BIOS technology that provides a mechanism to download and run a native x86 binary, from a network, before an operating system is booted.

The services that make up a PXE boot network install environment include:

- ▶ BOOTP (boot) server
- ▶ Trivial File Transfer Protocol (TFTP) server
- ▶ Network File System (NFS) server for the second stage of the install, Kickstart
- ▶ Domain name system (DNS), which is helpful but not mandatory

The chain of events in PXE boot is as follows:

1. The system BIOS uses the BOOTP protocol to download the pxeboot application, pxelinux.0.
2. pxelinux.0 is executed.
3. pxelinux.0 uses the TFTP protocol to download a configuration file that points to a Linux kernel and specifies kernel boot parameters.
4. pxelinux.0 then uses the TFTP protocol to download the kernel, and initrd loads them into RAM.
5. The kernel is executed with the boot flags specified in the PXE configuration file.

For PXE to function, you must configure a server with BOOTP, TFTP, NFS, and optionally DNS.

Configuring the BOOTP server

The ISC DHCPD server that comes with all Red Hat distributions also provides the BOOTP protocol. In the DHCPD configuration file, add the following lines to the subnet block that serves the systems you want to boot via PXE boot:

```
allow booting;
allow bootp;
```

Then add a block as shown in the following example for the PXE boot systems:

```
group {
    # PXE-specific configuration directives...
    filename "pxelinux.0";
    host system_name {
        hardware ethernet AA:BB:00:11:22:33:44:55;
        fixed-address blade7.bce.ibm.com;
    }
}
```

In this example, assume that you have a working domain name system and have created the A record and the reverse PTR for blade7.bce.ibm.com. If you don't already have a working domain name system on the network, you can enter the Internet Protocol (IP) address on the fixed-address line.

You should also know the MAC address of the system you want to boot. The MAC addresses for the two Ethernet interfaces of a BladeCenter blade are printed on the sheet metal case. You need to pull the blade nearly all the way out of the chassis to read the stickers with the MAC addresses, which are on the left-hand side (as you are facing the front of the blade). The MAC addresses are always one digit apart and are represented in hexadecimal. The lower number MAC address is associated with the network module in slot 2 on the back of the BladeCenter chassis. In Linux, it comes up as */dev/eth0*.

Configuring the TFTP server

Make sure your TFTP server is installed and working. To configure the server, follow these steps:

1. Generally TFTPd is run from inetd. Check the `/etc/xinetd.d` directory to confirm that TFTPd is enabled.
2. The default TFTP server download directory is `/tftpboot`. Copy the following files to the `/tftpboot` directory:
 - **pxelinux.0**: From the `syslinux` package
 - **vmlinux**: The Linux kernel from the `/images/pxeboot` directory on the first CD
 - **initrd-everything.img**: Also from the `/images/pxeboot` directory on the first CD
3. Make a subdirectory of `/tftpboot` called `pxelinux.cfg`.
4. In the `pxelinux.cfg` directory, create a `pxelinux.0` configuration file.
5. Verify the configuration file. Download `pxelinux.cfg` via TFTP from a known working machine. You should see the following line in the file:

```
lspl.ibm.com# tftp 10.0.0.10 pxelinux.cfg/pxelinux.0
```

Verifying that TFTP is working at this point can save you from a frustrating debugging task later.

After this is set up, `pxelinux.0` searches for its configuration file on the TFTP server:

1. It searches for the configuration file using its own IP address in uppercase hexadecimal. For example, `10.0.0.17` is `0A000011` in hexadecimal. Use the program included in the `syslinux` package called *gethostip* to compute the hexadecimal IP address for any host.
2. If that file is not found, it removes one hex digit and tries again. Ultimately, it tries to look for a file named *default*. For example, for `10.0.0.17`, `pxelinux.0` tries to fetch the file `0A000011`, `0A00001`, `0A0000`, `0A000`, `0A00`, `0A0`, `0A`, `0`, and finally a file named `default`, in that order.

The `pxelinux.0` configuration file should look like the following example:

```
default linux
serial 0,38400n8
  label linux
    kernel vmlinux-as2.1
    append load_ramdisk=1 initrd=initrd.img-as2.1
ks=nfs:10.0.0.10:/home/export/as2.1-qu2/ks.cfg
```

The kernel name and the `initrd` name should be the same as the files you copied into the `/tftpboot` directory earlier. You can call them anything you want as long as the names are consistent. You may want to include a reference to the distribution that they came from in the name.

PXE performs the first stage of installation to load of a special Linux boot kernel and initial RAM disk. It also passes, to the kernel boot parameters that specify the method, to retrieve and the location of the Kickstart configuration file. Now you are ready to continue to the second stage of the network installation, the Kickstart stage.

3.2.2 Red Hat Kickstart

Red Hat Kickstart installation is a system for automating a network or CD-ROM installation of the Red Hat Linux operating system. Kickstart is a feature implemented by the Red Hat installation program called *Anaconda*. *Anaconda* reads the Kickstart configuration files that supply all the information necessary to complete the installation. For example, it reads the

path to the packages, installation method, and disk partitions. A complete Kickstart file allows the operating system to be installed without any interaction from the operator at the console.

Kickstart requirements for BladeCenter

We recommend that you use the Dynamic Host Configuration Protocol (DHCP) and NFS protocol to perform network Kickstart installations on BladeCenter.

On the blade that was installed from the CD-ROM, confirm that the *nfs-util* and the DHCPD packages are installed.

Planning your network and setting up a DHCP server

Perform the following steps to plan your network and set up a DHCP server:

1. Determine which network address space you are going to use for your blades management network. This example uses the network 10.0.0.0/24.
2. Set up an interface on the first blade in that network.
3. Create an `/etc/dhcpd.conf` file, as shown in the following example, to serve a range of addresses in that network:

```
# This is a basic dhcpd.conf file
subnet 10.0.0.0 netmask 255.255.255.0 {
    authoritative;
    allow booting;
    allow bootp;
    option routers 10.0.0.1;
    range 10.0.0.50 10.0.0.100;
    default-lease-time 600;
    max-lease-time 7200;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 10.0.0.19;
    option domain-name "bce.ibm.com";
```

4. Confirm that there is a symbolic link from `/etc/rc3.d/S90dhcpd` to `/etc/rc.d/init.d/dhcpd`.
5. Start the DHCP server. Type the following command:

```
/etc/rc3.d/S90dhcpd start
```

Exporting the Red Hat distribution via NFS

To export the Red Hat distribution, follow these steps:

1. On the installed blade, mount the first CD-ROM of the Red Hat Advanced Server 2.1 distribution. Type the following command:

```
mount cdrom
```

This command should mount the device in the default location `/mnt/cdrom`.

2. Create a directory to export via NFS to the network, for example, `/home/export/as2.1-qu2`. Change the directory to `/mnt/cdrom` and issue the command:

```
tar cf - . | (cd /home/export/as2.1-qu2; tar xf -)
```

3. Change the directories to `/mnt`.
4. Unmount the CD-ROM. Type the following command:

```
umount cdrom
```

5. Repeat steps 3 and 4 for CD-ROMs 2 and 3. You don't need CD-ROM 4.

6. Add the following line to the `/etc/exports` file:

```
/home/export/as2.1-qu2 *(ro)
```

7. Create a symbolic link from /etc/rc3.d to /etc/rc.d/init.d/nfs and /etc/rc.d/init.d/nfslock.
8. Start the NFS daemon and nfslock daemon.
9. Place a Kickstart configuration file, such as the ks.cfg file in the following section, in the /home/export/as2.1-qu2 directory.

Your Kickstart installation system is ready to go.

3.2.3 Sample Kickstart configuration for BladeCenter

The following example shows a Kickstart configuration file that successfully installs the Red Hat Advance Server 2.1 Quarterly Update 2 on a BladeCenter blade:

```
# Sample Kickstart file to install Red Hat Advanced Server 2.1 Quarterly Update 2
# on a BladeCenter

install
text
lang en_US
langsupport --default en_US en_US
keyboard us
mouse none
skipx
network --bootproto dhcp
rootpw --iscrypted $1$T.ÉynáFG$op3zk2u1ZSdpWT2/M9Fhv/
firewall --disabled
authconfig --enablshadow
timezone America/Los_Angeles
bootloader --location=mbr
nfs --server blade1.bce.ibm.com --dir /home/export/as2.1-qu2
# Clear the disks and create new partitions and filesystems
clearpart --all --initlabel
part /boot --fstype ext2 --size=50 --ondisk=hda
part /usr --fstype ext2 --size=4096 --ondisk=hda
part swap --size=1000 --maxsize=2048 --ondisk=hda
part /home --fstype ext2 --size=4096 --ondisk=hda
part / --fstype ext2 --size=2000 --ondisk=hda
part /var --fstype ext2 --size=4096 --ondisk=hda
# Specify the packages
%packages --resolvedeps
@ Network Support
@ Classic X Window System
@ X Window System
@ GNOME
@ Messaging and Web Tools
@ NFS File Server
@ Windows File Server
@ Anonymous FTP Server
@ Web Server
@ Router / Firewall
@ DNS Name Server
@ Network Managed Workstation
@ Utilities
@ Software Development
@ Advanced Server
compat-libstdc++
shapecfg
ddd
IBMJava2-SDK
libpcap
```

```
freetype-devel
doxygen
apache-devel
mysqlclient9
nmap-frontend
mozilla-psm
mozilla-nspr
mysql-server
openldap-servers
libxml-devel
openldap12
htmlview
php-ldap
libmng-devel
netscape-communicator
memprof
bindconf
mozilla-nss
auth_ldap
glib-devel
mozilla
mysql
apacheconf
openssh-askpass
bind-devel
usbview
netscape-common
dhcp
tftp-server
# Post Installation Scripts
%post --nochroot
# Turn off unwanted daemon processes
rm /mnt/sysimage/etc/rc3.d/S[0-9][0-9] kudzu
rm /mnt/sysimage/etc/rc3.d/S[0-9][0-9] apmd
rm /mnt/sysimage/etc/rc3.d/S[0-9][0-9] gpm
rm /mnt/sysimage/etc/rc3.d/S[0-9][0-9] anacron
rm /mnt/sysimage/etc/rc3.d/S[0-9][0-9] sendmail
rm /mnt/sysimage/etc/rc3.d/S[0-9][0-9] lpd
rm /mnt/sysimage/etc/rc3.d/S[0-9][0-9] isdn
```




Plumbing: Network infrastructure

Often before the foundation is laid, trenching and piping for the plumbing are laid. In many instances, the T1 line is ordered before the servers are purchased.

The network is the plumbing of modern companies. It seems sometimes like the standards on which the Internet relies are as strong a force as gravity. In copper plumbing, we deal with gravity through pumps, valves, and other contraptions. In the Internet world, we have Dynamic Host Configuration Protocol (DHCP), domain name server (DNS), and Lightweight Directory Access Protocol (LDAP).

4.1 DHCP

DHCP is now very popular for managing IP address assignments. It is a standard in many corporate environments. It's especially popular with network administrators who have been maintaining static name service tables.

DHCP was first widely used for Windows desktops. Today Windows, Linux (and most UNIX), mobile and wireless clients all talk to DHCP. With this widespread use, security and availability become major concerns.

When we originally wrote this section, DHCP Version 3.0 was still in beta. At time of publication, the stable production version was 3.0p2, and 3.01rc11 was the latest release candidate.

4.1.1 Background

DHCP (RFC 1531) provides a method for passing network configuration information to hosts on a Transmission Control Protocol/Internet Protocol (TCP/IP) network. DHCP descended from the BOOTP protocol (RFC 951) used to boot diskless workstation over a TCP/IP network.

DHCP is based on a client-server model. The client broadcasts a request for network configuration information. The server assigns an IP address and transmits that address plus other network configuration information to the client. Finding the client in this instance is by means of the hardware MAC address. The network administrator assigns the address range controlled by the server. They establish any other information that is needed by a new client (hostname, default routes, etc) and never touch the new client box again.

Does this make grumpy administrators happy administrators? Yes, at least until the server shuts down at 5:00 in the morning.

4.1.2 Building in fault tolerance

Version 3 of the ISC DHCP server supports the DHCP failover protocol. The failover protocol allows two DHCP servers to share the same IP address pool or pools. The failover protocol defines a primary server role and a secondary server role. There are minor differences in how the primary and secondary servers work, but the differences in configuration are minimal.

The address allocation algorithm is part of the DHCP internal cluster code. Both servers take turns answering DHCP requests. They give out addresses from their respective address pools based on a hash of the client ID, unless a failure of the other server is detected.

For the nodes to keep track of the health status of their partner node, packets are sent back and forth on a private port. There are two modes of failure detection. If a node fails to respond to a predetermined number of failover status checks by a partner node, the node is deemed *dead*. The remaining functioning server goes into partner down state and takes all DHCP requests until the failed server is reactivated.

The other failure detection mode is when a server is responding to failure status requests on the private port but is unable to answer DHCP requests for clients. Since both nodes are always listening, the still functioning server responds to a client request out of turn after the initial client requests fails to be answered by the partner. This out of turn response happens after the first client request fails, but before the request times out. This insures that almost any DHCP server failure will be transparent to the clients.

After one of the servers fails, the remaining server continues to renew and distributed leases out of the address pool. During a prolonged outage (such as waiting for parts), the remaining server can reclaim all the addresses from the other server's part of the pool and reuse them.

The *omshell* is an interactive shell that allows you to query and change the partner failure state of a node. This change in failure state allows you to tell the remaining node to reclaim the failed nodes addresses and reuse them. When the failed server is put back online, it automatically detects that it has been offline and requests a complete update from its partner server. After the update completes, both servers resume normal operations.

Example 4-1 shows the sample configuration files for a simple two-node configuration.

Example 4-1 Configuration files for a simple two-node configuration

```
##### /etc/dhcpd.conf #####
Failover peer "192.168.10.11" {          # the other node of the cluster
    primary;                            # am I primary or secondary ?
    address 192.168.10.12;               # My address
    port 518;                            # Unused port to talk to the other node on
    peer address 192.168.10.11;         # My peer
    peer port 521;                       # port to talk to my peer on
    max-response-delay 30;              # number of seconds in which a response is not received from
the other node that it is assumed down.
    max-unacked-updates 10;            # Maximum un-acknowledged updates before other node is
considered down
    load balance max seconds 3;        # number of seconds before load balancing is bypassed
    mclt 3600;                          # maximum client lead time in this relationship
    hba ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff: # split between primary and
secondary, on
    00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00; # only needed in the primary config file
}

    lease-file-name "/etc/dhcpd.leases"; # lease database
    one-lease-per-client on;            # One lease per node
    ddns-update-style interim;         # dynamic DNS update type
    include "/etc/dhcpd.master";      # include file for pool information
##### end dhcpd.conf #####
##### start /etc/dhcpd.master #####
subnet 192.168.10.0 netmask 255.255.255.0 {
    pool{
        range 192.168.10.100 192.168.10.120;
        option subnet-mask 255.255.255.0;
        option broadcast-address 192.168.10.255;
        option domain-name-servers 192.168.10.12;
        option domain-name "foo.com";
        authoritative;
        default-lease-time 86400;
        max-lease-time 172800;
        min-lease-time 86400;
        failover peer "192.168.10.12";
        deny dynamic bootp clients;
    }
}
##### end /etc/dhcpd.master #####
```

4.1.3 Security concerns

Almost all networks need to implement some type of security. One way to enhance security with DHCP is the use of the "host" option in the DHCP configuration files. This host option allows you to specify an address or address pool for a particular host based on MAC and the hardware type. These host identifier options also allow you to deny giving systems without an entry an address or giving them an address on a non-secure network. The administrative overhead is much greater with this method since all hosts MAC and hardware type must be entered in the configuration file, but it is one way to add to the security of a wireless DHCP network segment.

Some larger DHCP implementations have developed a custom topology where when a user first logs in. The system they are on is given an address in an untrusted network. The user then logs into a system with a special password. After they are authenticated, the system logs the MAC address and adds a special host entry for that user in the trusted network DHCP configuration file. The next time the user boots up onto the network, they are given an address on the trusted network without any manual intervention. You can even set your DNS server up to handle DNS updates for you.

Although the standard for secure dynamic DNS updates is still being debated, version 3 offers a couple of different methods for making secure updates. For those of you who are supporting systems that require dynamic updates, such as Windows Active Directory, both methods offer a significant security enhancement over previous versions. Instead of leaving your DNS vulnerable or adding a DNS update key to each client, the DHCP server can be configured to send secure dynamic updates. Both A and PTR records can be securely updated on your DNS server using a secure key. After the DHCP server gives a host an address, the DHCP server sends an update to the DNS server. When the address is reclaimed by the DHCP server, an update is sent to the DNS server deleting the record. For this to work, both the DNS and DHCP servers must be set up with a secure zone.

The interim method is the only one you can use with failover. It is the closest to the standard. Your setup in BIND should look similar to Example 4-2.

Example 4-2 BIND setup file for security

```
Key dhcp_key {
Algorithm hmac-md5;
Secret shhh ;
};
zone " foo.com" {
    type master;
    file "/var/named/foo.com.zone";
    allow-update {key dhcp_key};
};
zone "10.168.192.in-addr.arpa" {
    type master;
    file "/var/named/10.168.192";
    allow-update {key dhcp_key};
};
```

In this example, we define a key called `dhcp_key` and use the `hmac-md5` algorithm with secret `"shhh"` (replace this with a real secret key). The zone `foo.com` allows dynamic updates to both A (zone `foo.com`) and PTR records (zone `10.168.192.in-addr.arpa`), but only by a host with the secret key defined by `dhcp_key`. You also need a corresponding entry in your

dhcpd.conf file to tell your DHCP server how to update the records. Example 4-3 shows the corresponding DHCP entry for the DNS example.

Example 4-3 File for DHCP entry

```
key dhcp_key {
algorithm hmac-md5;
secret shhh ;
};
zone " foo.com." {
primary 127.0.0.1;
key dhcp_key;
};
zone "10.168.192.in-addr.arpa." {
primary 127.0.0.1;
key dhcp_key;
};
```

These are similar to the statements we put into the BIND configuration file, with a few exceptions. The zones are configured with 127.0.0.1 as a primary address and must correspond to authority records on your DNS server.

Also note the "." appended to the end of the zone declaration and the lack of quotes on the zone names. These are minor differences in the way DHCP and BIND parse data.

There is a potential problem with allowing the DNS server to handle the updates. When a dynamic update is sent the DNS server, the DNS server does not immediately update the zone file. If for some reason the DNS server goes down before the record update is written from cache to the zone file, the update is lost. After a DHCP server makes an update, it does not try to make another update for that host until the lease is broken. In this case, you must manually edit the leases file or manually update the DNS data files.

Not all these features have been fully developed or tested, so your experience may vary until these features become more mature.

We should also note that 3.0p2 has a buffer overflow vulnerability. Hopefully 3.01 will be the current stable release by the time you read this book.

4.1.4 Conclusion

If you are currently using a DHCP server from another vendor, or using an older version of the ISC DHCP service, DHCP Version 3 is well worth the effort. Although the DNS update security standard has yet to be fully decided, the added security available in this release is very good. Along with the new the security features, the benefits of having redundant DHCP servers in itself are worth the installation hassles.

Even the best-built server eventually fails, so plan for it in advance. Some time in the future you will be glad you did.

4.2 DNS

One of the most critical components of most networks is the DNS server. Have you ever been unable to get to a crucial network resource and , had long waits for DNS service? A failed

network service infrastructure means lost revenue productivity and may put you into a remediation situation with your customers.

The following section references a two-node cluster implementation that allows DNS to be implemented in a Linux clustered environment that fails over crucial services.

Most of us have limited hardware budgets, which limits our spare parts supply. How long would it take you to get a new motherboard or hard drive for your primary DNS server? A hardware failure can render a server disabled for days or even weeks.

Relying on secondary DNS servers may work for the short term, but not for more than a couple of hours. That could spell trouble. Some systems can take as long as seventy-five seconds before the DNS query times out and is sent to the secondary DNS server. Slow resolver queries trigger customer calls to the help desk complaining of a broken or slow network. This, in turn, can lead to service level agreement (SLA) violation claims, and customers demanding reimbursements for loss of service.

Another problem that can come up while the primary server is down is that no updates can be done and changes must wait until the primary is back online. Even the most robust networks fall to their knees without a functioning DNS server. Make the primary DNS server for your domain highly available. It's easy, painless, and fairly inexpensive.

4.2.1 History

DNS handles the translation of computer network numbers to names. Most of us find that beer.foo.com is easier to remember than 192.168.22.134. DNS also keeps other information about networks, such as mail server names, in a standard format that other systems may query.

The DNS implementation we discuss here is Berkeley Internet Name Domain (BIND). BIND was first developed by University of California at Berkley under a grant by U.S. Defense Advanced Research Projects Administration (DARPA). BIND includes a name server daemon called *named*, a resolver library, and a set of tools to verify the operation of the server. Currently BIND is being maintained and developed by the Internet Software Consortium and is available for free download from their Web site, which is available at:

<http://www.isc.org/>

Most Linux operating system vendors include a pre-built version of BIND in their distributions. BIND is licensed under the GNU Public License (GPL), and it is also open source.

To make a highly available primary DNS server, two servers are needed. One of the servers is the primary node and one server is the backup node. The *heartbeat* daemon is used to track the network availability of the primary node by the backup node. Any current version of BIND should work for the DNS server. We recommend that you use the version of BIND that ships with your Linux distribution. To handle the server failover, use the Heartbeat package available from the Web at:

<http://www.linux-ha.org>

Only one node at a time is an active primary DNS server. The other node is in the "bull pen" getting ready to take over in case of a failure of the primary node. After the secondary node detects that the primary node has failed, the backup node takes over the primary node's IP address and starts its services. Unfortunately BIND, like many services, does not have a native function to replicate data between primary servers. Therefore, replication must be done using another tool. We recommend that you use the *rsync utility*.

Note: Keep in mind that in the event of a failover, you must replicate any updates to the secondary DNS back to the primary DNS, after the primary DNS comes back online.

Figure 4-1 illustrates a normal operation of the DNS.

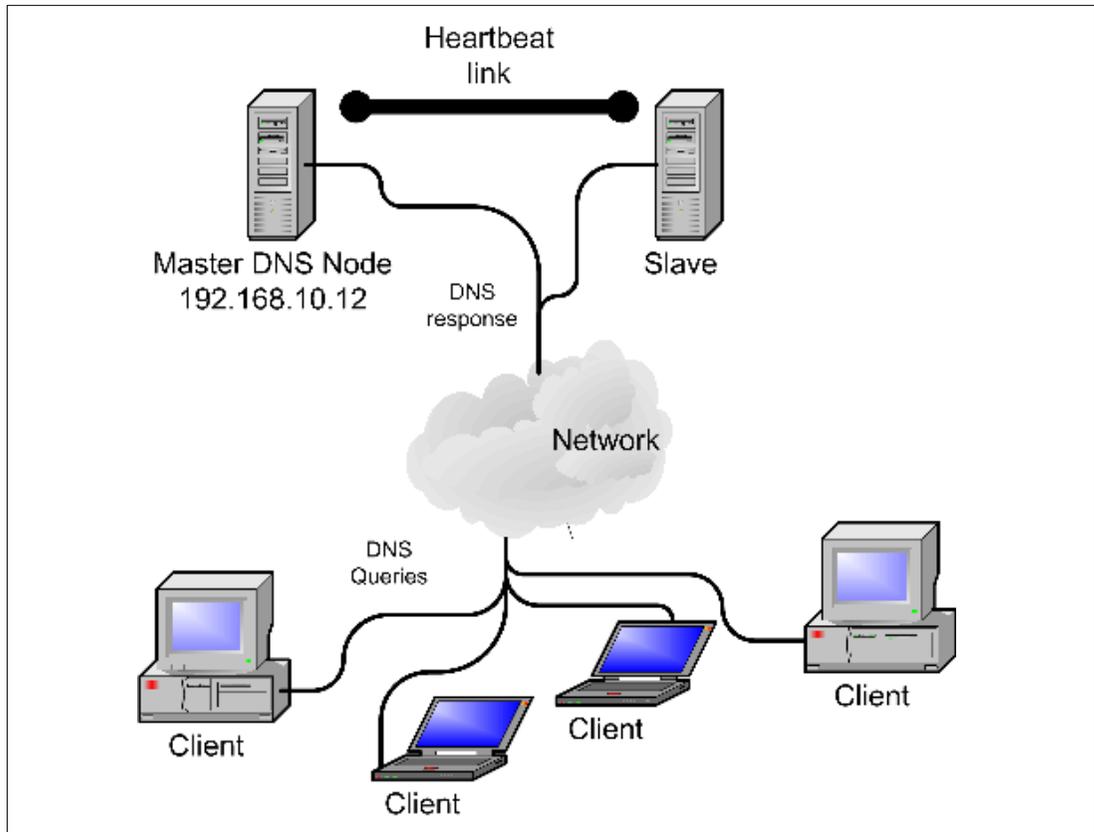


Figure 4-1 DNS normal operation

After the secondary node detects that the primary node is no longer on the network, the secondary node takes over the primary node's IP address. It starts DNS services and effectively becomes the primary DNS server.

Figure 4-2 illustrates a failed operation of the DNS. The biggest problem with this type of setup is keeping both servers' databases synchronized with any changes. BIND does not have any native functions to replicate data files between two primary servers. To keep the two databases synchronized, you must employ a shell script or custom utility that uses something like the `rsync` utility to synchronize the data between the primary and backup node. The `rsync` utility allows you only to update changed files in a particular tree. By just updating the changed files in the tree, you can minimize the network traffic and system overhead.

We set up a simple `rsync` command to run every ten minutes in a cron job to commit any updates to the secondary node. If the primary node dies, all updates since the last `rsync` are lost. Hopefully since only ten minutes of updates are lost, it is a minimal amount of work to recover. You can immediately update any updates by manually running `rsync` or with a small script. To further ensure data integrity, you must replicate any updates made to the secondary server, while it is acting as a primary server, back to the primary server before you bring it online as the primary server.

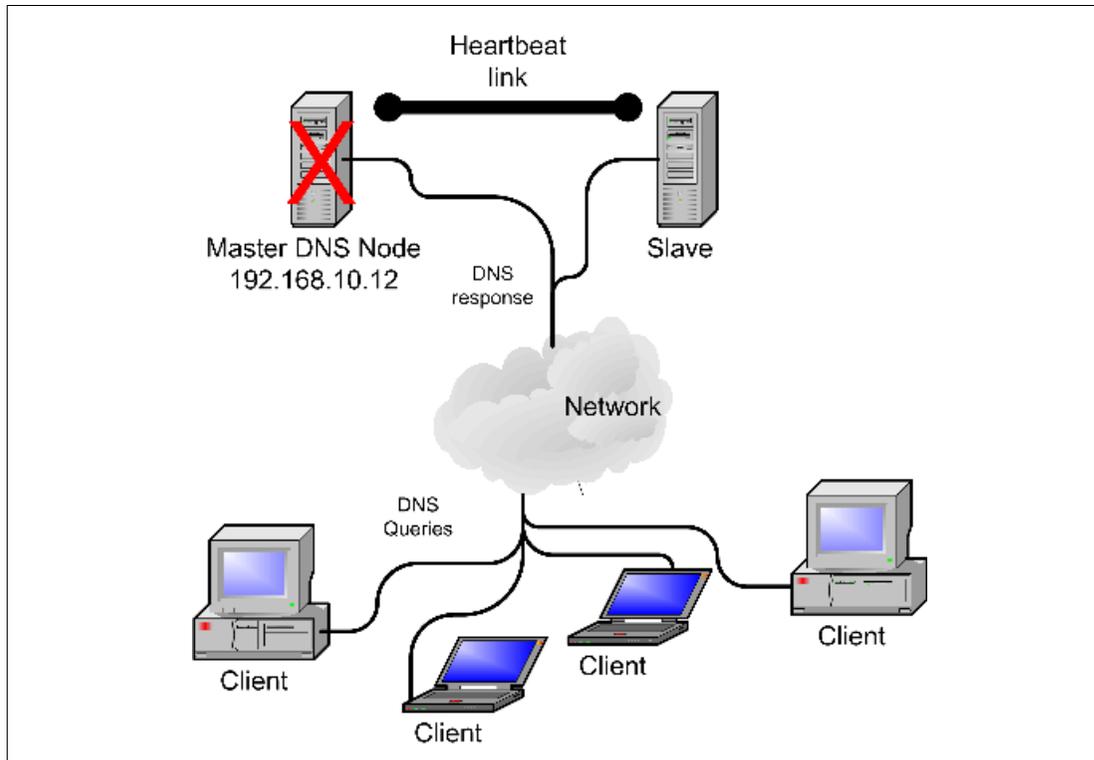


Figure 4-2 Failed master DNS server

4.2.2 Building a highly available DNS

This section discusses a server and cluster configuration for a highly available DNS.

The server configuration

To properly configure highly available DNS, you need two servers and at least two network ports on each. One network port is used for the heartbeat daemon's private port, while the other port is used to answer queries on the client network. You can download the latest version of BIND 9.1.3 from the Web at:

<http://www.isc.org>

Building the BIND server went off without any problems. BIND uses the standard GNU **configure** and **make** scripts. This means we expand the package and use the standard GNU build recipe. The same is true with Heartbeat, but you need to install the heartbeat patch included here before you build it:

1. Install the patch by copying to a file called `hbpatch`.
2. Copy the patch to the heartbeat `src` directory.
3. Run the following commands:

```
" patch -p0 < ./hbpatch
# cd <your_source_dir>
# ./configure
# make
```

4. After the build succeeds, run the following command to put everything in its place:

```
#make install
```

Example 4-4 shows the heartbeat patch.

Example 4-4 Heartbeat patch

```
#####  
--- heartbeat-0.4.9/stonith/expect.cTue Nov 21 17:56:46 2000  
+++ heartbeat/stonith/expect.cFri Jul6 15:06:34 2001  
@@ -36,7 +36,7 @@  
#include <unistd.h>  
#include <signal.h>  
#include <errno.h>  
-#include <sys/time.h>  
+#include <time.h>  
#include <sys/times.h>  
#ifdef _POSIX_PRIORITY_SCHEDULING  
# include <sched.h>  
--- heartbeat-0.4.9/heartbeat/hb_api.cSat Nov 11 12:05:50 2000  
+++ heartbeat/heartbeat/hb_api.cFri Jul6 15:08:46 2001  
@@ -74,7 +74,7 @@  
#include <hb_api.h>  
#include <hb_api_core.h>  
#include <sys/stat.h>  
-#include <sys/time.h>  
+#include <time.h>  
#include <unistd.h>  
static int api_ping_iflist(const struct ha_msg* msg, struct node_info * node  
#####
```

Cluster configuration

After you install heartbeat and BIND, there are three main files to edit to configure heartbeat for failover. “ha.cf” is the main cluster configuration that defines who’s in a cluster and how they communicate. The “haresources” file defines what IP address to use and what services to start upon a detected failure of the primary node. The third file is “authkeys” that is used for security keys. The following sample configurations are for ha.cf and haresources, which were used in testing this solution. We do not cover BIND setup for here for obvious reasons. You must also set up the permissions for rsync to work, but that can vary from the different distributions.

In the /etc/ha.d/haresources file (Example 4-5), you must define the failover actions to be performed in the event of a failure of the peer node. In this case, beer1.foo.com starts IP address 192.168.10.12 and starts the service named in case its peer node beer.foo.com fails.

Example 4-5 /etc/ha.d/haresources file

```
#### /etc/ha.d/haresources #####  
beer1.foo.com 192.168.10.12 named# server name to take over on failure, ip address of node  
, service to start.  
#### End of haresources#####
```

The next file that you must edit is the ha.cf file (Example 4-6). The ha.cf is the configuration file for the cluster server. You can define how much time it takes to declare a host dead after it

stops communicating. You also need to define the heartbeat hardware to use and the node names of the cluster.

Example 4-6 ha.cf file

```
##### /etc/ha/d/ha.cf #####
debugfile /var/log/ha-debug # Where to put debug output
logfile /var/log/ha-log # Where to put the log
logfacility local0 # Facility to use for syslog()/logger
keepalive 2 # keepalive: how many seconds between heartbeats
deadtime 10 # deadtime: seconds-to-declare-host-dead
udpport 694 # udpport What UDP port to use for communication?
udp eth1 # What interfaces to heartbeat over?
node beer1.foo.com # cluster members
node beer2.foo.com
##### End of configuration#####
```

After you configure heartbeat, you must copy the startup script for the named service and heartbeat into /etc/init . Do not place the scripts into the /etc/rc directories because you do not want these services to start automatically. If you have a failure, you must synchronize the data before failing back from the backup server to the primary server. With this type of configuration, you have to start heartbeat and BIND manually, but that's a small price to pay for a reliable DNS.

4.2.3 Conclusion

Although initially it is more work to set up a cluster than a single server, you will find the effort well worth it. With the lower costs of hardware and the use of free software, this solution is very cost effective when compared to commercial solutions. Plan ahead now and have one less emergency to deal with in the future.

4.3 LDAP

LDAP is a protocol for providing directory information. A directory is like the phonebook you would find in most households. It is a specialized database for information about people, and maybe places and things. LDAP is considered lightweight because it is the "little" cousin of the X.500 DAP protocol and the very heavyweight Resource Access Control Facility (RACF).

The concept of a directory as an information repository is a simple yet powerful one. We use it in everyday life. Consider the example of the ubiquitous telephone book (or any of its Web analogs), which lists the telephone numbers and addresses of people and business establishments. It's a simple creation, but one that is an important part of everyday life. Beyond public directories, we also create our own directories, whether implicitly or explicitly, such as e-mail address lists, keychains, or emergency numbers.

Directories can be similarly extended into IT. An organization can use a central directory to provide common authentication for its computer systems, a listing of available servers and printers, and a list of end-user mailboxes. In each of the major applications we examine, we look at how to integrate them with directory services.

4.3.1 LDAP servers

As the name suggests, LDAP is a network protocol that is used to access directories. However, common usage has evolved to refer to LDAP as a directory system in itself.

LDAP directories store information, similar to a database, but they contain more descriptive attribute-based data. The data held in LDAP directories is optimized for reading, so frequently changing data, such as transactions or e-mail messages, does not fit well in the model. Information in LDAP directories is arranged in a hierarchical structure. This allows you to separate data based on different criteria.

An LDAP directory is essentially a database, so what separates it from other databases? Aside from being oriented toward read operations, an LDAP directory contains data in a very well-defined structure. This is something that can, therefore, be easily accessed and referenced by several applications that are written to access LDAP.

For some time now, IT vendors have pushed their own flavor of the LDAP directory. Microsoft uses the Active Directory, Novell has Netware Directory Services, Sun has iPlanet Directory, and IBM has the IBM Directory Server. These directory server flavors provide the same basic functionality, although they may be optimized to work with other products from the same suite offered by the vendor.

The LDAP version we use throughout this redbook is the *OpenLDAP server*. OpenLDAP is one of two free open-source LDAP servers available today. The other is the University of Michigan LDAP server. Of the two, OpenLDAP is more commonly used and is included in most Linux distributions.

This performance and capacity is adequate for the majority of organizations. At some point, OpenLDAP will become a bottleneck. If OpenLDAP is overloaded, it does not necessarily fail. It exhibits two performance related behaviors:

- ▶ LDAP-enabled authentication becomes slow.
- ▶ The CPU and I/O utilization of the LDAP process, slapd (more on this daemon later), capitalizes the system on which it is running.

When these events occur, consider moving to a commercial LDAP server. Unlike most software migrations, migrating LDAP directory servers is quite simple. This is in no small part due to the prevalence of the open LDAP standards. When you implement a directory using OpenLDAP, your directory supports the LDAP specifications listed in Internet Engineering Task Force (IETF) RFCs 2829, 2830, 2840, and many others. These same standards are at the core of both OpenLDAP and commercial LDAP servers, such as IBM Directory server.

4.3.2 LDAP concepts

LDAP directories consist of entries or objects arranged in a hierarchy that can be used to match the structure of an organization. The entries can represent organizations, organization units, people, roles, company assets, and almost any type of entity. Figure 4-3 shows an example of an LDAP directory hierarchy.

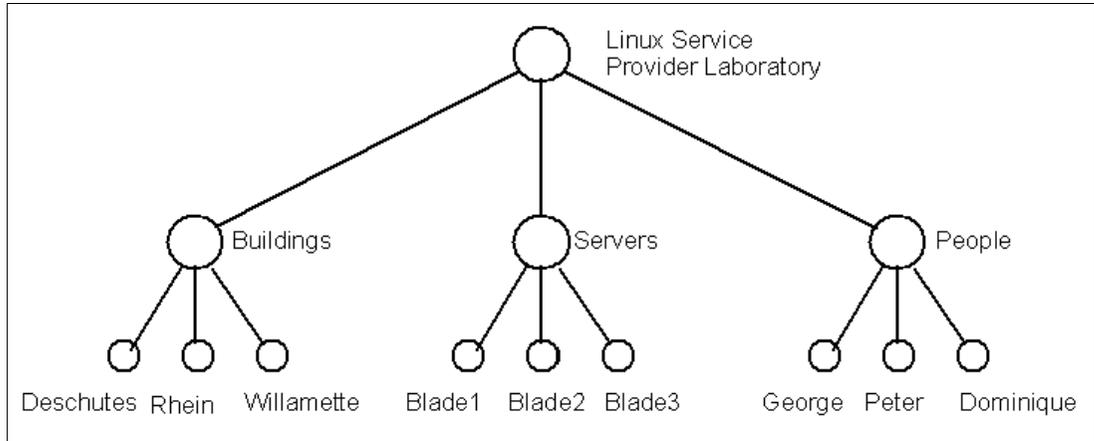


Figure 4-3 A simple LDAP directory hierarchy

LDAP objects

The most basic element of the LDAP information model is an object. An *entry* is a collection of attributes that has a globally unique Distinguished Name (DN). Example 4-7 shows an LDAP entry.

Example 4-7 An LDAP entry

```

dn: uid=dominique,ou=users,o=lspl.ibm.com
uid: dominique
cn: Dominique Cimafranca
givenname: Dominique
sn: Cimafranca
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
mail: dominique@lspl.ibm.com
  
```

The DN, in this case `uid=dominique,ou=users,o=lspl.ibm.com`, refers to the entry unambiguously. There can be no other entry in the LDAP directory with the same DN.

The attributes that we defined for this entry are:

- ▶ **uid:** A user ID for a user
- ▶ **cn:** Common name for a user
- ▶ **givenname:** The given name for a user
- ▶ **sn:** The surname for a user
- ▶ **mail:** E-mail address of a user

This is a basic example of an LDAP object that represents a user in an organization. In actual practice, this LDAP entry may contain more information such as telephone number, office location, mailing address, or job role.

Attributes, object classes, and schemas

The attributes that go into an object are not something that you can arbitrarily choose. They are part of the object classes that define the object and may be required or optional depending on the object class.

In our example, our entry is of object class `top`, `person`, `organizationalPerson`, and `inetOrgPerson`:

- ▶ The *top object class* is a required abstract object class required for any LDAP entry.
- ▶ The *person object class* is a structural object class. LDAP entries must have one structural object class. As the name implies, the person object class defines a person. The person object class requires the `cn` and `sn` attributes be defined in the entry.
- ▶ The *organizationalPerson object class* is a general purpose object class that holds attributes about people. It defines basic attributes that describe a person.
- ▶ The *inetOrgPerson object class* is also a general purpose object class that holds attributes about people. It defines attributes used in typical Internet and Intranet directory service deployments.

Several other predefined object classes are available. You may also choose to define your own object classes as you need them for your applications and organizations, but they must all be defined in the LDAP schema.

A schema defines valid object classes, the mandatory and optional attributes they contain, and rules on how data held by an object may be held.

LDAP hierarchy

As we mentioned before, LDAP directory entries are arranged in a hierarchical tree-like structure. Our example LDAP entry can be positioned along this tree, which is, in turn, part of the base (see Figure 4-4).

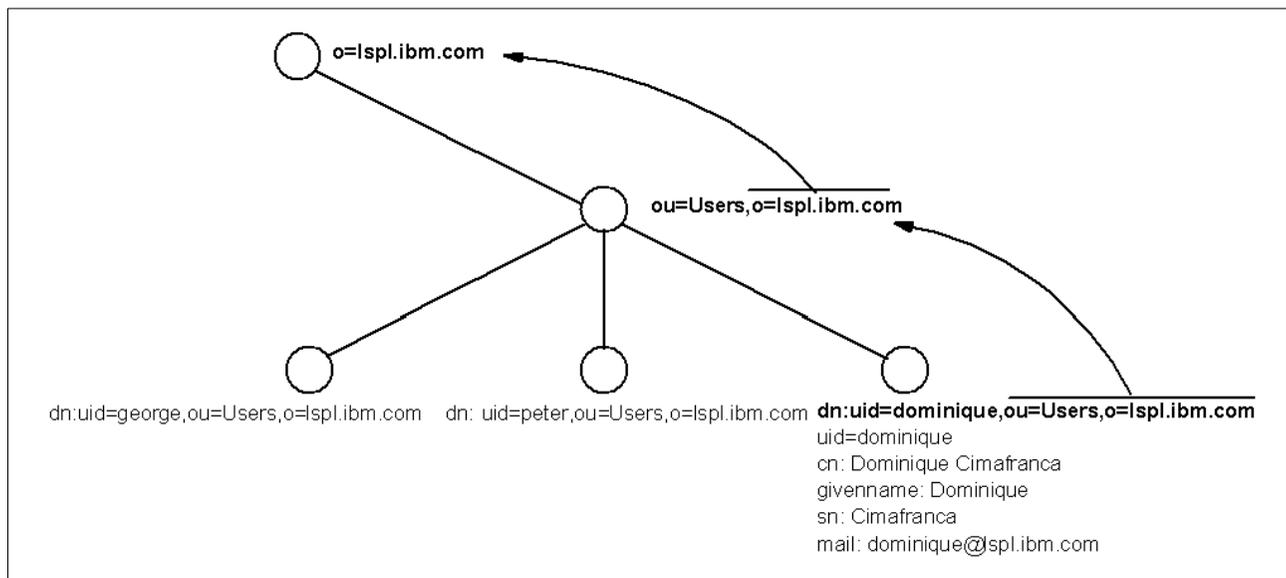


Figure 4-4 Visualizing our LDAP object in its tree

Notice that we have to define the entries for the organization and organizational unit that our users will be part of. Otherwise, the LDAP directory is invalid, and the LDAP directory server will not take them in.

Our example in LDAP Data Interchange Format (LDIF)

Lets look at a common form for expressing LDAP entries, the LDIF. LDIF is typically used to import and export directory information between LDAP-based directory servers or to describe a set of changes that are to be applied to a directory.

An LDIF file consists of a series of records separated by line separators. A record consists of lines that describe a directory entry, or lines describing a set of changes to be made to a directory entry.

Example 4-8 shows you an LDIF file that is used to initially populate an LDAP directory. This example is the LDIF representation of our LDAP hierarchy.

Example 4-8 LDIF file example.ldif

```
dn: o=lspl.ibm.com
o: lspl.ibm.com
objectclass: top
objectclass: organization

dn: ou=users, o=lspl.ibm.com
ou: users
objectclass: top
objectclass: organizationalUnit

dn: uid=dominique,ou=users,o=lspl.ibm.com
uid: dominique
cn: Dominique Cimafranca
givenname: Dominique
sn: Cimafranca
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
mail: dominique@lspl.ibm.com

dn: uid=george,ou=users,o=lspl.ibm.com
uid: george
cn: George Dolbier
givenname: George
sn: Dolbier
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
mail: george@lspl.ibm.com

dn: uid=peter,ou=users,o=lspl.ibm.com
uid: peter
cn: Peter Bogdanovic
givenname: Peter
sn: Bogdanovic
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
mail: peter@lspl.ibm.com
```

There are business situations where you may want to integrate your directory with an external one. If you find yourself in this situation, you may want to consider a commercial directory integrator. Commercial directory integrators allow real-time integration of multiple, independent, directories. We strongly recommend that you use a product such as IBM's Directory Integrator, which actually allows real-time integration of directories from multiple sources, on multiple platforms.

4.3.3 Working with OpenLDAP

Red Hat provides OpenLDAP packages in RPM format to support both client and server functions:

- ▶ **openldap-2.0.21-1**: Configuration files, libraries, and documentation for OpenLDAP
- ▶ **openldap-clients-2.0.21-1**: Client programs for OpenLDAP
- ▶ **openldap-devel-2.0.21-1**: OpenLDAP development libraries and header files
- ▶ **openldap-servers-2.0.21-1**: OpenLDAP servers and related files

The OpenLDAP base and client packages are installed by default in a standard RedHat installation. The server packages must be explicitly installed and configured. The OpenLDAP development libraries are only needed if you plan to compile applications or write programs that have LDAP functions.

In addition, RedHat also ships with some packages to LDAP-enable other applications. These are:

- ▶ **auth_ldap-1.4.8-3.i386.rpm**: LDAP authentication module for Apache
- ▶ **nss_ldap-172-3.i386.rpm**: Name server switch (NSS) and pluggable authentication library modules for LDAP that allow applications like login, FTP, mail, etc. to authenticate against LDAP
- ▶ **php-ldap-4.0.6-16.i386.rpm**: LDAP module for PHP applications

Here are the general steps that we follow to get a basic OpenLDAP server up and running and populate it with entries.:

1. Install the OpenLDAP server.
2. Configure the OpenLDAP server using its configuration file.
3. Configure the OpenLDAP client using its configuration file.
4. Start the LDAP server.
5. Add entries to the LDAP directory.

Installing the OpenLDAP server

As we mentioned earlier, the OpenLDAP client is already installed by default on a typical Red Hat installation. To get our LDAP server up and running, we only need to install the server components and modify the configuration files.

To install the server package, simply run:

```
rpm -ivh openldap-servers-2.0.21-1.rpm
```

Configuring the OpenLDAP server

The behavior of the OpenLDAP server is controlled by the `/etc/openldap/slapd.conf` configuration file. Example 4-9 shows a sample `slapd.conf` file. If you want to test your own OpenLDAP server, modify your own `slapd.conf` file to follow the entries shown here.

Example 4-9 *slapd.conf* contents

```
include      /etc/openldap/schema/core.schema
include      /etc/openldap/schema/cosine.schema
include      /etc/openldap/schema/inetorgperson.schema
include      /etc/openldap/schema/nis.schema
include      /etc/openldap/schema/redhat/rfc822-MailMember.schema
include      /etc/openldap/schema/redhat/autofs.schema
include      /etc/openldap/schema/redhat/kerberosobject.schema

database     ldbm
suffix       "o=lspl.ibm.com"
rootdn       "cn=Manager,o=lspl.ibm.com"
rootpw       {SHA}5en6G6MezRroT3XKqkdP0mY/BfQ=
directory    /var/lib/ldap
index        objectClass,uid,uidNumber,gidNumber,memberUid  eq
index        cn,mail,surname,givenname                      eq,subinitial

defaultaccess read
access to attr=userPassword
            by self write
            by anonymous auth
            by dn="cn=Manager,o=lspl.ibm.com" write
access to dn.one="ou=users,o=lspl.ibm.com"
            by self write
            by dn="cn=Manager,o=lspl.ibm.com" write
            by * read
```

Note: If you want some idea of what the entries in the configuration file mean, read on. Otherwise, skip to the next step in “Configuring the OpenLDAP client” on page 42.

The first group of lines, as shown here, are the defaults provided by the RedHat OpenLDAP packages:

```
include      /etc/openldap/schema/core.schema
include      /etc/openldap/schema/cosine.schema
include      /etc/openldap/schema/inetorgperson.schema
include      /etc/openldap/schema/nis.schema
include      /etc/openldap/schema/redhat/rfc822-MailMember.schema
include      /etc/openldap/schema/redhat/autofs.schema
include      /etc/openldap/schema/redhat/kerberosobject.schema
```

In later chapters, you modify these files to include other custom schemas.

The next group of lines, as shown here, define the characteristics of our directory:

```
database     ldbm
suffix       "o=lspl.ibm.com"
rootdn       "cn=Manager,o=lspl.ibm.com"
rootpw       {SHA}5en6G6MezRroT3XKqkdP0mY/BfQ=
directory    /var/lib/ldap
index        objectClass,uid,uidNumber,gidNumber,memberUid  eq
index        cn,mail,surname,givenname                      eq,subinitial
```

A single LDAP server can actually manage several directories. In our case, we only define a single one.

The *rootdn* specifies the distinguished name that manages the database. This user has full access to the LDAP directory. This DN may or may not be an entry on the LDAP directory itself. The *rootpw* specifies the password for the rootdn.

Generating rootpw: You generate the contents of *rootpw* using the `slappasswd` command. For example, you use the plaintext password “secret” as shown here:

```
s secret
```

This generates the following results:

```
{SSHA}5Q1mUZQSFmU6kn6qLwBn1J1erHAq+o5Y
```

Other encryption schemes are available such as {CRYPT}, {MD5}, {SMD5}, and {SHA}. To use these other hashing schemes, for example {SHA}, you enter:

```
-s secret
```

This then generates:

```
{SHA}5en6G6MezRroT3XKqkdP0mY/BfQ=
```

Copy the generated password into the *rootdn* field of the `slapd.conf` file.

Alternatively, you may choose not to use encryption, in which case your *rootdn* field is now:

```
rootpw secret
```

The database line specifies the backend database to use. This can be either `ldbm`, `shell`, or `passwd`. You normally use `ldbm`.

The suffix specifies the DN suffix of queries that will be passed to this backend database. Multiple suffix lines can be given and at least one is required for each database definition. In our case, we specify our organization’s suffix, `o=lspl.ibm.com`.

The directory specifies the location of the database and index files that OpenLDAP generates.

The index lines specify the index type to be maintained for the listed attributes.

The last group of lines, as shown here, define the different types of access to the directory objects:

```
defaultaccess read
access to attr=userPassword
    by self write
    by anonymous auth
    by dn="cn=Manager,o=lspl.ibm.com" write
access to dn.one="ou=users,o=lspl.ibm.com"
    by self write
    by dn="cn=Manager,o=lspl.ibm.com" write
    by * read
```

The default access mode is *read* for all users.

However, for the attribute, `userPassword` may only be used for authentication purposes. A user who connects to the LDAP server using their DN can modify the `userPassword` attribute, but only for his own LDAP entry. The *rootdn* for this directory has write access to all `userPassword` attributes.

We permit everyone read access to the entries found in the “ou=users,o=lspl.ibm.com” branch. There is an exception for the userPassword attribute, whose access we have defined earlier. As before, a user who connects to the LDAP server using their DN can modify all attributes. The rootdn has full write access.

Configuring the OpenLDAP client

The behavior of the OpenLDAP server is controlled by /etc/openldap/ldap.conf configuration file. Example 4-10 shows this file.

Example 4-10 ldap.conf contents

```
HOST 127.0.0.1
BASE o=lspl.ibm.com
```

We tell the LDAP client the IP address of the host. In this case, the LDAP client and server are on the same machine. We also specify the default base DN to use when making queries.

Starting the OpenLDAP server

Start the OpenLDAP server by typing the following command:

```
service ldap start
```

The LDAP service should start. If it generates an error, check your /etc/openldap/slapd.conf file for any possible errors.

Note: If you want to start OpenLDAP in verbose mode to show debugging information when working with LDAP applications, run the following command instead:

```
slapd -d 1
```

The **slapd** command invokes the LDAP daemon directly. The **-d** flag is the debugging level. The higher the number is, the more debugging information it shows.

Populating the LDAP directory

With the OpenLDAP server now up and running, we can start populating the directory with entries. There are two ways to do this:

- ▶ Offline using the **slapadd** command
- ▶ Online using the **ldapadd** command

For now, we use the online method.

Copy the contents of the example LDIF file we showed earlier into an actual file onto your system. Then, enter the following command:

```
ldapadd -D "cn=Manager,o=lspl.ibm.com" -W -x < example.ldif
```

Note the following explanation:

- ▶ The **-D** flag in OpenLDAP commands denotes the DN that you use to bind to the OpenLDAP server.
- ▶ The **-W** flag in OpenLDAP commands means to prompt for the password of the DN.
- ▶ The **-x** flag means to use simple authentication.

At this point, you should see some messages confirming that the contents of the LDIF file were added to the directory. If you encounter an error message, verify the contents of the LDIF file to make sure you did not make any typographical errors.

Searching OpenLDAP

You can search, add, modify, or delete entries in the OpenLDAP directory, provided you have the right permissions as defined in the `slapd.conf` file. Let's start first with some basic search operations, using `ldapsearch`.

To search for a particular attribute of a DN, enter:

```
x {uid=dominique,ou=users,o=lspl.ibm.com} mail
```

After you enter your password, you should see the following result:

```
dn: uid=dominique,ou=users,o=lspl.ibm.com
uid: dominique
cn: Dominique Cimafranca
givenName: Dominique
sn: Cimafranca
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
mail: dominique@lspl.ibm.com
```

You can also search for multiple attributes from a particular DN:

```
-x {uid=dominique,ou=users,o=lspl.ibm.com} givenname cn
```

If you want all the readable attributes from a DN, type:

```
-x {uid=dominique,ou=users} "*"
```

Note that, in this example, we did not need to specify the suffix. We already instructed the client to use a base of "o=lspl.ibm.com".

If you want to list out all the DNs of a particular branch in the directory, enter:

```
-x {uid=dominique,ou=users} "*"
```

Modifying LDAP entries

Earlier we mentioned that LDIF files may be used for both adding and modifying LDAP entries. The format for an LDIF file when used for modify operations is similar to the file format for an add operation.

Suppose, for example, that we wanted our LDAP entry to have additional attributes. We highlighted these in bold in Example 4-11.

Example 4-11 Modified LDAP entry with new attributes

```
dn: uid=dominique,ou=users,o=lspl.ibm.com
uid: dominique
cn: Dominique Cimafranca
givenName: Dominique
sn: Cimafranca
objectClass: top
objectClass: person
objectClass: organizationalPerson
```

```
objectClass: inetOrgPerson
objectClass: account
objectClass: posixAccount
mail: dominique@lspl.ibm.com
loginShell: /bin/bash
uidNumber: 1000
gidNumber: 1000
homeDirectory: /homenfs/nfs/dominique
gecos: Dominique Cimafranca
```

We create an LDIF file with the contents listed in Example 4-12.

Example 4-12 LDIF file for modification modify.ldif

```
dn: uid=dominique,ou=users,o=lspl.ibm.com
changetype: modify
add: objectclass
objectclass: account
-
add: objectclass
objectclass: posixAccount
-
add: loginShell
loginShell: /bin/bash
-
add: uidNumber
uidNumber: 1000
-
add: gidNumber
gidNumber: 1000
-
add: homeDirectory
homeDirectory: /homenfs/home/Dominique
-
add: gecos
gecos: Dominique Cimafranca
```

Then we run the command:

```
ldapmodify -D "cn=Manager,o=lspl.ibm.com" -W -x -f modify.ldif
```

This makes the required modifications to the LDAP entry.

Setting passwords for the OpenLDAP entries

Since we are working with user accounts in this LDAP directory, it makes sense to add passwords to them. The schema of one of the object classes we used, `inetOrgPerson`, allows us to use the attribute `userPassword`.

The command to add or change passwords to a DN is `ldappasswd`.

To set up a password for a user as the administrator, enter:

```
-x "uid=dominique,ou=users,o=lspl.ibm.com"
```

The `-S` flag prompts you for the new password for this user.

In this example, we set up the userPassword attribute for the entry "uid=dominique,ou=users,o=lspl.ibm.com". Since this password entry does not exist yet, we need to enter it using our administrator credentials.

If you want to include the plaintext password in the command line, use the following command instead:

```
-x "uid=dominique,ou=users,o=lspl.ibm.com"
```

To change a password for a user using that user's DN, type:

```
ldappasswd -D "cn=dominique,ou=users,o=lspl.ibm.com" -W  
"uid=dominique,ou=users,o=lspl.ibm.com"
```

This prompts for the old password and the new password.

Additional resources

This has been a cursory introduction into OpenLDAP, with emphasis on getting a working installation up and running. For more information on OpenLDAP, go to the main Web site at:

<http://www.openldap.org>

In particular, look for the administration guide, which you can find at:

<http://www.openldap.org/doc/admin/>

For a more in-depth discussion of LDAP, see the excellent tutorial on LDAP Version 3 by Adam Williams. The tutorial covers the basics of LDAP and goes deeper into the workings of OpenLDAP, with several examples. This tutorial is available via File Transfer Protocol (FTP) at:

<ftp://ftp.kalamazoolinux.org/pub/pdf/ldapv3.pdf>

4.3.4 gq: A graphical LDAP browser

You may be feeling dizzy from all the command-line options that we've shown. Fortunately, an open-source graphical LDAP browser is available that you can use – *gq*. What's more, *gq* is part of the standard packages installed by the Red Hat default setup.

Setting up the LDAP browser

To set up the LDAP browser, follow these steps:

1. Bring up the *gq* client. Type the following command:

```
# gq
```

This opens the client interface as shown in Figure 4-5.

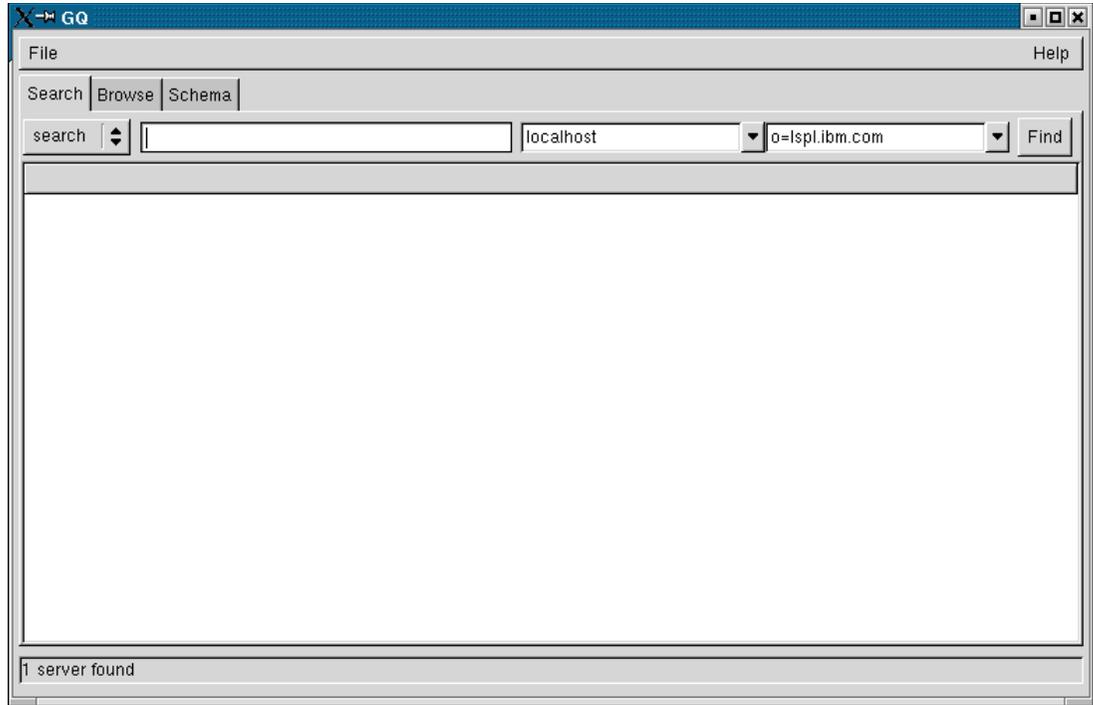


Figure 4-5 gq interface

2. Configure the client for your LDAP server. Click **File-> Preferences** to bring up the Preferences dialog (Figure 4-6).
3. Click the **Servers** tab to set up some LDAP servers to manipulate with gq.
4. Click **New**.

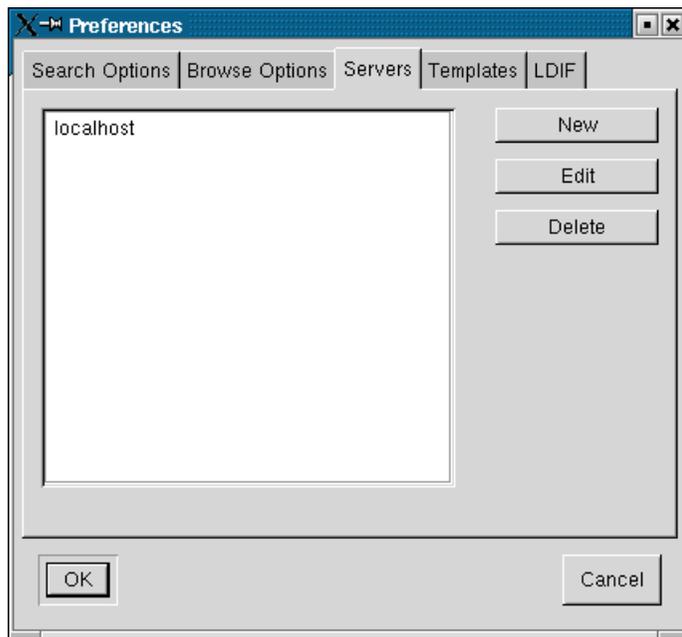


Figure 4-6 gq Servers tab

5. The New server window opens (Figure 4-7). On the General tab, enter the details of the server to which you will be connecting.

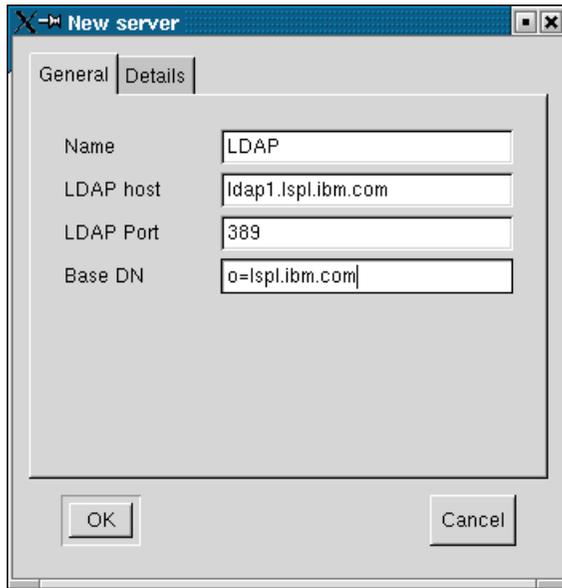


Figure 4-7 New server General page

6. Click the **Details** tab (Figure 4-8).
7. Specify more information such as the DN to bind to the LDAP server and the password. Also enter your search attribute.

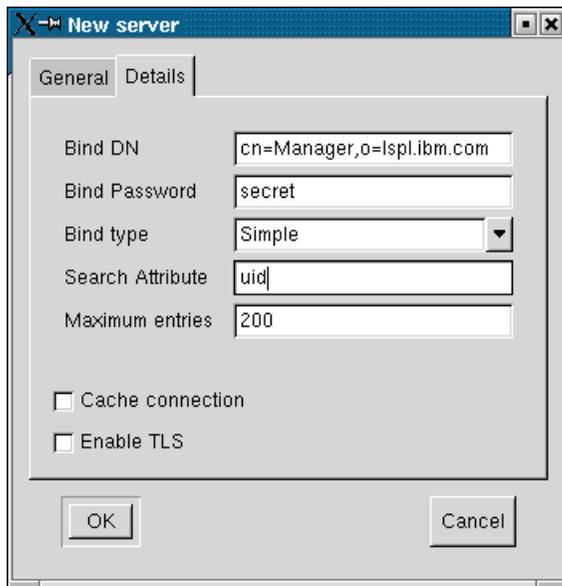


Figure 4-8 New server Details page

8. Click **OK**.

At this point, your LDAP client is ready to go.

Working with gq

Now let's perform the most basic function, which is to search:

1. On the main window (Figure 4-9), select the LDAP server you just configured. Type the wildcard * in the search field and click **Find**. This shows you all the LDAP entries matching your search pattern. Now you see listed all the LDAP entries you created so far.

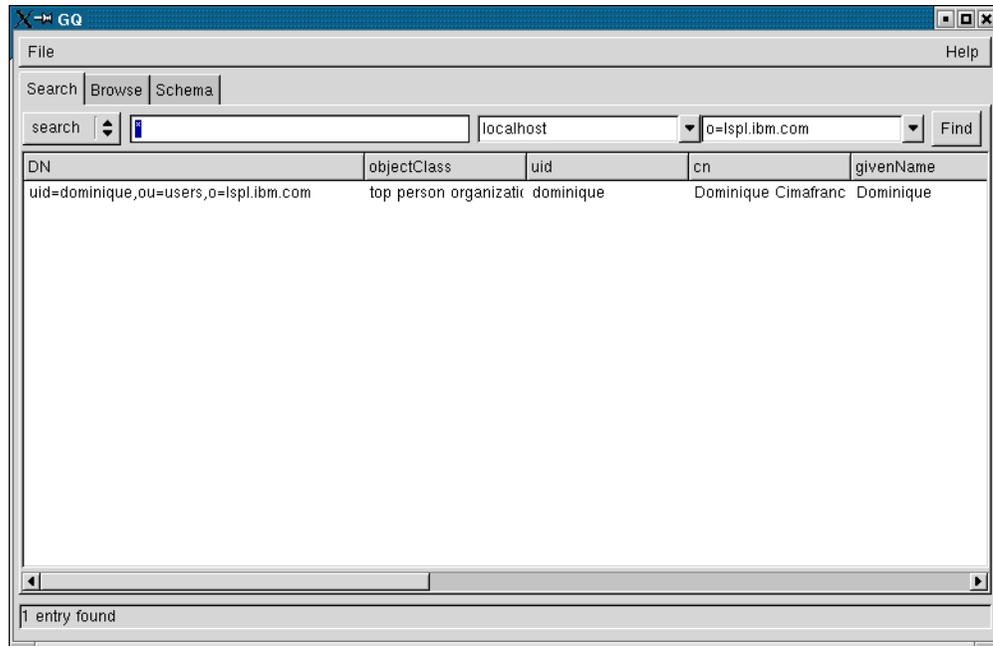


Figure 4-9 gq server list

2. Double-click an existing entry. This shows you the attributes and classes of the LDAP entry.

3. You can edit these attributes of the entry from the window (Figure 4-10) that opens. Click **Apply** to commit the changes to the LDAP directory.

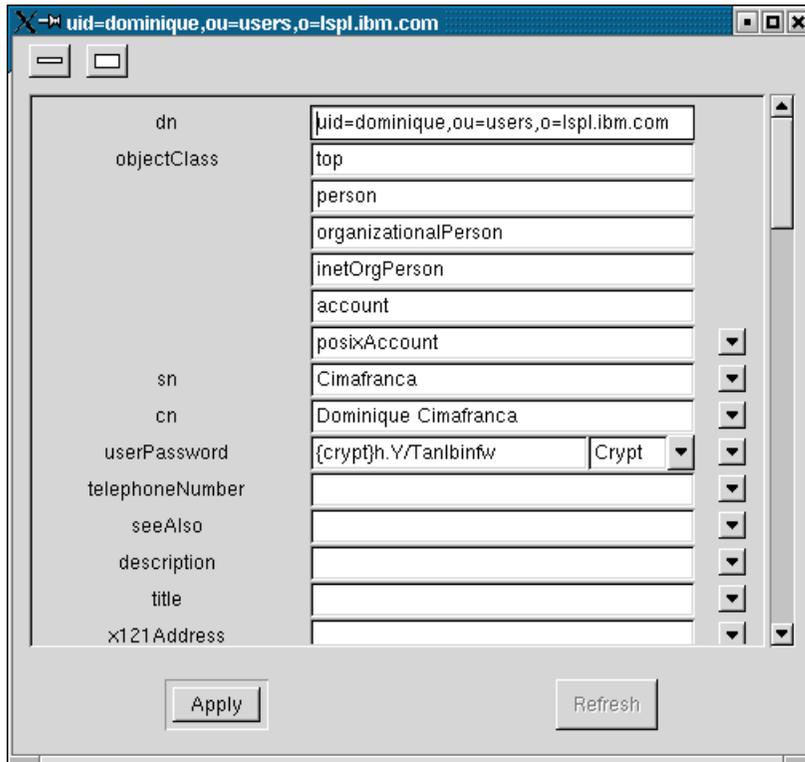


Figure 4-10 *gq server attributes*

If you receive an object class violation after editing an entry, this means that you missed entering a required attribute. Review the entry for the missing field before you commit it.

How do you add a new entry? You can do that by using an existing entry as a template.

1. Right-click an entry and choose **Use as template**.
2. This brings up a form (Figure 4-11) for an LDAP object that follows the object classes and attributes of the template. Note that this form is similar to the form shown earlier. Be sure you enter all the required fields. Otherwise the LDAP server (and gq) issues a message about an object class violation. Click **OK** when you are finished.

dn	ou=users,o=lspl.ibm.com
objectClass	top person organizationalPerson inetOrgPerson account posixAccount
sn	
cn	
userPassword	<input type="text"/> Clear
telephoneNumber	
seeAlso	
description	
title	
x121Address	

Figure 4-11 GQ new entry from template

- From the main window, click the **Browse** tab to see a tree view of the LDAP directory (Figure 4-12). You can also view and edit the details of the LDAP objects from this view by highlighting an entry. Right-clicking an entry allows you create new entries or export the LDAP directory tree to an LDIF file.

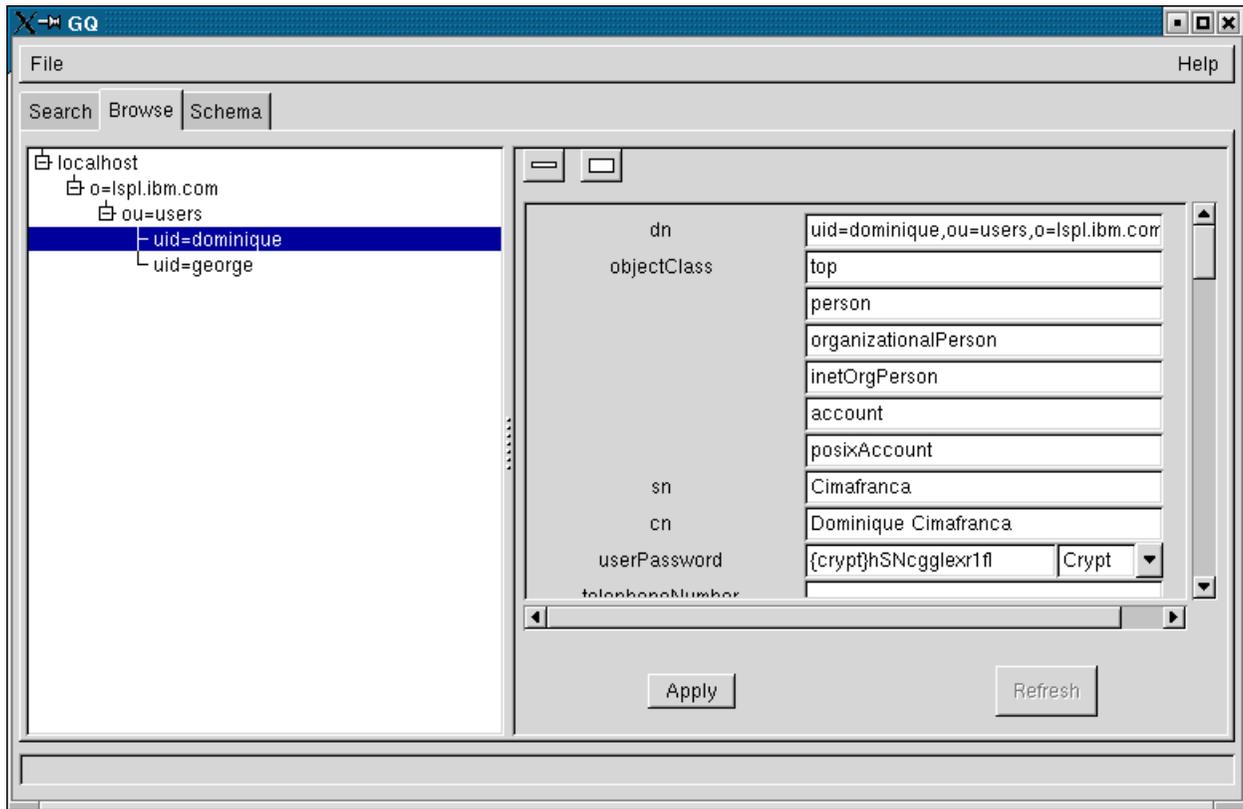


Figure 4-12 gq Browse page to browse the LDAP directory

The Schema page (Figure 4-13) is helpful if you're designing an LDAP directory and wondering which object classes go with which attributes. This page provides a view of all the object classes. It shows you what the required and optional attributes of each are. It gives you a list of the available attributes and tells you which object classes make use of them.

All in all, gq is a very handy tool for managing your LDAP directory.

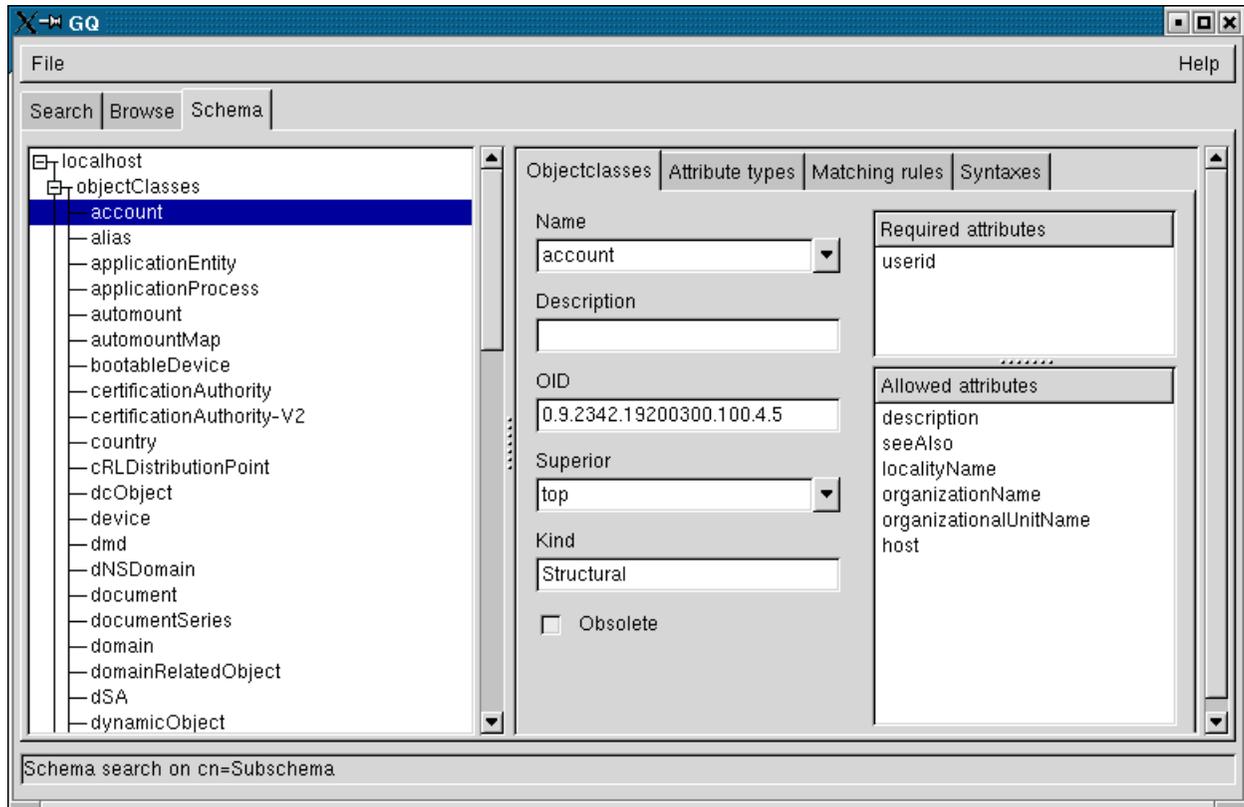


Figure 4-13 gq Schema page

Why can't you just use gq?

You may be wondering why we went through the pain of setting up the LDAP directory using the command line options. The reason is that gq works on an already-existing LDAP directory. The contents may be very basic, but it should give gq enough structure in the directory tree to work with.

Also gq is only ideal for viewing and editing entries one at a time. If you need to make massive changes, like reading in hundreds of entries or adding attributes to all the objects in an LDAP branch, you can't do that with gq. You need to use the command-line LDAP tools.

4.3.5 Server authentication with LDAP

Now that we have a properly running LDAP server, let's look at practical applications for it. Let's start with something basic but immediately useful – *central authentication* for a group of servers.

Central authentication is useful in a BladeCenter environment where users need to maintain accounts across several blades. This means that users can log on to any server using the same combination of username and password.

This functionality is similar to Network Information Services (NIS). You can think of it as having a centralized `/etc/passwd` file.

This configuration is possible using the stock packages from a Red Hat distribution. Apart from the OpenLDAP server packages, you also need the `pam_ldap` and `nss_packages`.

RFC2307 describes authentication using LDAP in more detail.

Pluggable authentication module

PAM is a system service used by applications and servers to determine who has the right to perform an action. When you log in to your system, the login program provides PAM with the username and password. PAM returns an answer as to whether the pair is valid.

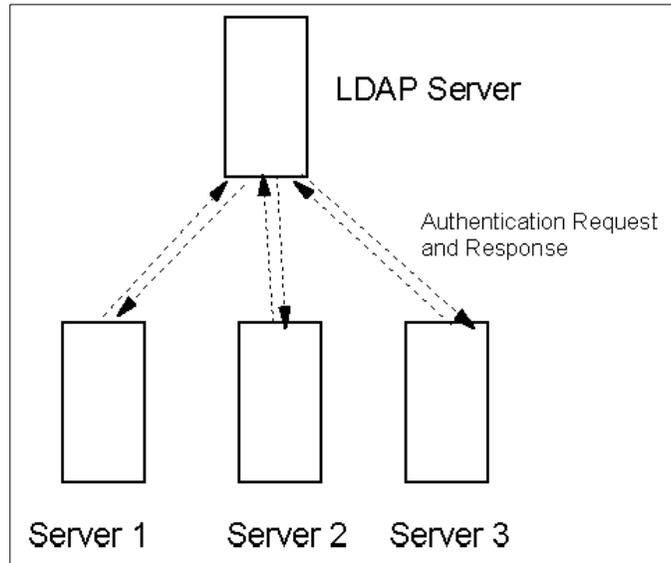


Figure 4-14 Servers requesting authentication information from LDAP server

PAM is a way by which several applications can use the same authentication system within a server.

We can extend this concept to servers in a network. PAM can check with the LDAP directory to verify the username and password:

<http://www.kernel.org/pub/linux/libs/pam/Linux-PAM.html/pam.html>

For more information about PAM, see the FAQ on the Web at:

<http://www.kernel.org/pub/linux/libs/pam/FAQ>

Name server switch (NSS)

NSS provides processes with a common interface to repositories of system information, such as the correlation between a username and UNIX user ID. If PAM is about authentication, NSS is about the maps that are commonly used by several applications on UNIX systems.

Various programs need to be configured to work correctly in the local environment. Traditionally, this was done by using files such as `/etc/passwd`. However, other name services, such as NIS and DNS, became popular and were hacked into the C library, usually with a fixed search order.

The GNU C Library, which Linux uses, implements the maps using NSS.

The databases available in the NSS are:

- ▶ **aliases**: Mail aliases
- ▶ **ethers**: Ethernet numbers
- ▶ **group**: Groups of users
- ▶ **hosts**: Host names and numbers
- ▶ **netgroup**: Network wide list of host and users
- ▶ **networks**: Network names and numbers

- ▶ **protocols**: Network protocols
- ▶ **passwd**: User passwords
- ▶ **rpc**: Remote procedure call names and numbers
- ▶ **services**: Network services
- ▶ **shadow**: Shadow user passwords

NSS can also query LDAP for this information.

Schemas and required attributes

We can place all the information that PAM and NSS use into LDAP to provide a form of common authentication for several applications. RFC 2307 describes the attributes that an LDAP entry must have to be used for UNIX system authentication.

The attributes are:

- ▶ uid
- ▶ cn
- ▶ userPassword
- ▶ loginShell
- ▶ uidnumber
- ▶ gidnumber
- ▶ homedirectory
- ▶ gecos

The LDAP entry must also be of object class posixAccount and account. See Example 4-13.

Example 4-13 Modified LDAP entry with new attributes, with authentication attributes highlighted

```
dn: uid=dominique,ou=users,o=lspl.ibm.com
uid: dominique
cn: Dominique Cimafranca
givenName: Dominique
sn: Cimafranca
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: account
objectClass: posixAccount
mail: dominique@lspl.ibm.com
loginShell: /bin/bash
uidNumber: 1000
gidNumber: 1000
homeDirectory: /homenfs/nfs/dominique
gecos: Dominique Cimafranca
```

You can add these additional attributes using gq or LDIF files.

Configuring a server to authenticate against an LDAP server

Populate the LDAP directory with the user accounts that you want to test the system against:

1. On the server, log in as root and enter the command:

```
# authconfig
```

This brings up the text-based menu shown in Figure 4-15.

2. Select LDAP as the authentication method. In the Server field, enter the IP address of the LDAP server. In the Base DN field, enter the base DN.

The choices you make on this window update the following files:

- **/etc/nsswitch.conf**: The configuration file for NSS.
- **/etc/ldap.conf**: This is the configuration file for the server to act as LDAP client.

Click **Next**.

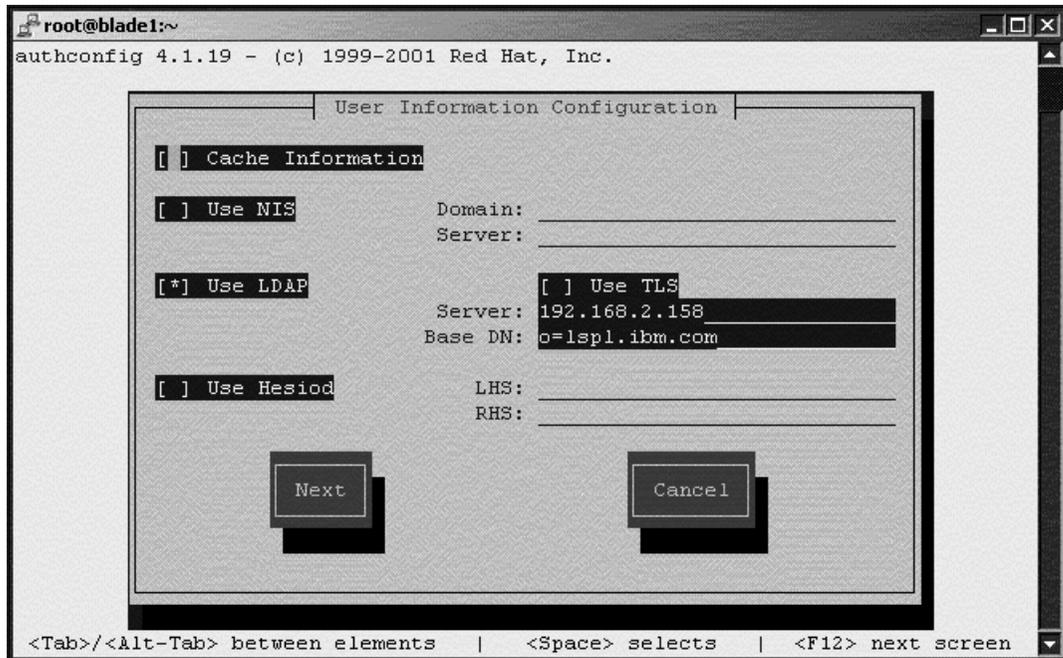


Figure 4-15 Authconfig main menu

3. The display shown in Figure 4-16 opens. Select **Use LDAP Authentication** as one of the authentication methods. As before, specify the Server and Base DN.
The choices you make on this display affect the contents of `/etc/pam.d/login`.
Click **OK**.

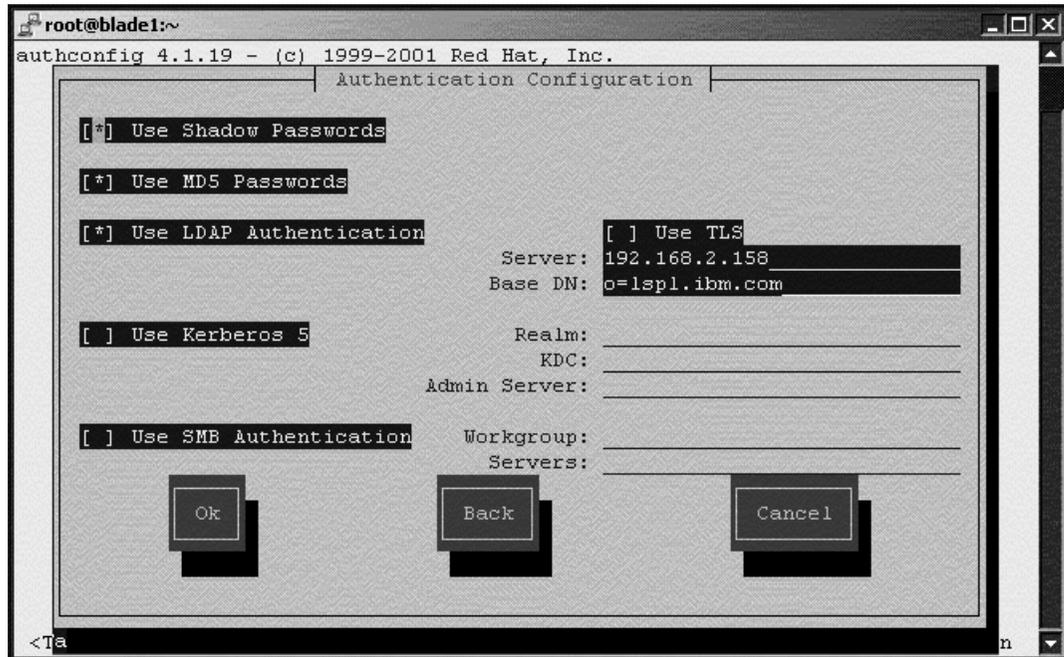


Figure 4-16 Authconfig menu

The server is ready to authenticate against the LDAP server. You can test this by logging in to the server using the user ID and password of one of the accounts on the LDAP directory.

```
-l dominique 127.0.0.1
```

If you logged in successfully, then you know that LDAP authentication works. If for some reason your LDAP authentication failed, check the following items:

- ▶ Can you contact your LDAP server? Perform a query on your LDAP server using the **ldapsearch** tool:

```
ldapsearch -D "cn=Manager,o=lspl.ibm.com" -h 192.168.2.158 --x
{uid=dominique,ou=users,o=lspl.ibm.com} ""*
```

- ▶ Does your LDAP entry have the right attributes? You should be able to retrieve the attributes as shown in (example no.), along with the `userPassword`.
- ▶ Recheck your `authconfig` settings.

Home directories

You may have noticed that when you log on to an account defined in your LDAP directory, it doesn't have a home directory. The simplistic approach is to create the specified home directories for your users on the server, but this would be a wrong solution. The home directories are not portable across the different servers.

The answer is to use a shared directory, either NFS or Samba. Better yet, you can use `automount` to mount the user home directories only when they are needed.

Automating the conversion with scripts

Red Hat includes MigrationTools from PADL to automate the conversion of Linux authentication files to LDAP. These are found in the `/usr/share/openldap/migration` directory. MigrationTools are a set of Perl scripts for migrating users, groups, aliases, hosts, netgroups, networks, protocols, RPCs, and services from existing name services to LDAP.

Here is a brief description of all migration scripts:

- ▶ **migrate_base.pl** creates naming context entries, including subordinate contexts such as `ou=people` and `ou=devices`.
- ▶ **migrate_aliases.pl** migrates aliases in `/etc/aliases`.
- ▶ **migrate_group.pl** migrates groups in `/etc/group`.
- ▶ **migrate_hosts.pl** migrates hosts in `/etc/hosts`.
- ▶ **migrate_networks.pl** migrates networks in `/etc/networks`.
- ▶ **migrate_passwd.pl** migrates users in `/etc/passwd`.
- ▶ **migrate_protocols.pl** migrates protocols in `/etc/protocols`.
- ▶ **migrate_services.pl** migrates services in `/etc/services`.
- ▶ **migrate_netgroup.pl** migrates netgroups in `/etc/netgroup`.
- ▶ **migrate_netgroup_byuser.pl** migrates the `netgroup.byuser` map. It requires `revnetgroup`.
- ▶ **migrate_netgroup_byhost.pl** migrates the `netgroup.byhost` map. It requires `revnetgroup`.
- ▶ **migrate_rpc.pl** migrates RPCs in `/etc/rpc`.

These scripts are useful if you want to study the LDAP entries required by Red Hat and to automate the actual migration. However, be careful when using the scripts if you are not familiar with them. They will migrate all the existing information, even information that you may not want migrated. For example, the root account should never be on an LDAP server, but instead be on individual machines.

Be especially careful about using the `migrate_all_nisplus_online.sh` and `migrate_all_nisplus_offline.sh` scripts that Red Hat provides. You may use them instead to generate LDIF files that you can study and edit before you enter them into the LDAP directory using the `ldapadd` or `ldapmodify` tools.

Extending to other applications

The example we use in the previous section illustrates how to use LDAP as common authentication for UNIX logins through PAM. But the same PAM authentication can be used by several other applications, such as FTP and mail.

This is a double-edged sword. It gives you convenience for a centralized authentication. PAM-LDAP and NSS-LDAP are quite easy to set up. However, you must be careful about other applications using LDAP authentication when you don't want them to.

You may have to go through the configuration files in `/etc/pam.d/` to verify that only the applications you select will use LDAP.

Additional resources

For more information on PAM, NSS, and LDAP, read the Red Hat Reference Guide that ships with your distribution.

Also consult the following sources:

- ▶ The Quick Start Guide on the OpenLDAP Web site:
<http://www.openldap.org/doc/admin/quickstart.html>
- ▶ The LDAP Linux HOWTO from the Linux Documentation Project:
<http://www.redhat.com/mirrors/LDP/HOWTO/LDAP-HOWTO.html>

One area we did not go deep into is Samba authentication with LDAP. This is covered in the tutorial “Using an LDAP directory for Samba authentication” by Tom Syroid. You can find this document on the Web at:

http://www-1.ibm.com/servers/esdd/tutorials/smb_ldap.html

For more on Samba LDAP integration, see the following Web sites:

<http://www.coe.tamu.edu/cs/Manuals/Samba/Samba-LDAP-HOWTO.html>
http://www.unav.es/cti/ldap-smb/ldap-smb-2_2-howto.html

4.3.6 Apache authentication with LDAP

We showed you how to provide authentication for a server using PAM, NSS, and LDAP. There are other ways to provide authentication. For example, some applications may incorporate LDAP support and authentication directly in the application itself (instead of relying on an external module like PAM).

Let’s look at an example of LDAP authentication performed by an application itself – basic authentication on the Apache Web server. The Apache project already includes a module called *mod_auth_ldap* that performs this function.

Basic authentication is the simplest method of authentication. When a Web server directory is protected using this method, Apache sends an authentication request to the Web browser. Upon receiving this request, the browser asks the user to supply a user name and password.

Schemas and required attributes

The LDAP authentication module for Apache works with the objects that we created so far. In particular, only the user ID (uid) and userPassword attributes are necessary. No modifications are necessary.

The Apache LDAP module is flexible enough to use other attributes that you specify.

Required packages

The Apache package is installed by a Red Hat Advanced Server default installation. If the package was not installed, you must install it yourself using the following command:

```
# rpm -ivh apache-1.3.27-2.rpm
```

Next, you must install the *auth_ldap* package by entering the following command:

```
# rpm -ivh auth_ldap-1.4.8-3.rpm
```

Enabling Apache to authenticate against an LDAP server

To use LDAP authentication, you only have to edit the Apache configuration file in */etc/httpd/conf/httpd.conf*.

Find and uncomment the following two lines in `/etc/httpd/conf/httpd.conf`:

```
LoadModule auth_ldap_module modules/mod_auth_ldap.so
AddModule auth_ldap.c
```

Password protecting a Web directory using LDAP

Consider the case where you want to password protect the contents of `/var/www/secret`. In this case, you need to add the following directives into your `httpd.conf` file:

```
Alias /secret/ "/var/www/secret"

<Directory "/var/www/secret">
    Options Indexes FollowSymLinks
    AuthType Basic
    AuthName "Secret files"
    AuthLDAPURL ldap://blade2.lspl.ibm.com/ou=Users,o=lspl.ibm.com
    Require valid-user
    Order allow,deny
    Allow from all
</Directory>
```

Restart the server for the changes to take effect. Enter the following command:

```
# service httpd restart
```

When you try to access the secret directory of your Web server, you see the pop-up authentication box.

The four lines involved in the authentication process are:

- ▶ **AuthType Basic:** This tells Apache that we are using basic authentication.
- ▶ **AuthName “Secret Files”:** This is the label that is placed in the pop-up authentication box.
- ▶ **AuthLDAPURL:** This tells Apache to run a search against the LDAP directory located at `blade2.lspl.ibm.com`, under the `ou=Users,o=lspl.ibm.com` branch. By default, `auth_ldap` searches against the `uid` attribute.
- ▶ **Require valid-user:** This line tells Apache to permit any authenticated user access to this directory.

Another combination using .htaccess

Another possibility of LDAP authentication is to use `.htaccess`. Rather than write the Authentication directives into `httpd.conf`, you may opt to put them in a file called `.htaccess` inside a directory that you want to protect.

You can find complete documentation for `mod_auth_ldap` on the Web at:

http://www.rudedog.org/auth_ldap/



Wiring: File services with Samba and NFS

High-availability file sharing services are an important component of a robust network infrastructure. Within an organization, they may be used to share directories and files for ordinary users. Within a portal implementation, they may be used to provide a common file system for high-availability services like web and e-mail.

This chapter looks at two core open source components for file sharing, Samba and Network File System (NFS).

5.1 Working with Samba

The Samba software suite is a collection of programs that implements the Server Message Block (SMB) protocol for UNIX systems. SMB is sometimes also referred to as the Common Internet File System (CIFS), Local area network (LAN) Manager, or NetBIOS protocols.

Samba is a mainstay server for networks that need to integrate UNIX and Windows systems. It allows you to blend a mix of Windows and Linux machines together without requiring a separate Windows NT, 2000, or 2003 server.

As of this writing, the latest stable version of Samba is 2.2.8.

5.1.1 Required Samba packages

You need to install the following packages to set up your Linux server as a Samba client:

- ▶ samba-common
- ▶ samba-client

To set up a Samba server, you need to install the following packages:

- ▶ samba-common
- ▶ samba-client
- ▶ samba
- ▶ samba-swat (optional)

The samba package is the server itself. The samba-swat package is a Web-based user interface for configuring the Samba server.

5.1.2 Configuring Samba as a basic file server

The Samba server configuration file installed by the RPM packages is stored in `/etc/samba/smb.conf`. You can modify this file directly editing it or by using the Samba Web Administration Tool (SWAT). To begin, we configure Samba by directly editing the file.

The SAMBA configuration file `smb.conf` is divided into two main sections:

- ▶ **Global settings:** These settings affect the general operation of the server.
- ▶ **Share definitions:** These settings are used in defining shares. A share is a directory on share server that is accessible over the network by SMB client systems. The share definitions define home directories, printer shares, and shared directories.

As shipped, the `smb.conf` file is sufficient to help you start sharing user home directories and printers. A Linux user can mount their designated home directory (and currently, only their home directory) on the server from either a Windows or Linux clients. If printers are defined on the server, the user can also connect to the printer.

We suggest that you make the following modifications to `smb.conf`:

- ▶ `workgroup = MYGROUP`: This is the workgroup of the Windows clients that will connect to the Samba server.
- ▶ `hosts allow = 192.168.2`: This filter allows you to restrict which hosts, defined by the IP address, can connect to the Samba server.

We revisit this configuration file in subsequent sections of this chapter.

5.1.3 Adding Samba users

Next, we need to define Samba users on the server. For basic configurations, Samba accounts correspond to Linux user accounts on the server.

To create a Samba user, use the **smbadduser** command. Example 5-1 shows a sample session, assuming we already have an existing Linux user account called *user*.

Example 5-1 Creating a Samba user

```
[root@blade11 root]# smbadduser user:user
Adding: user to /etc/samba/smbpasswd
Added user user.
-----
ENTER password for user
New SMB password:
Retype new SMB password:
Password changed for user user.
Password changed for user user.
```

The first parameter of **smbadduser** is the Linux user account ID (*user*). The second parameter, separated by the colon (:), is the designated Windows ID, which is how the account appears to Windows systems.

Executed for the first time, the **smbadduser** command creates the `/etc/samba/smbpasswd` file. The home directory of *user* is automatically shared by Samba server because of the definition in the `smb.conf` file shown in Example 5-2.

Example 5-2 Home directories shared in Samba

```
[homes]
  comment = Home Directories
  browseable = no
  writable = yes
  valid users = %S
  create mode = 0664
  directory mode = 0775
```

5.1.4 Samba passwords

To change the Samba password, run the **smbpasswd** command. The root user running **smbpasswd** can change the Samba password of another user. An ordinary user running **smbpasswd** can change their own Samba password, after they authenticate their old password.

The password to Samba is not the same as the password to the Linux server. You can make them equivalent, but it does not mean that they will be stored in the same location. If you change one, it does not automatically change the other.

This is the down side of a basic Samba installation, which can be remedied with a common authentication database, such as LDAP.

5.1.5 Connecting to the Samba server using smbclient

If the Samba server is not running, start it now. You can use the `service` command to do this as shown here:

```
# service smb start
```

From another machine or from one of the other blades, connect to the server using the `smbclient` command as shown here:

```
% smbclient //blade11.lspl.ibm.com/user -U user
```

This prompts you for the password, after which you are connected using an FTP-like client.

5.1.6 Connecting to the Samba server using smbmount

Running as root, you can connect to the Samba server using the `smbmount` command. Define a mount point and enter the following command:

```
# smbmount //blade11.lspl.ibm.com/user mountpoint -o username=user
```

This prompts you for the password of the user account you are logging on as.

5.1.7 Connecting to the Samba server from a Windows machine

If you modified your workgroup directive of `/etc/samba/smb.conf` to reflect the workgroup of your Windows machines, your Samba server should be visible when you open Network Neighborhood or My Network Places.

You can connect to the file share as you would with any Windows folder. You must authenticate using your username and password.

5.1.8 Automatically mounting a Samba directory at boot time

To automatically mount a Samba shared directory at boot time, edit the `/etc/fstab` file of the Samba client. Add the following line or something similar:

```
//blade11/user /mnt/data smb username=user,password=mypassword 0 0
```

When you type `mount -a` or when you boot up the server, the shared directory from blade11 is mounted on `/mnt/data` automatically.

But wait. The username and password are visible to everyone who can read the `/etc/fstab` file, which is usually the case. This is *bad* security. Instead, you should use this line:

```
//blade11/user /mnt/data smb credentials=/etc/samba/credentials 0 0
```

You have to create a file called `/etc/samba/credentials` with the following contents:

```
username=user  
password=mypassword
```

Make sure that you restrict the file so that only root can read it:

```
# chmod 600 /etc/samba/credentials
```

5.1.9 Sharing additional directories

Throughout our examples, we have worked with the home directories defined in the Linux accounts. How about a case where you want to share a common directory with several users?

You can define a new share in `/etc/samba/smb.conf`. For example, to make available the directory `/var/data` as a share called *blast*, you would add the following lines in `smb.conf`:

```
[blast]
  comment = Blast data
  path = /var/data
  valid users = user
  public = no
  writable = yes
  create mask = 0765
```

The directory is accessible to the valid users on other servers.

5.1.10 For more information on Samba

If you're looking for more in-depth information on Samba, see the main Samba Web site. Documentation, source code, and updates are also available on the Web at:

<http://www.samba.org>

You may also want to refer to the book *Using Samba*, by Robert Eckstein, David Collier-Brown, and Peter Kelly. A softcopy version of this book is available on the Web at:

<http://www.oreilly.com/catalog/samba/chapter/book>

Plus, you may want to refer to the Redpaper *Deploying Samba on IBM @server BladeCenter*, REDP3595. Parts of the Samba section for this Redbook were adapted from this paper.

5.2 Working with NFS

Another method of sharing files is through the NFS. NFS was developed to allow machines to mount a disk partition on a remote machine as if it were on a local hard drive. Like Samba, this allows for fast, seamless sharing of files across a network.

NFS predates Samba by several years. It was originally written by Sun Microsystems to link together UNIX servers. These days, non-UNIX and non-Linux clients can use NFS.

NFS on Linux was written by Olaf Kirch and Alan Cox. You can find the project on the Web at:

<http://nfs.sourceforge.net>

The current implementation of NFS on Linux runs it within the kernel, which enables better performance.

5.2.1 Required NFS packages

You need the *nfs-utils package*. The `nfs-utils` package provides a daemon for the kernel NFS server and related tools.

5.2.2 Configuring NFS

The main configuration files you need to edit to set up an NFS server is `/etc/exports`. The `/etc/exports` file contains a list of entries, each entry indicating a volume to be shared and how it is to be shared. An entry in `/etc/exports` typically looks like this example:

```
directory    machine1(option11,option12) machine2(option21,option22)
```

The directory entry represents the directory that you want to share. If you share a directory, then all directories under it, within the same file system, are also shared.

The entries machine1 and machine2 represent the client machines that have access to the directory. The machines may be listed by their DNS address or their IP address. Using IP addresses is more reliable and more secure.

When specifying hosts to be allowed to use a particular exported filesystem, a variety of methods can be used, including:

- ▶ **Single host:** Where one particular host is specified with a fully qualified domain name, hostname, or IP address.
- ▶ **Wildcards:** Where an asterisk (*) or question mark (?) character is used to take into account a grouping of fully qualified domain names or IP addresses or those that match a particular string of letters.
- ▶ **IP networks:** These networks allow the matching of hosts based on their IP addresses within a larger network. For example, 192.168.0.0/28 allows the first 16 IP addresses, from 192.168.0.0 to 192.168.0.15, to access the exported file system but not 192.168.0.16 and higher.



Doorways: Web serving and messaging

Every building needs an access way, such as a doorway. For most software applications, this is a communications protocol. For modern organizations, the access way is through electronic communication mechanisms such as the telephone, Web applications, e-mail, and instant messaging.

In any building, the entrance is the first architectural feature that you come into contact with. Architects use an entrance to form an impression in the mind of the person entering the building. You should use great care that the entrances you design and implement is robust and usable.

This chapter shows you how to implement several electronic entrances, including Web serving, instant messaging, and e-mail.

6.1 Web serving

What can be said about Web serving that has not been said before? One of the most popular Web servers is the open source Apache Web server. In fact, the Apache Web server is a core component of IBM's WebSphere Web application environment.

6.1.1 The Apache Web server

The Apache Web server is probably the most popular project of the Apache Software Foundation. It started in 1995 when a group of Webmasters created an updated version of the National Center for Super Computing Applications (NCSA) Hypertext Transfer Protocol (HTTP) daemon, a public domain HTTP server that had not seen any new development since 1994. The first public release of the Apache HTTP Server was based on version 1.3 of the NCSA HTTP daemon plus a set of patches from the Apache group.

The popularity of the Apache HTTP Server started to rise. Since April 1996, the Apache HTTP Server has become the most popular Web server on the Internet. It is available for a multitude of platforms. Because of its open source nature, it is also used in other products. For example, IBM's HTTP Server is based on the Apache code. Currently about 60% of all Web sites run a version of the Apache HTTP Server according to the NetCraft Web server survey. You can learn about the details of this survey on the Web at:

<http://www.netcraft.com/survey/>

In 2002, the Apache HTTP Server project took a new step by releasing version 2.0. Using a newly developed runtime layer, Apache 2.0 can now use the features of the different platforms on which Apache runs better. For a complete list of new features in Apache 2.0, see:

http://httpd.apache.org/docs-2.0/new_features_2_0.html

This chapter discusses installing and configuring Apache 2.0. If you are interested in Apache 1.3 or simply installing Apache, see *Deploying Apache on IBM @server BladeCenter*, REDP3588.

6.1.2 Installing Apache HTTP Server Version 2.0

The easiest way to install Apache 2.0 is to install a Linux distribution that contains it. Since we are using Red Hat Advanced Server 2.1 on our blade servers, we must use another way to install Apache 2.0. If there is a module or patch that is not part of a standard RPM, it is useful to know how to download and build Apache from source.

We compile Apache 2.0 from the source in a separate directory so it does not interfere with the standard Apache 1.3 installation. The Secure Sockets Layer (SSL) module comes with a base Apache source package. The Perl and PHP modules are also installed from source.

6.1.3 Installing Apache HTTP Server and the SSL module

To install the Apache HTTP Server and the SSL module, follow these steps:

1. Download the latest version of the apache2 source from:

<http://www.apache.org/dist/httpd>

We downloaded the *httpd-2.0.43.tar.gz* package.

2. Extract the source using the following command:

```
tar -xzf httpd-2.0.43.tar.gz
```

3. Go to the source directory:

```
cd httpd-2.0.43.
```

4. Configure the source with the following code:

```
./configure --with-mpm=prefork --enable-modules=all --enable-mods-shared=all  
--enable-cgi --enable-ssl --enable-proxy --enable-so --with-ssl=/usr/include/
```

We configure Apache to build all modules as shared modules and to build the SSL and proxy modules. We also choose the prefork processing module. This is a non-threaded module. This is because Perl in Red Hat Advanced Server 2.1 is still version 5.6. Perl 5.6 does not support multithreading. Instead, you need Perl V5.8. If you do not want to build `mod_perl`, you can choose a threaded processing module, such as *worker*.

5. Start the compilation with the **make** command. This could take a while to finish.
6. If the compilation was successful, you can install Apache 2.0 with **make install**. Apache 2.0 is now installed in the directory `/usr/local/apache2`. This way of installing Apache does not create a script for starting and stopping Apache in the `/etc/init.d` directory. This means you cannot use the **service** and **chkconfig** commands.

6.1.4 Installing the Perl module

To install the Perl module, follow these steps:

1. Download the latest version of the Perl module from:

<http://perl.apache.org/dist/>

You must get the 2.0 or later version of `mod_perl`. The older versions (1.x) are not compatible with Apache 2.0. In our case, the package was named *mod_perl-2.0-current.tar.gz*.

2. Extract the `mod_perl` source with the following command:

```
tar -xzf mod_perl-2.0-current.tar.gz
```

Note: You can still run only one Apache server at a time because standard HTTP and HTTP Secured (HTTPS) ports are required by Apache. If you want both versions running at the same time, configure one Apache server to use non-standard ports.

3. Go to the source directory. Enter the following command:

```
cd mod_perl-1.99_07/
```

4. Configure `mod_perl` with:

```
perl Makefile.PL  
MP_AP_PREFIX=/usr/local/apache2/MP_INST_APACHE2=1
```

We point `mod_perl` to our Apache 2.0 directory and tell it to ignore the already installed version 1.3 of Apache and its modules.

5. Compile `mod_perl` using the **make** command.
6. Test the newly created module with the following command:

```
make test
```

7. If the test completed with no errors, you can install `mod_perl` with the following command:

```
make install
```

Now `mod_perl` is installed into our Apache 2.0 directory.

6.1.5 Installing the PHP module

Install the PHP module by following these steps:

1. Download the latest version of the PHP source from:

<http://www.php.net/downloads.php>

In our case, the package was called *php-4.2.3.tar.gz*.

2. Extract the PHP source code. Type the following command:

```
tar -xzf php-4.2.3.tar.gz
```

3. Go to the source directory. Type the following command:

```
cd php-4.2.3
```

4. Configure the PHP module with the command shown in Example 6-1.

Example 6-1 Configuring PHP with options

```
./configure --prefix=/usr/local/apache2/  
--with-config-file-path=/usr/local/apache2/conf/  
--enable-force-cgi-redirect  
--disable-debug  
--enable-pic  
--disable-rpath  
--enable-inline-optimization  
--with-bz2  
--with-db3  
--with-curl  
--with-dom=/usr  
--with-exec-dir=/usr/local/apache2/bin  
--with-freetype-dir=/usr  
--with-png-dir=/usr  
--with-gd  
--enable-gd-native-ttf  
--with-ttf  
--with-gdbm  
--with-gettext  
--with-ncurses  
--with-gmp  
--with-iconv  
--with-jpeg-dir=/usr  
--with-openssl  
--with-png  
--with-pspell  
--with-regex=system  
--with-xml  
--with-expat-dir=/usr  
--with-zlib --with-layout=GNU  
--enable-bcmath  
--enable-exif  
--enable-ftp  
--enable-magic-quotes  
--enable-safe-mode  
--enable-sockets  
--enable-sysvsem  
--enable-sysvshm  
--enable-discard-path  
--enable-track-vars  
--enable-trans-sid
```

```
--enable-yp
--enable-wddx
--without-oci8
--with-pear=/usr/local/apache2/pear
--with-ldap
--with-ldap-ssl
--with-kerberos=/usr/kerberos
--with-ldap
--with-mysql=/usr
--with-pgsql
--enable-memory-limit
--enable-shmop
--enable-versioning
--enable-calendar
--enable-dbx
--enable-dio
--enable-mcal
--with-apxs2=/usr/local/apache2/bin/apxs
```

We used several options. You can remove or add options to suit your environment. Some of these options also require that you install the level RPM package or packages for that option. If you see errors in the configuration process, either install the necessary development package or remove the option.

5. Compile the PHP module with the `make` command.
6. If the compilation completed with no errors, install the PHP module. Use the following command:

```
make install
```

Our Apache installation is complete now. The next task is to configure Apache and test it.

6.1.6 Configuring and testing Apache

In the `/usr/local/apache2` directory, you see the complete Apache 2.0 environment. We call this directory the *Apache 2.0 root*. In this section, all references to files and directories are against this root unless explicitly mentioned. Let us start with the configuration:

1. Go to the `conf/` directory and open the `httpd.conf` file in an editor.
2. Scroll down until you come to a set of lines all beginning with `LoadModule`. At the end of this section, add the following line:

```
LoadModule perl_module modules/mod_perl.so
```

This loads the Perl module.

3. Verify that the following line is also present in that section:

```
LoadModule php4_module modules/libphp4.so
```

In our case, it was present. This loads the PHP module. The SSL module, as part of the basic Apache 2.0 distribution, is already installed and configured.

4. Scroll further down until you see the code in Example 6-2.

Example 6-2 Making Apache recognize modules

```
###Section 3:Virtual Hosts . Edit this section so that it looks like the following.
#Bring in additional module-specific configurations
<IfModule mod_ssl.c>
```

```

Include conf/ssl.conf
</IfModule>
# mod_perl configuration
PerlModule Apache2
Alias /perl//usr/local/apache2/perl/
<Location /perl/>
SetHandler perl-script
PerlResponseHandler ModPerl::Registry
PerlOptions +ParseHeaders
Options +ExecCGI
</Location>
#php configuration
AddType application/x-httpd-php .php .php4 .php3 .phtml
AddType application/x-httpd-php-source .phps
###Section 3:Virtual Hosts

```

-
5. Save the changes you made and close the httpd.conf file.
 6. Copy the **php.ini** file (recommended) from the source directory of PHP (where you performed configure, make, etc.) and copy it into the conf/ directory with the name php.ini. In our case, it looked like this:

```
cp /root/apache/php-4.2.3/php.ini-recommended /usr/local/apache2/conf/php.ini
```

7. If you want to use the SSL module with Apache 2.0, generate a key for use in the encryption process. In you do not intend to use SSL, you can skip to the next step. The easiest way to get this key is to copy one from the Apache 1.3 distribution. If you want a real key, follow the instructions in the SSL module documentation. To copy the Apache 1.3 key, we did this:

```
[root@portall root] #cp -a /etc/httpd/conf/ssl.crt/ /usr/local/apache2/conf/
[root@portall root] #cp -a /etc/httpd/conf/ssl.key/ /usr/local/apache2/conf/
```

Your Apache 2.0 configuration is now complete. The next step is to create two small test scripts for mod_perl and PHP and test everything:

1. In the Apache 2.0 root, create a directory perl/:

```
mkdir perl
```
2. In this directory, create a file called *test.pl*. It should have the same content as in [ED REF].
3. In the directory htdocs/, create a file called *test.php* and add the following line to it:

```
<?php phpinfo();?>
```
4. Start Apache 2.0 with the following command:

```
./bin/apachectl startssl
```

Remember that you should do this in the Apache 2.0 root. If you do not want to use SSL in the command, replace *startssl* with *start*.
5. Open a browser and type the name of your server in the Web address bar. You should see a test page for Apache.
6. In the Web address field of your browser, replace *http://* with *https://* to test SSL. You should see the same page again, but this time it is encrypted as indicated by the dialogs that may appear and the closed lock in the status bar of your browser.
7. Add *test.php* or */test.php* after the URL. Now you should see an information page from PHP. The version of PHP is indicated at the top.
8. In the URL, replace *test.php* with *perl/test.pl*. You now should see a page with the message It worked!!!.

If you can see every test page, then your Apache 2.0 setup is configured and working correctly.

6.1.7 Load balancing and Linux Virtual Server (LVS)

The purpose of blade-based server technology is to allow several servers to work together. To do this, the Web application needs a mechanism to balance the Web request load among multiple Web servers.

In this chapter, we build a Web cluster using a set of blade servers in our IBM *@server* BladeCenter. The clustering is done using two open source projects: LVS and Keepalived.

We build the Web cluster in two ways. The first way is without high availability. The second way provides high availability and uses more nodes and services.

We use the LVS project on the Directors to provide the load balancing. We use LVS to create two virtual IP addresses. The virtual IP address 10.10.10.10 is created on the public local area network (LAN) and provides access to the Web cluster for the clients. The other virtual IP address is 192.168.100.1, It provides access for Realservers to the outside world. Therefore, it is created on the private LAN.

Realservers

Realserver is the name given to computers in an LVS cluster that actually performs the services that will be virtualized. Realservers are grouped by service that is virtualized to a virtual server. A Realserver can be part of different groups at the same time and, therefore, can have different services running at the same time.

There is no need to install any software on the Realserver specific to LVS. The LVS code is only needed on IBM Director. Still, you may need to install cluster-related software on Realservers. For example, on an LVS for Web server, you may install or configure a Network File System such as Coda, Intermezzo, etc. to share the document root for the Web sites.

Be careful with the Realservers when you are clustering services that use sessions. When a client connects for a second time to the cluster, it may be directed to another Realserver other than the first Realserver. You have to make sure that any information the first Realserver has is also known by the second Realserver. You can, for example, use a common database that contains this information. LVS itself also has provisions for this. You can set a persistence time for each virtual server. If you have this enabled, then LVS connects the same client to the same Realserver for as long as the persistence timer keeps running.

The Realserver can be any operating system that has decent TCP/IP support. This depends on the way the virtual server is created. See the LVS documentation to see which LVS method is the best for your application and which operating system is supported by that method.

The Directors

The Directors with the LVS software provide the routing, filtering, and packet forwarding functions to create the virtual servers. These functions can run on any machine that is at least an Intel processor 486 and preferably with two Ethernet adapters. The Linux installed on it should be at least one with a kernel 2.2.14 or greater. The latest kernel from the 2.4 series is preferred. There are three ways to set up a virtual server:

- ▶ **Network Address Translation (NAT):** The public and private LANs are different network and IP packages that go through the Director and are rewritten so that it seems as if they came from the Director itself.

- ▶ **IP tunneling:** The Director delivers IP packages from clients to Realservers using an IP tunnel. The Realserver then answers directly to the client.
- ▶ **Direct routing:** The Director and the Realserver are all on the same physical network. This network is used by the Director to forward packages from the clients to the Realservers. The Realserver then answers the clients using another network.

In our setup, we use NAT. LVS has different algorithms to perform load balancing. The best choice depends on which service the virtual server delivers and how it is used. You can find the complete list of algorithms and explanations on the Web at:

<http://www.linuxvirtualserver.org/docs/scheduling.html>

We are also going to implement high availability into our virtual server. This means two things. Realservers that are no longer responding will be removed from the virtual server. And, we will provide failover between the two Directors if the active Director goes down. We will implement this using the Keepalived daemon. It provides health checks for Realservers and removes a Realserver from a virtual server if it stops responding. It also provides failover for the Director using Virtual Routing Redundancy Protocol (VRRP) Version 2, which is a protocol used for router failover.

6.1.8 Installing the Web cluster

In this section, we install all the software needed for the Web cluster.

Installing the Directors

In our Web cluster, we use Red Hat Linux Advanced Server 2.1 for the Directors. When configuring the network, make sure you reserve three IP addresses, one for each Director and one for the public virtual IP address. After you install Red Hat Linux on the Directors, apply all updates for Red Hat Linux. Then install the kernel-source package for the kernel that you will run. We will need this source later.

Installing LVS

The kernels from Red Hat already include the LVS code. But this is not the latest version of the LVS code. If you prefer, you can build a new kernel with the latest LVS code. We do not do that in our setup.

You need to install the ipvsadm code. This is a small tool that allows you to configure LVS and view its status. In our setup, we worked with kernel version 2.4.18-18.7.x. This kernel includes version 1.0.4 of LVS. Follow these steps:

1. Download the ipvsadm source RPM from:

<http://www.linuxvirtualserver.org/software/index.html>

For our kernel and version of LVS, we downloaded the package ipvsadm-1.21-3.src.rpm.

2. Compile this source RPM using the following command:

```
rpm --rebuild ipvsadm-1.21-3.src.rpm
```

3. Install the compiled RPM using the following command:

```
rpm -ivh /usr/src/redhat/RPMS/i386/ipvsadm-1.21-3.i386.rpm
```

Installing Keepalived

Now that LVS is installed, we continue by installing Keepalived:

1. Download the latest Keepalived source RPM from:

<http://www.Keepalived.org/download.html>

We downloaded Keepalived-0.7.6-1.src.rpm.

2. Create a symbolic link called `/usr/src/linux` in the kernel source code. We used the following command:

```
cd /usr/src;ln -s linux-2.4.18-18.7.x/linux;cd
```

This link is needed so Keepalived can find the source code of LVS.

3. Compile the Keepalived source RPM using the following command:

```
rpm --rebuild Keepalived-0.7.6-1.src.rpm
```

4. Install the compiled RPM using the following command:

```
rpm -ivh /usr/src/redhat/RPMS/i386/Keepalived-0.7.6-1.i386.rpm
```

The Directors are now ready.

Note: The compilation of the source RPM of ipvsadm and Keepalived has to be done only once. You can use the compiled packages directly on the other Director.

6.1.9 Configuring the Web cluster

We set up the cluster in two steps. In the first step, we use only one Director. This means no Director failover and two Realservers. With this, you can test the basic functioning of the cluster. Then we set up the cluster using both Directors and all the Realservers. We also add support for both the HTTP and HTTPS protocols.

Configuring the basic Web cluster

We configure Director 1 to create the virtual server and use Realservers 1 and 2 as the servers part of the virtual server:

1. Make sure that you set the network configuration of all the blade servers as indicated in the previous sections.

2. For LVS to work, enable the forwarding of IP packets by the kernel. Type the following command:

```
echo "1">/proc/sys/net/ipv4/ip_forward
```

3. To enable IP forwarding by default, edit the file `/etc/sysctl.conf` and set the line `net.ipv4.ip_forward=0` to `=1`.

4. Open the `/etc/Keepalived/Keepalived.conf` file in an editor. Edit the files to look like the one shown in Example 6-3.

Example 6-3 Configuration of Keepalived for the basic Web cluster

```
!Configuration File for Keepalived
global_defs {
!who will get alert emails
notification_email {
    root@localhost
}
```

```

!the alert emails will come from
notification_email_from Keepalived@localhost
!use this server to send the alert mails
smtp_server 127.0.0.1
smtp_connect_timeout 30
!the name/id of this load balancer
!(it should be unique)
lvs_id LVS_DIRECTOR_1
!this defines the VIP on the public LAN,the virtual server
vrrp_instance VI_1 {
!this director will start as master
state MASTER
!create the VIP on interface eth1
interface eth1
!each virtual server needs a unique id
virtual_router_id 51
!the priority determines who is master in failover setups,
!the highest priority becomes master
priority 200
!how often check for failover between master and backup
advert_int 1
!authentication for the synchronisation between master and backup
authentication {
auth_type PASS
auth_pass 1111
}
!the IP addresses that are part of this virtual server
virtual_ipaddress {
10.10.10.10
}
!this defines the gateway on the private LAN
!setup is the same as above,only different interface (eth0)
!different id (52)and different IP address
vrrp_instance VI_GATEWAY {
state MASTER
interface eth0
virtual_router_id 52
priority 200
advert_int 1
authentication {
auth_type PASS
auth_pass 1111
}
virtual_ipaddress {
192.168.100.1
}
!this links the VIPs on the public and private LAN so that they both are
!taken over by the backup in case of failover
vrrp_sync_group VSG_1 {
group {
VI_1
VI_GATEWAY
}
}
!this defines the IP address 10.10.10.10 if our virtual server
!for port 80,the http port
virtual_server 10.10.10.10 80 {
!time between the health checks of the realservers
delay_loop 6
!load balancing algorithm (rr =round robin)
lb_algo rr

```

```

!type of LVS (NAT =network address translation)
lb_kind NAT
!protocol of the virtual IP
protocol TCP
!realserver 1 on port 80
real_server 192.168.100.10 80 {
    !weight is used in weighted algorithms
    weight 1
    !type of health check,get a html page
    HTTP_GET {
        !which url to test
        url {
            path /
            !checksum of the recieved page to test with
            !you need to set this for your own environment
            digest ff20ad2481f97b1754ef3e12ecd3a9cc
        }
        !on what port must we test
        connect_port 80
        !timeout value for the test
        connect_timeout 3
        !how many retries before marking server dead
        nb_get_retry 3
        !how long to wait between retries
        delay_before_retry 3
    }
}
!realserver 2,similar setup as realserver 1
real_server 192.168.100.11 80 {
    weight 1
    HTTP_GET {
        url {
            path /
            digest ff20ad2481f97b1754ef3e12ecd3a9cc
        }
        connect_port 80
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
    }
}

```

5. Modify the digest values for each Realserver. You can find the digest value using the **genhash** command. In our example, we created an index page that showed us the name of the Realserver with a different background each time. This helps us to see if everything is working correctly. We used the following steps to create the digest values for Realserver 1:

a. Run the following command:

```
genhash -s 192.168.100.10 -p 80 -u /
```

At the end of the output from this command, we saw the output shown in Example 6-4.

Example 6-4 Output of the genhash command

```
-----[ HTML MD5 resulting ]-----  
0000 ff 20 ad 24 81 f9 7b 17 -54 ef 3e 12 ec d3 a9 cc ..$.>.....  
-----[ HTML MD5 final resulting ]-----  
ff20ad2481f97b1754ef3e12ecd3a9cc
```

- b. The last line is the value to be filled in at the digest parameter. Do this for all Realservers that you defined. If you have the same index page on each Realserver, then you need only to run **genhash** once and fill in the same value for each Realserver.
6. Save the Keepalived configuration file and close the editor.
7. Make sure all the Apache Web servers on the Realservers are started.
8. To help in the initial debugging, we let Keepalived output extra information into the system log. Edit the Keepalived start script `/etc/init.d/Keepalived.init` and change Keepalived to `Keepalived -d` in the start and restart sections.
9. Start Keepalived with the following command:

```
/etc/init.d/Keepalived.init start
```
10. Watch the system log file with `less /var/log/messages`. At the end, you should see a lot of information from Keepalived.
11. With `ipvsadm`, check whether Keepalived set up LVS properly.
12. Using `ip addr list`, you should see the virtual IP addresses added to the Director.
13. If all these commands give you the correct information, you can test the virtual server. On a client, browse to the virtual IP address 10.10.10.10 and reload it a couple of times. You should see all your Apache servers working.

Your basic cluster should work now. You can check everything by using the commands mentioned above and looking at the log files on the Director and Realservers.

Configuring the complete Web cluster

Now that we have the basic clustering setup, we expand the configuration to use all six of our blade servers. This means adding a second Director that provides failover for the first Director, adding two Realservers to our virtual server, and creating a virtual server for HTTPS support.

The configuration of the three new blade servers is identical to that of the configuration we already used. Only the network configuration, the index page on the Apache server, and the `Keepalived.conf` file on the second Director are different.

Follow these steps:

1. Make sure the Realservers are running and have Apache set up for both HTTP and HTTPS support.
2. Edit the `Keepalived.conf` file on Director 1. This is the master. Make it look like the file shown in Example 6-5.

Example 6-5 Configuration for Keepalived

```
!Configuration File for Keepalived  
global_defs {  
    !who will get alert emails  
    notification_email {
```

```

    root@localhost
}
!the alert emails will come from
notification_email_from Keepalived@localhost
!use this server to send the alert mails
smtp_server 127.0.0.1
smtp_connect_timeout 30
!the name/id of this load balancer
!(it should be unique)
lvs_id LVS_DIRECTOR_1
!this defines the VIP on the public LAN,the virtual server
vrrp_instance VI_1 {
    !this director will start as master
    state MASTER
    !create the VIP on interface eth1
    interface eth1

```

Note: The client cannot be the Director or any of the Realservers. It needs to be a separate machine connected to the public LAN.

```

    !this enabled a daemon,part of LVS,that synchronises both director
    lvs_sync_daemon_interface eth1
    !each virtual server needs a unique id
    virtual_router_id 51
    !the priority determines who is master in failover setups,
    !the highest priority becomes master
    priority 200
    !how often check for failover between master and backup
    advert_int 1
    !authentication for the synchronisation between master and backup
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    !the IP adresses that are part of this virtual server
    virtual_ipaddress {
        10.10.10.10
    }
!this defines the gateway on the private LAN
!setup is the same as above,only different interface (eth1)
!different id (52)and different IP address
vrrp_instance VI_GATEWAY {
    state MASTER
    interface eth0
    lvs_sync_daemon_interface eth0
    virtual_router_id 52
    priority 200
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.100.1
    }
}
!this links the VIPs on the public and private LAN so that they both are
!taken over by the backup in case of failover
vrrp_sync_group VSG_1 {
    group {
        VI_1

```

```

    VI_GATEWAY
}
!this defines the IP address 10.10.10.10 if our virtual server
!for port 80,the http port
virtual_server 10.10.10.10 80 {
    !time between the health checks of the realservers
    delay_loop 6
    !load balancing algorithm (rr =round robin)
    lb_algo rr
    !type of LVS (NAT =network address translation)
    lb_kind NAT
    !protocol of the virtual IP
    protocol TCP
    !realserver 1 on port 80
    real_server 192.168.100.10 80 {
        !weight is used in weighted algorithms
        weight 1
        !type of health check,get a html page
        HTTP_GET {
            !which url to test
            url {
                path /
                !checksum of the recieved page to test with
                !you need to set this for your own environment
                digest ff20ad2481f97b1754ef3e12ecd3a9cc
            }
            !on what port must we test
            connect_port 80
            !timeout value for the test
            connect_timeout 3
            !how many retries before marking server dead
            nb_get_retry 3
            !how long to wait between retries
            delay_before_retry 3
        }
    }
}
!realserver 2,similar setup as realserver 1
real_server 192.168.100.11 80 {
    weight 1
    HTTP_GET {
        url {
            path /
            digest ff20ad2481f97b1754ef3e12ecd3a9cc
        }
        connect_port 80
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
    }
}
real_server 192.168.100.12 80 {
    weight 1
    HTTP_GET {
        url {
            path /
            digest ff20ad2481f97b1754ef3e12ecd3a9cc
        }
        connect_port 80
        connect_timeout 3
        nb_get_retry 3
    }
}

```

```

        delay_before_retry 3
    }
}
real_server 192.168.100.13 80 {
    weight 1
    HTTP_GET {
        url {
            path /
            digest ff20ad2481f97b1754ef3e12ecd3a9cc
        }
        connect_port 80
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
    }
}
!same virtual server but now using port 443 =HTTPS
virtual_server 10.10.10.10 443 {
    delay_loop 6
    lb_algo rr
    lb_kind NAT
    protocol TCP
    !this means that a client will be routed to the same virtual server
    !for a period of 10 minutes.Needed for SSL connections,adjust for
    !your own setup
    persistence_timeout 600
    real_server 192.168.100.10 443 {
        weight 1
        !use HTTPS instead of HTTP
        SSL_GET {
            url {
                path /
                digest ff20ad2481f97b1754ef3e12ecd3a9cc
            }
            connect_port 443
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
        }
    }
    real_server 192.168.100.11 443 {
        weight 1
        SSL_GET {
            url {
                path /
                digest ff20ad2481f97b1754ef3e12ecd3a9cc
            }
            connect_port 443
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
        }
    }
    real_server 192.168.100.12 443 {
        weight 1
        SSL_GET {
            url {
                path /
                digest ff20ad2481f97b1754ef3e12ecd3a9cc
            }
            connect_port 443

```

```

        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
    }
}
real_server 192.168.100.13 443 {
    weight 1
    SSL_GET {
        url {
            path /
            digest ff20ad2481f97b1754ef3e12ecd3a9cc
        }
        connect_port 443
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
    }
}
}

```

Here again, adjust the digest value for each HTTP_GET or SSL_GET test to the value that corresponds to your setup.

3. Copy the Keepalived.conf file from Director 1 to Director 2.
4. Edit the Keepalived.conf file on Director 2 and make the changes listed in Table 6-1. You must change the state and priority statements twice, once for each virtual IP address.

Table 6-1 Changes needed for Keepalived.conf file

Change	To
lvs_id LVS_DIRECTOR_1	lvs_id LVS_DIRECTOR_2
state MASTER	state BACKUP
priority 200	priority 100

5. Enable the forwarding of IP packets on Director 2. You can do this by entering:


```
echo "1">/proc/sys/net/ipv4/ip_forward
```
6. To enable IP forwarding by default, edit the file /etc/sysctl.conf and set the line net.ipv4.ip_forward =0 to =1.
7. Start Keepalived on Director 1. Type the following command:


```
/etc/init.d/Keepalived.init start
```
8. Confirm that Keepalived has started correctly by checking /var/log/messages, ipvsadm, and ip addr list.
9. Test the cluster using a browser. Remember, the client must not be the Realservers or the Director nor on the public LAN. Also test both HTTP and HTTPS.
10. If the cluster works, start Keepalived on Director 2. Type the following command:


```
/etc/init.d/Keepalived.init start
```
11. Again, check that Keepalived has started correctly by looking at /var/log/messages, ipvsadm, and ip addr list. You should see in /var/log/messages that it is in backup state.
12. Test the cluster again from a client.
13. Stop Apache on one of the Realservers using the command:


```
service httpd stop
```

In the logs, you should see that Keepalived has removed it from the virtual server. You should not be able to access it from the client.

14. Shut down Director 1 to test failover of the Directors using the following command:

```
shutdown -h now
```

15. Keep testing the cluster with the clients. Director 2 should take over the load balancing.

If everything works, you now have a high availability Web cluster.

6.2 E-mail

E-mail is one Internet application that people and organizations can no longer do without. It is quickly replacing the telephone as the primary means of business communication. Some even argue that Internet based e-mail is as critical to our economy as the road system. Open source mail transfer agents (MTAs) handle much of the e-mail traffic flowing through the Internet today.

This section shows you how to set up an e-mail system using open source components.

6.2.1 How Internet e-mail systems fit together

E-mail essentially involves moving messages from one system to another and storing those messages until they are ready to be accessed by their users. While the concept is simple, making sure that it actually works is not always so easy. There are different types of technology involved in moving a message from one part of the world to another.

Mail transfer agents

An MTA is a program that transports mail between servers. You can think of this as the equivalent of a post office on the Internet.

MTAs ensure that the messages are properly formatted and sent to their destination. When a message comes through an MTA, it identifies the sender and recipient from the message's headers. Then it passes them on for delivery to another MTA or to a mail delivery agent (MDA), which manages mail locally.

MTAs make routing decisions about the messages that go through them:

- ▶ If a message recipient matches a local mailbox, the message is passed to that mailbox.
- ▶ If a message is invalid, an error message is returned to the sender.
- ▶ If a message is valid but not local, the domain name is used to decide which servers accept mail for that address, according to DNS records.

The following sections highlight some MTAs that are worth noting.

Sendmail

Sendmail is by far the most widely used MTA today. It services approximately 80% of the e-mail traffic on the Internet. All UNIX and Linux distributors package a version of Sendmail to meet basic e-mail needs.

Sendmail has a long history with the Internet and has grown and evolved over the years. It has also taken on some complexity to configure, if only because of the sheer number of options available to it.

Sendmail is currently distributed under the Sendmail license. While it is still open source, it's important to note that Sendmail is currently under the ownership of Sendmail, Inc., a company formed by Eric Allman, the lead developer. You can find the Sendmail license on the Web at:

<http://www.sendmail.org/license-info.html>

Open source Sendmail's Web site is at:

<http://www.sendmail.org>

You can find Sendmail, Inc. on the Web at:

<http://www.sendmail.com>

Sendmail, Inc. sells enhanced commercial versions of the Sendmail MTA.

Postfix

Postfix was one of the first alternatives to Sendmail as an MTA. Postfix was written from the ground up by Wietse Venema while on a research fellowship to IBM. It was first known as Vmailer, and then as the IBM Secure Mailer, before finally being called Postfix.

Design wise, Postfix attempts to be fast, easy to administer, and secure, while being compatible with Sendmail. Externally, it works just like Sendmail, making use of the same transport mechanisms. Internally, it is completely different.

You can find Postfix on the Web at:

<http://www.postfix.org>

qmail

Another popular replacement for Sendmail is qmail. Also designed from the ground up by Daniel J. Bernstein, qmail is smaller than either Postfix or Sendmail. qmail is known for introducing the *maildir* format, which is important for supporting scalable mail systems.

To learn more about qmail, go to the Web site at:

<http://www.qmail.org>

Domain Name Services

The Internet is a worldwide communications network with hundreds of thousands of servers. To reliably send a message from one server to another, e-mail systems have to work hand-in-hand with the Domain Name System, the global naming service.

DNS tells an MTA which server to send a message to, based on the domain part of the e-mail address. It does this by looking up mail exchanger (MX) type records from DNS.

Just because a server is listed as a destination by the DNS system does not mean that it is the final destination. It could still relay messages to another e-mail server that only it knows about.

Message access

What happens after the message is received on the server? The message must then be retrieved by the user. The class of applications that does this is known as *Mail Retrieval Agents* (MRAs).

There are two ways to do this as explained in the following sections.

- ▶ **Internet Message Access Protocol (IMAP):** The IMAP is a powerful protocol. It allows all sorts of devices to access and manipulate server-based mail from any location, from multiple mailboxes. A device with limited local storage capacity can simply view message headers to avoid downloading huge binary attachments that it are unable to display. For example, e-mail stored on an IMAP server can be manipulated from a desktop computer at home, a workstation at the office, and a notebook computer while traveling, without needing to transfer messages or files back and forth between these computers.
- ▶ **Post Office Protocol (POP3):** POP3 uses a tiny subset of IMAP functions to download e-mail from servers to client programs. However, it is limited to retaining or deleting the messages held on the remote server. POP is favored by many ISPs because it is simpler to configure and administer than IMAP.

These days, IMAP and POP3 are usually implemented on the same daemon. Some IMAP/POP3 daemon implementations include:

- ▶ The University of Washington IMAP (UW-IMAP) server is a basic IMAP/POP3 server included in several UNIX and Linux distributions. Its authentication system and mailbox system are tied very closely with UNIX/Linux user accounts.

You can learn more about the UW-IMAP server on the Web at:

<http://www.washington.edu/imap/>

- ▶ *Courier* is an interesting project because it actually includes almost everything you need to run a mail system, including an Simple Mail Transfer Protocol (SMTP) MTA. At the same time, it provides IMAP, POP3, Webmail, and mailing list services.

Courier can function as an intermediate mail relay, relaying mail between an internal LAN and the Internet. Or it can perform final delivery to mailboxes. Courier uses maildirs as its native mail storage format, but it can also deliver mail to legacy mailbox files.

Courier can provide mail services for regular operating system accounts and for virtual mail accounts, managed by an Lightweight Directory Access Protocol (LDAP), MySQL, or PostgreSQL-based authentication database.

You can learn more about Courier on the Web at:

<http://www.courier-mta.org/>

- ▶ *Cyrus IMAP* is another implementation of the IMAP/POP3 protocol, this time from Carnegie Mellon University. Cyrus IMAP is on the Web at:

<http://asg.web.cmu.edu/cyrus/>

6.2.2 Building an e-mail server with Sendmail and UW-IMAP

Out of the box, Red Hat comes with the necessary software components to build a basic e-mail server that is accessible by e-mail clients like Microsoft® Outlook, Eudora, or Mozilla. Let's look at an implementation using the stock packages Sendmail and UW-IMAP.

DNS prerequisites

To receive e-mail from across the Internet, your mail server must be properly recognized as a destination by another mail server. This information may either be hard coded in that server (in the case of a mail relay routing mail to your server), or encoded in the DNS system, if your mail server is set to receive e-mail from the world at large.

The following types of records must exist in your DNS for your domain:

```
mail.lsp1.ibm.com.      IN A 192.168.100.58
lsp1.ibm.com.          IN MX 10 mail.lsp1.ibm.com.
```

Installing and activating Sendmail

Sendmail is installed by default in a standard Red Hat Advanced Server installation. However, because the Sendmail package that comes with the installation CD has security vulnerabilities, you need to update it with the latest packages from Red Hat. At the time of this writing, the version of Sendmail we used is Sendmail-8.11.6-26.72.

To update Sendmail with the new RPM packages, enter the following commands:

```
# rpm -U sendmail-8.11.6-26.72.i386.rpm
# rpm -U sendmail-cf-8.11.6-26.72.i386.rpm
```

Then restart the Sendmail daemon:

```
# service sendmail restart
```

Installing and activating UW-IMAP

Unlike Sendmail, the IMAP daemon is not installed by default. You install it yourself, using the command:

```
# rpm -ivh imap-2001a-10.0as.i386.rpm
```

The IMAP server is controlled by the *xinetd daemon*. *xinetd* is a replacement for *inetd*, the Internet services daemon. It is not compatible with *inetd*, but is preferred for a variety of reasons. *xinetd* listens for requests on several common TCP/IP ports and invokes the proper daemon whenever it receives a request. See the following Web site for more information:

<http://www.xinetd.org/faq.html>

By default, the mail services are disabled. To enable IMAP and POP3 services at startup time, execute the following commands:

```
# chkconfig --level 345 imap on
# chkconfig --level 345 ipop3 on
```

Now, every time you restart the server, the IMAP and POP3 daemons should be activated along with *xinetd*. For the very first time you do this, you may want to restart the *xinetd* daemon:

```
# service xinetd restart
```

Configuring Sendmail

Sendmail comes with hundreds of options. It's very configurable, but also very complex. For starters, let's look at the default Sendmail configuration that we installed from the stock RPMs.

The main configuration file for Sendmail is */etc/sendmail.cf*. This file is very cryptic and difficult to debug. You normally do not need to modify this file directly. Instead, you modify the slightly more readable */etc/mail/sendmail.mc* file and generate the *sendmail.cf* file from it.

Rather than take you through the configuration options line by line, we simply modify the lines that we need and explain those changes as we go along.

Starting with Red Hat 7.x, the default Sendmail configuration is to only send e-mail. The ability to receive e-mail has been disabled. Keeping the receive capability off, by default, prevents servers from being inadvertently activated on the Internet and used by external parties for relaying spam.

Since we are building a mail server, we must reconfigure Sendmail to receive e-mail. There is only one line that we need to modify. The original line instructs Sendmail to only receive e-mail originating from the server on which it runs:

```
DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')
```

We need to comment out this line by placing `dn1` in front of it. Now it should read:

```
dn1 DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')
```

Edit the `sendmail.mc` file to reflect the changes above. Then, generate the new `sendmail.cf` file. You may want to back up your `sendmail.mc` file first.

```
# cp /etc/mail/sendmail.mc /etc/mail/sendmail.mc.bak
# m4 /etc/mail/sendmail.mc > /etc/sendmail.cf
```

Now, restart Sendmail:

```
# service sendmail restart
```

This configuration should be enough for now for some of the basic work that we will do.

If you want to test sending mail to the server, follow the command sequence shown in Example 6-6.

Example 6-6 Testing the mail server

```
[user@blade2 user]$ mail -v user@blade1.lspl.ibm.com
Subject: Hello, world!
My first e-mail message.
Cc:
user@blade1.lspl.ibm.com... Connecting to blade1.lspl.ibm.com. via esmtp...
```

On the mail server, you can test if the e-mail was received as shown in Example 6-7.

Example 6-7 Checking if mail was received

```
[user@blade1 user]$ mail
Mail version 8.1 6/6/93. Type ? for help.
"/var/spool/mail/user": 1 message 1 new
>N 1 user@blade2.lspl.ibm Mon Apr 28 14:53 16/683 "Hello, world!"
& 1
Message 1:
From user@blade2.lspl.ibm.com Mon Apr 28 14:53:32 2003
Date: Mon, 28 Apr 2003 14:53:20 -0700
From: Dominique Cimafranca <user@blade2.lspl.ibm.com>
To: user@blade1.lspl.ibm.com
Subject: Hello, world!
My first e-mail message.
& d
& q
```

Configuring IMAP and POP3

No configuration needs to be done for the IMAP and POP3 server. The IMAP server that comes with Red Hat is ready for use.

Test whether your IMAP and POP3 server is working as shown in Example 6-8.

Example 6-8 Testing POP3 and IMAP

```
[user@blade1 user]$ telnet 127.0.0.1 110
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^'.
+OK POP3 localhost.localdomain v2001.78rh server ready
quit
+OK Sayonara
Connection closed by foreign host.
[user@blade1 user]$ telnet 127.0.0.1 143
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^'.
* OK [CAPABILITY IMAP4REV1 LOGIN-REFERRALS STARTTLS AUTH=LOGIN] localhost.localdomain
IMAP4rev1 2001.315rh at Mon, 28 Apr 2003 15:11:34 -0700 (PDT)
. logout
* BYE blade1.lspl.ibm.com IMAP4rev1 server terminating connection
. OK LOGOUT completed
Connection closed by foreign host.
```

Configure your IMAP or POP3 client to connect to the server. This should be trivial, since most modern e-mail clients already include wizards. Figure 6-1 shows the E-mail Server Names panel of the Internet Connection wizard. This example illustrates how to set up an e-mail client. You can access this window through the Microsoft Windows control panel, Microsoft Internet Explorer, Outlook Express, or any other Windows application that uses the wizard to configure Internet setting.

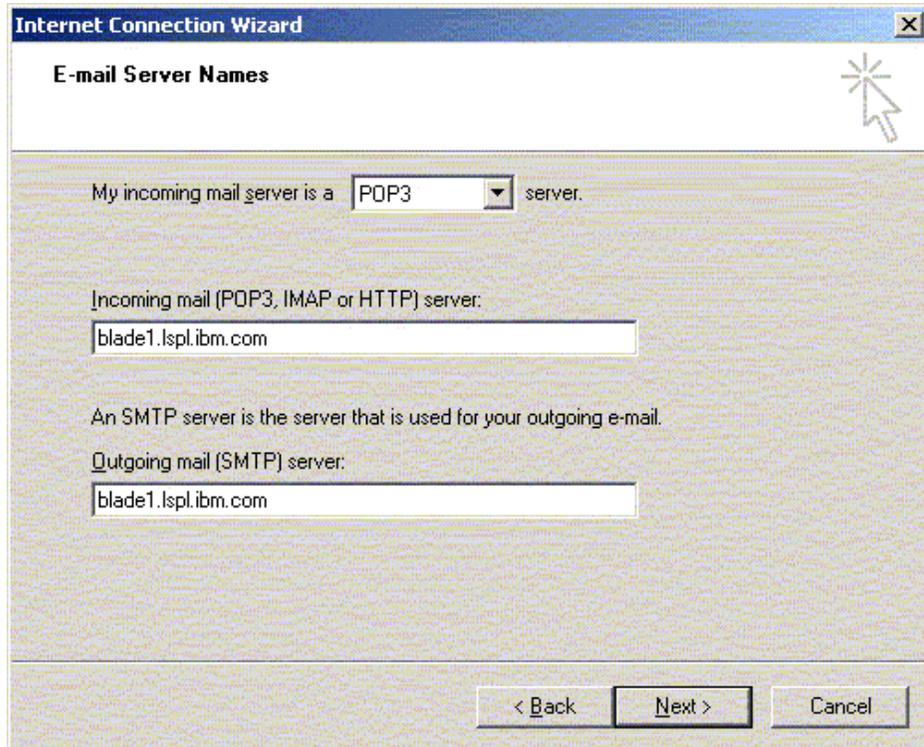


Figure 6-1 Setting up an e-mail client

Adding user accounts

With the IMAP daemon that ships with Red Hat, each e-mail account requires a corresponding UNIX account on the server.

To add a user to the e-mail server, you also need to create a UNIX account. The safe way to do this is with a command like the one shown in the following example:

```
# adduser newuser -s /bin/false
```

We add the `-s` flag to specify a `/bin/false` shell for the mail user. This prevents the user from logging into the server.

Pros and cons of the UW-IMAP approach

There are pros and cons of using the stock IMAP server that ships with Red Hat.

On the positive side, consider these points:

- ▶ It's very easy to install and set up.
- ▶ It works fairly well in an organization with only a small number of e-mail users.
- ▶ User accounts on the server are also e-mail accounts, so you can create e-mail users with standard Linux tools.

On the down side, consider these points:

- ▶ It has very limited growth potential because it is limited to only one server. On a modern Linux server, it can serve perhaps up to 500 users.
- ▶ In regard to management of users, essentially you're using UNIX tools for managing users. This may not be adequate for a large number of users. Before you experience

performance problems at 500 users, you will most likely encounter management problems at 100 users.

- ▶ User accounts on the server are also e-mail accounts. This may raise some security concerns as to what privileges are given to users.
- ▶ UW-IMAP uses the mbox format. That is, it places all incoming e-mail into a single file. This is adequate for single server configurations, but not if you're placing the mail store into a shared file system for high availability.

The bottom line is to use the Sendmail and IMAP packages that ship with Red Hat only for a small organization, typically under a hundred users.

6.2.3 Replacing Sendmail with Postfix

We've shown you how to set up a small mail server quickly and with little hassle by using the stock packages that ship with the Red Hat distribution. We've also discussed the drawbacks of our previous approach.

Let's now take an alternative path and replace Sendmail and UW-IMAP with other components. We start with Postfix.

Getting Postfix

You can find the Postfix source on the Web at:

<http://www.postfix.org>

Pre-packaged RPMs are also available, courtesy of Simon J. Mudd, on the Web at:

<http://postfix.w10.org/>

For our main installation, we use the pre-packaged RPM. postfix-2.0.9-1.rh73.rpm works with Red Hat Advanced Server 2.1. Look for it in the Red Hat 7.3 branch of the Web site.

Removing Sendmail

Some files of Sendmail conflict with Postfix. Therefore, you have to remove Sendmail before you install Postfix.

If you've been running your Sendmail in production, tell the server to stop receiving e-mail and flush the queue. To do that, run the following commands:

```
# service sendmail stop
# sendmail -q30m
```

This tells Sendmail to stop listening to port 25 and flush the remaining messages every 30 minutes.

Now remove Sendmail:

```
# rpm -e sendmail
```

If you're working with versions of Red Hat other than Advanced Server, you may see a message indicating that some packages such as fetchmail and mutt will break. If you're not using these packages, you can also remove fetchmail and mutt.

Prior to removing Sendmail, remove these two packages.

```
# rpm -e fetchmail
# rpm -e --nodeps sendmail
```

Installing Postfix

Installation of Postfix is straightforward. Enter the following statement:

```
# rpm -ivh postfix-2.0.9-1.rh73.rpm
```

The general layout of the location of the files is:

- ▶ /etc/postfix: Configuration files
- ▶ /usr/libexec/postfix: Postfix utilities
- ▶ /var/spool/postfix: Mail processing directories
- ▶ /usr/sbin: Main Postfix executables

Configuring Postfix

The main Postfix configuration files are located in /etc/postfix/master.cf and /etc/postfix/main.cf. There are a few hundred configuration options for Postfix. In most cases, you only really need to configure at least three parameters in /etc/postfix/main.cf.

What domain to use for outbound mail

The myorigin parameter specifies the domain portion of any outgoing mail from the server. The default is to use the value of the variable \$myhostname, which defaults to the name of the server as in myorigin = \$myhostname.

You may want to change that into \$mydomain, which defaults to the parent domain of the server name, as in myorigin = \$mydomain. This option is more desirable.

What domains to receive mail for

The mydestination parameter specifies what domains this server will process locally, instead of forwarding to another machine. The default receives mail for the server itself:

```
mydestination = $myhostname localhost.$mydomain
```

You should list all the possible names that will be used to refer to this server in any e-mail message to avoid mail delivery loops, for example:

```
mydestination = $myhostname localhost.$mydomain $mydomain
```

A more complete example is shown here:

```
mydestination = $myhostname localhost.$mydomain mail.$mydomain www.$mydomain
```

What clients to relay mail for

By default, Postfix relays mail for clients in authorized networks. Authorized client networks are defined by the mynetworks parameter. The default is to authorize all clients in the IP subnet to which the local machine is attached.

If you are working in a small network, the default settings are sufficient. If you want to set it explicitly, enter:

```
mynetworks = 192.168.100.0/28, 127.0.0.0/8
```

We don't want our Postfix server being used by evil spammers, so we also add the following line:

```
relay_domains = $mydestination
```

Starting Postfix

Postfix is set up to start automatically when you reboot. To start it the first time around, use the following command:

```
# service postfix start
```

Postfix and UW-IMAP

Postfix works with the UW-IMAP package with no modifications necessary to either application. You can perform the same set of tests on this setup that you did for Sendmail.

User accounts that you create on the system are e-mail accounts.

6.2.4 Replacing UW-IMAP with Courier

Courier evolved out of several related projects pertaining to mail. Courier implements SMTP extensions for mailing list management and spam filtering. Therefore, Courier can function as an intermediate mail relay, relaying mail between an internal LAN and the Internet. Or it can perform final delivery to mailboxes.

Courier uses maildirs as its native mail storage format, but it can also deliver mail to legacy mailbox files. Courier's configuration is set by plain text files and Perl scripts. Most of Courier's configuration can be adjusted from a Web browser, using Courier's Web-based administration module.

Courier can provide mail services for regular operating system accounts. Courier can also provide mail services for virtual mail accounts, managed by an LDAP, MySQL, or PostgreSQL-based authentication database.

In the following example, we use the IMAP/POP3 portion of Courier together with Postfix.

Getting Courier

Courier is not included as part of the stock packages from Red Hat. You have to download it from the Web at:

<http://www.courier-mta.org/download.php>

Courier is not available as RPM, only as source. However, it was packaged in such a way that you can compile RPMs directly from the source code. We show you how to do this later.

Removing UW-IMAP

If you installed the UW-IMAP server, remove it to avoid conflicts in the use of the IMAP and POP3 ports. To do this, enter the following command:

```
# rpm -e imap-2001a
```

Dependencies

Courier requires the following packages:

- ▶ expect
- ▶ fam
- ▶ fam-devel
- ▶ postgresql-libs
- ▶ postgresql-devel
- ▶ mysql
- ▶ mysql-devel
- ▶ openssl-perl

Make sure that you install these packages before you proceed.

Compiling Courier

You cannot compile Courier as root. You must do it as an ordinary user. To begin, log on as an ordinary user using the `su` command:

```
# su - user
```

Before you compile Courier, create a mirror image of the main RPM directory:

```
% mkdir rpm
% mkdir rpm/SOURCES
% mkdir rpm/SPECS
% mkdir rpm/BUILD
% mkdir rpm/SRPMs
% mkdir rpm/RPMS
% mkdir rpm/RPMS/i386
% echo "%_topdir /home/user/rpm" >> /home/user/.rpmmacros
```

Then, build the RPMs:

```
% rpm -ta courier-0.42.tar.bz2
```

This creates the following files in the `rpm/RPMS/i386` directory:

- ▶ **courier-0.41.0-1.i386.rpm**: The base mail server
- ▶ **courier-smtpauth-0.41.0-1.i386.rpm**: For authenticated ESMTP in the ESMTP server
- ▶ **courier-ldap-0.41.0-1.i386.rpm**: Module for authentication against an LDAP directory
- ▶ **courier-mysql-0.41.0-1.i386.rpm**: Module for authentication against a MySQL database
- ▶ **courier-pgsql-0.41.0-1.i386.rpm**: Module for authentication against a PostgreSQL database
- ▶ **courier-fax-0.41.0-1.i386.rpm**: Module for sending e-mails by fax
- ▶ **courier-pop3d-0.41.0-1.i386.rpm**: The POP3 server
- ▶ **courier-imapd-0.41.0-1.i386.rpm**: The IMAP server
- ▶ **courier-webmail-0.41.0-1.i386.rpm**: The Webmail server
- ▶ **courier-webadmin-0.41.0-1.i386.rpm**: Web-based administration tool
- ▶ **courier-maildrop-0.41.0-1.i386.rpm**: Maildrop filter, so you can tie Courier to spam filters and antivirus programs
- ▶ **courier-mlm-0.41.0-1.i386.rpm**: Mailing list manager
- ▶ **courier-sendmail-wrapper-0.41.0-1.i386.rpm**: Soft links to the Sendmail packages, only for systems older than Red Hat 7.3.
- ▶ **courier-maildrop-wrapper-0.41.0-1.i386.rpm**: Soft links to the maildrop package, if it is installed separately.

Installing Courier IMAP and POP3

Set up IMAP and POP3 services for Courier. Enter the following command:

```
# rpm -ivh courier-0.42.0-1.i386.rpm
# rpm -ivh courier-imapd-0.42.0-1.i386.rpm
# rpm -ivh courier-pop3d-0.42.0-1.i386.rpm
```

Adding users and creating maildirs

Courier uses maildirs to store messages. Maildir uses stores messages in individual files instead of placing them all in a single file.

Maildirs versus mbox: Mbox is the traditional way to store mail on UNIX-based mail servers. Individual messages are simply concatenated together and lumped in a single file. A special marker is placed where one message ends and the next message begins. Only one process can access the mbox file in read/write mode. Concurrent access requires a locking mechanism. Anytime someone needs to update the mbox file, everyone else must wait for the update to complete. This is adequate for small organizations, but as we mentioned before, it does not scale very well.

Maildirs were originally implemented in the qmail mail server. Individual messages are saved in separate files, one file per message. There is a defined method for naming each file and a defined procedure for adding new messages to the maildir. No locking is required. Multiple processes can use maildirs at the same time.

For more information on this, see the following Web sites:

<http://www.courier-mta.org/mbox-vs-maildir/>
<http://www.washington.edu/imap/IMAP-FAQs/index.html>

As with many things on the Internet, this is a controversial issue.

After you install Courier, every time you create a user, a maildir directory is also created.

For existing users, you must create maildirs manually, as shown here:

```
# maildirmake /home/user/Maildir  
# chown -R user:user /home/user/Maildir
```

Configuring Postfix to send e-mail to maildirs

Since we are using a new format, we must tell Postfix to use maildir. Go to the `/etc/postfix/main.cf` directory and add the line:

```
home_mailbox = Maildir/
```

Then, issue the command:

```
# postfix reload
```

From here on, you can perform the same tests that you performed earlier.

Webmail with Squirrelmail

Courier includes a Webmail package called *Squirrelmail*. After compiling the Courier RPMs, one of the files you should see is `courier-webmail-0.41.0-1.i386.rpm`. Before you install this package, install the Apache Web server on the system on which you want Webmail to run. The Courier Webmail package installs itself into the default document tree of the Red Hat Apache installation.

To access the Webmail interface, point your Web browser to:

```
http://yourservername/cgi-bin/webmail
```

This should bring up the Squirrelmail Web interface.

6.2.5 Virtual users and domains with Courier and Postfix

The method in the preceding section still suffers from the need to create UNIX system accounts to have e-mail accounts. Some service providers want the ability to create virtual e-mail accounts without any corresponding UNIX system accounts.

The ideal way of doing this is to put all virtual account information into a relational database (such as MySQL or PostgreSQL) or a directory (such as OpenLDAP). This approach gives the e-mail system much more flexibility and scalability.

The steps shown in the following section encompass an intermediate step. We list virtual users into a hashed database file, without creating system users on the server itself.

Configuring Courier for virtual users and domains

Create a user that will receive e-mail for the virtual users and domains. In this example, we call this user *courier*:

```
# adduser courier
```

The plan is to put all incoming mail for the virtual users in the directory `/home/courier/lspl.ibm.com/username`, with *username* being a directory for each virtual user.

Virtual users are placed in a map file located in the directory `/etc/courier/userdb`. The map file can be named anything. For our convention, we use the domain name. Example 6-9 shows the contents inside the file.

Example 6-9 Contents of `/etc/courier/userdb/lspl.ibm.com`

```
sacha@lspl.ibm.com
home=/home/courier/lspl.ibm.com/sacha|mail=/home/courier/lspl.ibm.com/sacha|uid=508|gid=508
|systempw=1GoSZm99NNnwE
darlene@lspl.ibm.com
home=/home/courier/lspl.ibm.com/darlene|mail=/home/courier/lspl.ibm.com/darlene|uid=508|gid=508|systempw=xSQk0h/Y.6Rac
```

In our example, we have the information for two users, `sacha@lspl.ibm.com` and `darlene@lspl.ibm.com`. To generate the entry a user, for example, `sacha@lspl.ibm.com`, enter:

```
# userdbpw | userdb lspl.ibm.com/sacha@lspl.ibm.com set
home=/home/courier/lspl.ibm.com/sacha mail=/home/courier/lspl.ibm.com/sacha uid=508 gid=508
systempw
```

For user `sacha@lspl.ibm.com`, we set the following fields:

- ▶ The home directory is set to `/home/courier/lspl.ibm.com/sacha`.
- ▶ The maildir directory is also set to `/home/courier/lspl.ibm.com/sacha`.
- ▶ The uid has to be set to the uid of the user *courier*, which in our case is 508.
- ▶ The gid also has to be set to the gid of the user *courier*, which is also 508.
- ▶ The `systempw` field tells `userdbpw` to query for the password of this user.

For the next user, we run:

```
# userdbpw | userdb lspl.ibm.com/darlene@lspl.ibm.com set
home=/home/courier/lspl.ibm.com/darlene mail=/home/courier/lspl.ibm.com/darlene uid=508
gid=508 systempw
```

Then, run `makeuserdb` to generate the hashed database file `/etc/courier/userdb.dat`:

```
# makeuserdb
```

Create the maildir directory for each user. Instead of calling them maildir, we name them after each user. They have to be owned by the actual system user courier:

```
# mkdir /home/courier/lsp1.ibm.com
# maildirmake /home/courier/lsp1.ibm.com/sacha
# maildirmake /home/courier/lsp1.ibm.com/darlene
# chown -R courier.courier /home/courier/lsp1.ibm.com
```

We can now check if these accounts are active on POP3 as shown in Example 6-10.

Example 6-10 Checking if POP3 is active

```
[root@blade3 user]# telnet 127.0.0.1 110
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
+OK Hello there.
user sacha@lsp1.ibm.com
+OK Password required.
pass smile
+OK logged in.
```

Notice that the user name for these virtual mail users is their complete e-mail address (including domain name), not just the user name portion.

Configuring Postfix to deliver to virtual Courier mailboxes

As we configured Postfix to deliver e-mail messages to the maildir format that Courier uses, we must tell Postfix to deliver these messages to the virtual mailboxes.

Add the following lines from Example 6-11 to /etc/postfix/main.cf.

Example 6-11 Defining virtual mailbox delivery for Postfix

```
virtual_transport = virtual
virtual_mailbox_base = /home/courier
virtual_mailbox_maps = hash:/etc/postfix/vmailbox
virtual_mailbox_domains = $myhostname localhost.$mydomain $mydomain
virtual_uid_maps = static:508
virtual_gid_maps = static:508
```

The directive virtual_mailbox_maps tells Postfix to look in a hashed file called /etc/postfix/vmailbox.db (created from the file /etc/postfix/vmailbox) to map e-mail addresses to mailboxes on the system.

The contents of /etc/postfix/vmailbox are:

```
sacha@lsp1.ibm.com      lsp1.ibm.com/sacha/
darlene@lsp1.ibm.com   lsp1.ibm.com/darlene/
```

To create the hashed file /etc/postfix/vmailbox.db, run the following command:

```
# postmap /etc/postfix/vmailbox
```

The directive `virtual_mailbox_base` tells Postfix what pathname to use as a prefix for every entry in `/etc/postfix/vmailbox`. Since our prefix is `/home/courier`, the message for `sacha@lspl.ibm.com` is sent to `/home/courier/lspl.ibm.com/sacha`.

The directive `virtual_mailbox_domains` tells Postfix for which domains to receive virtual e-mail.

The directives `virtual_uid_maps` and `virtual_gid_maps` tells Postfix what UIDs and GIDs to use. Here, we tell Postfix to use 508, which points to the courier account we created. Make sure there is no space between the words `static:` and 508.

After you make the changes to the Postfix configuration file and creating and hashing the `/etc/postfix/vmailbox` file, restart Postfix:

```
# service postfix restart
```

To test the service, follow the sequence shown in Example 6-12.

Example 6-12 Testing the mail server

```
[root@blade3 postfix]# mail -v darlene@lspl.ibm.com
Subject: Hello, world!
Hello
Cc:
[root@blade3 postfix]# telnet 127.0.0.1 110
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^'.
+OK Hello there.
user darlene@lspl.ibm.com
+OK Password required.
pass smile
+OK logged in.
list
+OK POP3 clients that break here, they violate STD53.
1 432
retr 1
+OK 432 octets follow.
Return-Path: <root@blade3.lspl.ibm.com>
X-Original-To: darlene@lspl.ibm.com
Delivered-To: darlene@lspl.ibm.com
Received: by blade3.lspl.ibm.com (Postfix, from userid 0)
        id 197A46400D; Mon, 5 May 2003 17:02:38 -0700 (PDT)
To: darlene@lspl.ibm.com
Subject: Hello, world!
Message-Id: <20030506000238.197A46400D@blade3.lspl.ibm.com>
Date: Mon, 5 May 2003 17:02:38 -0700 (PDT)
From: root@blade3.lspl.ibm.com (root)
Hello
quit
+OK Bye-bye.
Connection closed by foreign host.
```

Pros and cons of this approach

With the steps we've taken, we disassociated system accounts from e-mail accounts. The system that we set up is usable, but not perfect. Why?

- ▶ We're still limited to a single server, because all our configuration files and maps are carried in a single location. To get around this, we can share these files using NFS (since they're primarily read-only).
- ▶ Configuration files and maps are all in text. We have to maintain separate files for Courier and Postfix. To get around this, we can write user management scripts to edit both configuration files simultaneously.

The work-arounds we mentioned are just that – work-around solutions. They are shortcuts that don't really address the fundamental problems of producing a reliable mail system.

A reliable mail system should have:

- ▶ Multiple mail servers in active standby or in a load-balancing configuration. If a mail server fails, the other servers should be able to take over the load.
- ▶ A shared file system, either through shared SCSI storage or through a network-shared file system, either Samba or NFS. The shared file system itself must be robust.
- ▶ A centralized directory that the mail server can refer to for account information and authorization. As with the shared file system, the directory must also be robust.
- ▶ Multiple mail relays and mail exchangers to queue the mail in case the network is temporarily not reachable.

Figure 6-2 illustrates a more reliable mail system.

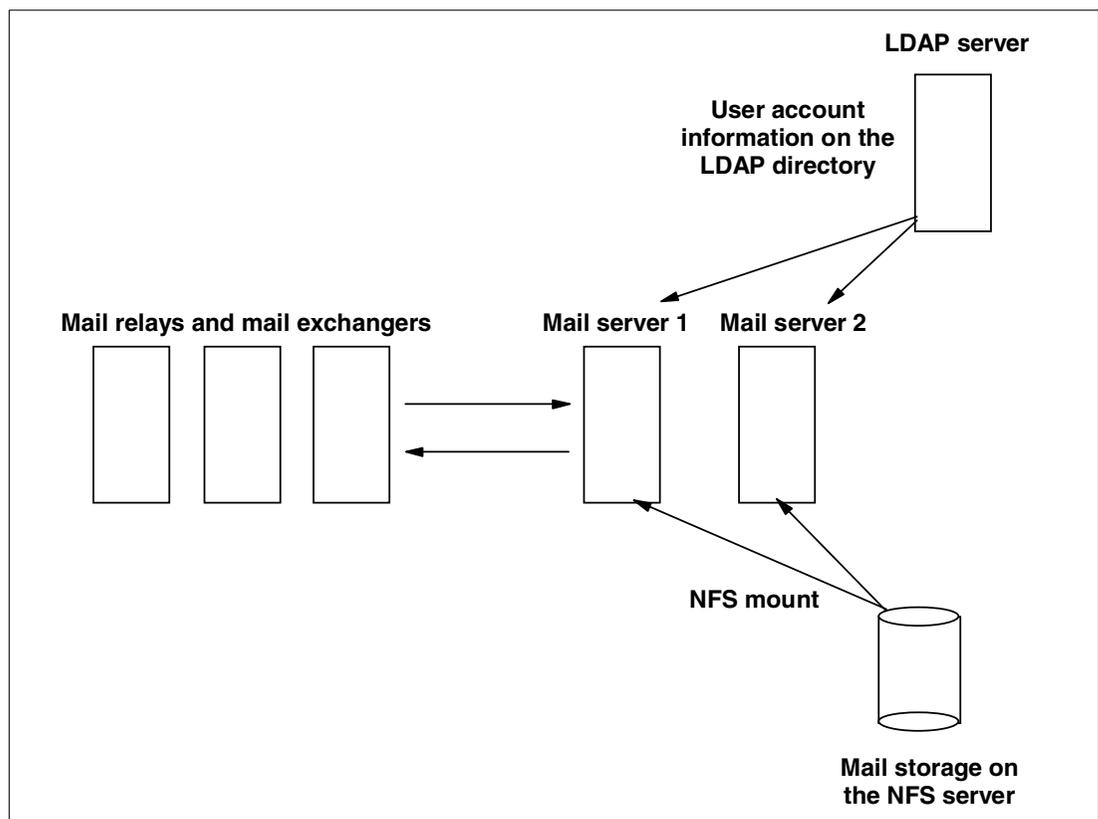


Figure 6-2 A more reliable mail system

Earlier in this redbook, we show how to set up highly-available shared file services and directory services. Now we focus on setting up mail servers that authenticate against an LDAP directory.

6.2.6 Virtual mail servers with Postfix, OpenLDAP, and Courier

The Center for Realtor Technology (CRT), the technology arm of the National Association of Realtors (NAR) in the U.S., has put together an excellent HOW-TO document on setting up a virtual mail server using Postfix, OpenLDAP, and Courier. The original document is posted on the Web at:

<http://www.crt.realtors.org/projects/email-redir/paper-html/implementation.html>

Note: CRT-NAR supports the Java Mail Manager (JAMM) project. JAMM is a Web application to manage virtual e-mail account information stored in an LDAP directory. It is meant as an administration tool for sites using mail servers, such as Postfix and Courier-IMAP, that store virtual user information in LDAP, such as OpenLDAP. You can find JAMM on the Web at:

<http://jamm.sourceforge.net>

We reproduce this setup in our BladeCenter as explained in the following sections.

Design objectives

Figure 6-3 shows how the components interact with each other. Postfix accepts incoming mail from SMTP and delivers it to the maildir mailboxes on the file system. Postfix delivers the mail itself for virtual users and uses procmail as the MDA for local users.

Courier, as we've already seen, provides remote access to the maildir mailboxes via the IMAP and POP protocols.

User account information is stored in an LDAP directory since LDAP provides a mechanism for searching for valid users, getting user information, and

authenticating users. It is possible to avoid an LDAP directory. However, we've already seen that it is difficult to maintain two sets of account information.

Configuring OpenLDAP

OpenLDAP setup is documented in detail in Chapter 4, "Plumbing: Network infrastructure" on page 25. We only give the minimum amount of detail necessary for the configuration files. Review the steps in the previous chapter if you are not sure about how to proceed with the OpenLDAP configuration.

For simplicity, we assume a fresh LDAP installation. We are not configuring LDAP to accommodate authentication for multiple applications.

Our LDAP tree appears as shown in Figure 6-4.

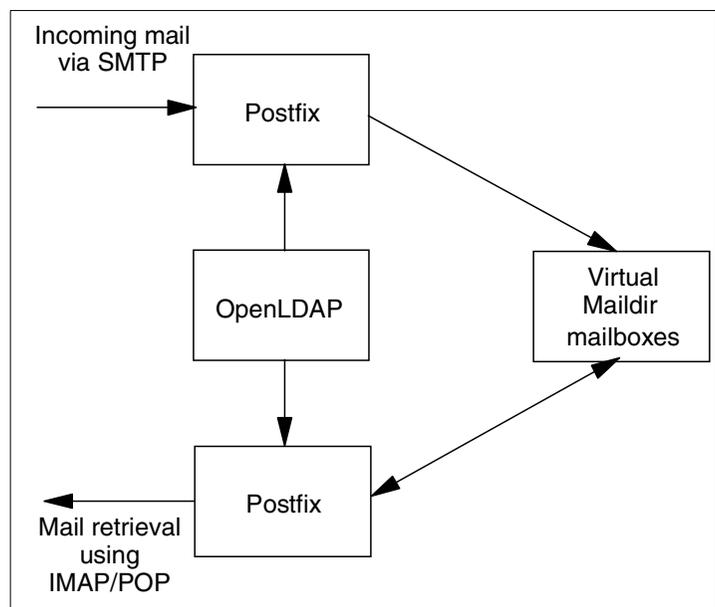


Figure 6-3 How the components fit together

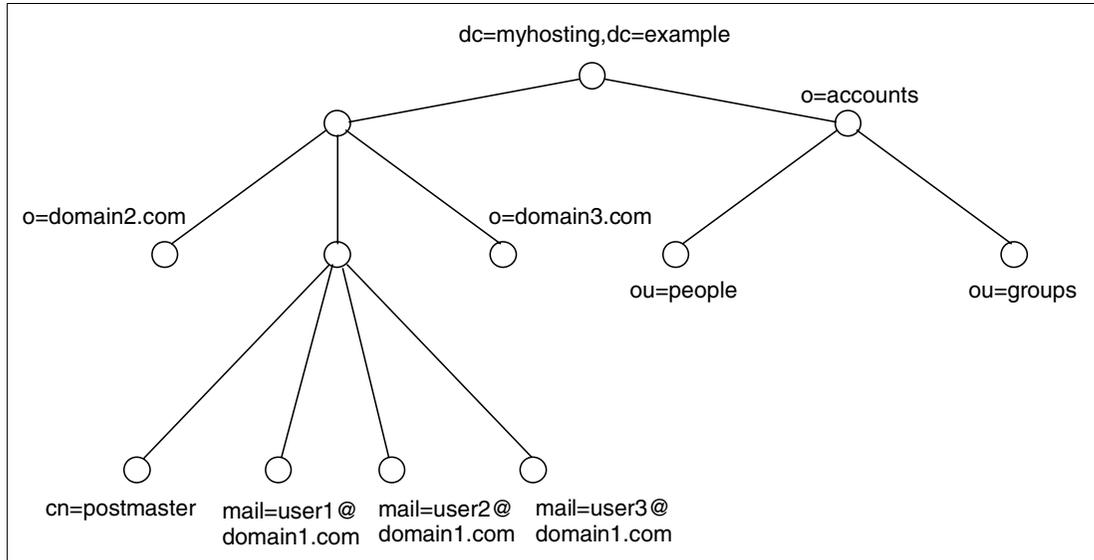


Figure 6-4 Our LDAP tree for virtual mail services

Changes to `/etc/openldap/slapd.conf`

You need to include Courier's schema file into the schemas recognized by your LDAP directory. Courier includes its required schema in `authlib/authldap.schema` in the Courier distribution. Copy `authlib/authldap.schema` to `/etc/openldap/schema/courier.schema`. `courier.schema` depends on `cosine.schema` and `nis.schema`.

At a minimum, your `/etc/openldap/slapd.conf` should have the lines listed in Example 6-13.

Example 6-13 `/etc/openldap/slapd.conf` configuration

```

include          /usr/local/etc/openldap/schema/cosine.schema
include          /usr/local/etc/openldap/schema/nis.schema
include          /usr/local/etc/openldap/schema/courier.schema
database         ldbm
directory        /usr/local/var/openldap-ldbm
suffix           "dc=myhosting,dc=example"
rootdn           "cn=Manager,dc=myhosting,dc=example"
rootpw           secret
index objectClass pres,eq
index mail,cn    eq,sub
access to dn=".*,o=([^,]+),o=hosting,dc=myhosting,dc=example"
    attr=userPassword
    by self write
    by group/organizationalRole/roleOccupant=\
       "cn=postmaster,o=$1,o=hosting,dc=myhosting,dc=example" write
    by anonymous auth
    by * none
access to dn=".*,o=([^,]+),o=hosting,dc=myhosting,dc=example"
    by self write
    by group/organizationalRole/roleOccupant=\
       "cn=postmaster,o=$1,o=hosting,dc=myhosting,dc=example" write
    by * read
access to *
    by * read
  
```

Set up the OpenLDAP server as shown in Example 6-13 and start it.

The base directory entries

The base directory entries are shown in Example 6-14.

Example 6-14 Base directory entries for our LDAP directory

```
dn: dc=myhosting, dc=example
objectClass: top
dn: cn=Manager, dc=myhosting, dc=example
objectClass: top
objectClass: organizationalRole
cn: Manager
dn: o=hosting, dc=myhosting, dc=example
objectClass: top
objectClass: organization
o: hosting
```

Place these entries in an LDIF file and add them to the OpenLDAP directory.

Adding domains

Under the design specification, we set up a mail server that supports multiple domains. Each domain should have a postmaster and abuse entries at a minimum. These are the accounts to which undelivered mail and abuse reports will be sent.

A tree for domain1.com has the entries as shown in Example 6-15.

Example 6-15 LDAP entries for a domain

```
dn: o=domain1.com, o=hosting, dc=myhosting, dc=example
objectClass: top
objectClass: organization
o: domain1.com
dn: cn=postmaster, o=domain1.com, o=hosting, dc=myhosting, dc=example
objectClass: top
objectClass: organizationalRole
objectClass: CourierMailAlias
cn: postmaster
mail: postmaster@domain1.com
maildrop: postmaster
dn: mail=abuse@domain1.com, o=domain1.com, o=hosting, dc=myhosting, dc=example
objectClass: top
objectClass: CourierMailAlias
mail: abuse@domain1.com
maildrop: abuse
```

Place these entries in an LDIF file and add them to the OpenLDAP directory.

Adding users

Example 6-16 shows an example entry for a user.

Example 6-16 LDAP entry for a mail user

```
dn: mail=user1@domain1.com, o=domain1.com, o=hosting, dc=myhosting, dc=example
objectClass: top
objectClass: CourierMailAccount
mail: user1@domain1.com
homeDirectory: /home/vmail/domains
uidNumber: 101
gidNumber: 101
mailbox: domain1.com/user1
```

The mail ID of this user is user1@domain1.com.

The homeDirectory and mailbox point to the physical mailbox on the file system. The actual mailbox of user1@domain1.com is /home/vmail/domains/domain1.com/user1. This is placed in a shared storage for failover.

The uidNumber and gidNumber are required but not used. They should be set to the uid and gid of the vmail user (which we create later). This user is similar to the courier mail account we created in our earlier example.

To give user1@domain1.com the role of postmaster, modify the postmaster entry in the LDAP directory and add user1@domain1.com as a roleOccupant as shown in Example 6-17.

Example 6-17 Defining a role for a user

```
dn: cn=postmaster, o=domain1.com, o=hosting, dc=myhosting, dc=example
changetype: modify
add: roleOccupant
roleOccupant: mail=user1@domain1.com, o=domain1.com, o=hosting,
dc=myhosting, dc=example
```

The user1@domain1.com does not yet have a password, so add a password using **1dapasswd**.

Configuring Postfix

The Postfix RPM (from <http://postfix.w10.org/>) includes LDAP support. Therefore, there is no need to recompile Postfix. All the changes that we need to make are in /etc/postfix/main.cf.

Transport map

The transport table maps domains to message delivery transports or relay hosts. In the case of our virtual domains, we want to map them to the virtual delivery agent that comes with Postfix (as defined in /etc/postfix/master.cf). Example 6-18 shows a transport table.

Example 6-18 A transport map

```
domain1.com virtual:  
domain2.comvirtual:  
domain3.comvirtual:
```

Write this table to the text file `/etc/postfix/transport`. Then, run the `postmap` program against the file as shown here:

```
# postmap /etc/postfix/transport
```

Edit `/etc/postfix/main.cf` to tell Postfix that there is a transport table. Look for the `transport_maps` field in the configuration file, and point it to the transport table. Modify the `mydestination` field so that it uses the transport maps you just defined:

```
transport_maps=hash:/etc/postfix/transport  
mydestination = $myhostname, localhost.$mydomain, $mydomain, $transport_maps
```

Configuring Postfix to use LDAP sources (aliases)

Tell Postfix all about the LDAP server, and configure Postfix accordingly so that it queries the right attributes from the LDAP server. Edit `/etc/postfix/main.cf` so that it has the lines shown in Example 6-19.

Example 6-19 Modifications to /etc/postfix/main.cf

```
virtual_maps = ldap:alias  
aliases_server_host = blade9.lspl.ibm.com, blade10.lspl.ibm.com  
aliases_search_base = o=hosting,dc=myhosting,dc=example  
aliases_query_filter = (&(mail=%s)(objectClass=CourierMailAlias))  
aliases_result_attribute = maildrop  
aliases_bind = no  
aliases_cache = yes
```

Postfix allows us to define multiple LDAP sources. It tries the LDAP servers listed until it finds one that responds.

Configuring Postfix to use LDAP sources (accounts)

Finally, tell Postfix how to look up valid user accounts on the system. First define a real UNIX account user that receives the mail into its home directory, in this case `vmail` (following our settings in the LDAP server):

```
# useradd -u 101 -g 101 vmail
```

You need to do this on all the mail servers that you are setting up.

Then, define this account in the `/etc/postfix/main.cf` file. Tell Postfix to use this for checking accounts. In the `/etc/postfix/main.cf` file, add the lines listed in Example 6-20.

Example 6-20 Configuring Postfix to check accounts from LDAP

```
virtual_mailbox_base=/home/vmail/domains  
virtual_mailbox_maps=ldap:accounts  
virtual_minimum_uid=101  
virtual_uid_maps=static:101
```

```
virtual_gid_maps=static:101
accounts_server_host=blade11.lspl.ibm.com, blade12.lspl.ibm.com
accounts_search_base=o=hosting,dc=myhosting,dc=example
accounts_query_filter=(amp(mail=%s)(objectClass=CourierMailAccount))
accounts_result_attribute=mailbox
accounts_cache=yes
accounts_bind=no
local_recipient_maps=$alias_maps UNIX:passwd.byname $virtual_mailbox_maps
```

Configuring Courier

The Courier RPMs that we compiled earlier includes an LDAP package. Install that package to enable Courier to query an LDAP server.

authdaemonrc

Courier uses an authentication daemon to keep authentication separate from the rest of the system. This is listed in the `authmodulelist` field of `/etc/courier/authdaemonrc`. At a minimum, your `authmodulelist` should read:

```
authmodulelist="authldap authpam"
```

The default settings on the `authdaemonrc` daemon include substantially more daemons. You can leave it as is or pare it down to the minimum.

/etc/courier/authldaprc

All LDAP parameters are in `/etc/courier/authldaprc`. Here are some key parameters that we specified:

```
LDAP_SERVER          blade11.lspl.ibm.com
LDAP_PORT            389
LDAP_BASEDN          dc=myhosting, dc=example
LDAP_GLOB_UID         vmail
LDAP_GLOB_GID         vmail
LDAP_HOMEDIR         homeDirectory
LDAP_MAILDIR          mailbox
LDAP_CRYPTPW          userPassword
LDAP_AUTHBIND1
```

At this point, setup of the mail server is complete. Restart the Postfix and Courier daemons for the changes to take effect.

Thoughts on setting up a high-available virtual mail system

As you can see from the steps we've taken, setting up a robust virtual mail system is not a trivial task. There are several components to consider:

- ▶ The MTA (Postfix or Sendmail)
- ▶ The mail handling agent (Courier, or alternatively, UW-IMAP and qmail)
- ▶ The centralized authentication database (OpenLDAP, or alternatively MySQL and PostgreSQL)
- ▶ A robust message store (high-availability NFS or high-availability Samba).

You have to know these individual components fairly well to get your system up and running.

Is it worth the effort to do this? As always, it's a matter of trade-offs. By increasing the number of features and the number of users that we can support, we also increase the complexity of our installation.

If you're setting up a mail server for a small company, and you don't expect it to grow beyond a couple hundred users in a single domain whose basic needs can be fulfilled by IMAP and POP3, you can get by with the Sendmail-UW-IMAP. You can put the message store on a highly-available file share, and share that message store across two servers. It's simple to implement, but with some growth limitations.

Even if you have to support multiple domain names for the same server, that objective is achievable with the basic Sendmail package with the Red Hat distribution. You can use LDAP and pluggable authentication module (PAM) to synchronize accounts.

If you're an application service provider who is supporting a large numbers of users, anywhere from hundreds to thousands, with a bit more flexibility and features, we recommend that you use a setup as we've outlined here. Overall, the features of the software presented allow for a more flexible growth path.

Keep in mind that we've barely scratched the surface of open-source mail server technologies. There are dozens of possible configurations. There are other things that we have not explored, such as the use of IMAP proxies or a relational database as a message store.

Ultimately, you have to decide on the trade-offs between complexity, usability, ease-of-implementation, and cost.

Alternative configurations

In the course of looking for resources for virtual mail systems, we've come across other interesting implementations and documents that are worth a look:

- ▶ ISPMAN, originated by Atif Ghaffar, is a system to consolidate user account information for Web, e-mail, FTP, and DNS into an LDAP directory. It works with Postfix, Courier, and Cyrus. It provides a Web-based front end for managing user accounts and allows for delegated administration. For more information, see:

<http://www.ispman.org>

- ▶ Dan Kuykendall wrote a document outlining the use of qmail, vmailmgr, and Courier. The document is available on the Web at:

<http://www.clearrivertech.com/linux/HOWTO/Qmail-VMailMgr-Courier-imap-HOWTO/index.html>

- ▶ Luc de Louw wrote a document outlining the use of Postfix, Cyrus, and the Web-based cyradm. The document is available on the Web at:

<http://www.delouw.ch/linux/postfix.phtml>

6.2.7 Dealing with spam and viruses

Spam is the scourge of e-mail. Every day, thousands of unsolicited and unwanted e-mail cross into our mail servers advertising products we don't want or hawking dubious schemes.

Spam floods the Internet with multiple copies of the same message. Most spam is commercial advertising, often for dubious products, get-rich-quick schemes, or quasi-legal services. Why is spam so prevalent? Because it costs the sender very little to send, Most of the costs are paid for by the recipient or the carriers rather than by the sender.

The Web site from Abuse.net discusses the evils of spam in much more detail. You can find this site at:

<http://spam.abuse.net>

At the same time, viruses are getting more sophisticated and automated. E-mail is one of the popular means of disseminating them. While Linux is at present fairly well-insulated from the virus problem, many client computers still use Windows, which is highly susceptible to viruses. Couple this with virus-susceptible Windows programs in wide use, such as Outlook, Microsoft Office, and Internet Explorer, and the problem multiplies.

Tools for dealing with spam

Fortunately, all is not hopeless. There are open source tools for dealing with spam, as explained in this section.

SpamAssassin

SpamAssassin is a mail filter to identify spam. It uses a rule base to test an incoming e-mail's headers and message body to determine whether it is spam. If an e-mail is determined to be spam, it is altered to reflect this so the user has the option of deleting it. SpamAssassin can also reject e-mail originating from blacklisted sources. And it can cooperate with a spam-tracking database to check incoming messages against a list of known spam messages.

SpamAssassin is more sophisticated than the simple keyword matching provided by most SMTP antivirus software. SpamAssassin uses a scoring system to mark messages as spam only when they have enough spam characteristics in total. This is used in combination with other features, which results in very few false positives.

SpamAssassin doesn't block spam outright. Instead, it tags messages as probable spam by changing the Subject line and message headers. SpamAssassin uses this method because no automated system can recognize spam with 100% certainty. Instead, SpamAssassin identifies probable spam e-mail, but leaves the choice of what to do with the users.

SpamAssassin is open source software and is available on the Web at:

<http://eu.spamassassin.org/index.html>

Anomy Sanitizer

Anomy Sanitizer is an e-mail attachment scanner. Its purpose is to protect your mail server at the gateway by blocking infected payloads and other types of exploits. The Sanitizer performs the following actions:

- ▶ Disables potentially dangerous HTML code, such as Javascript, within incoming e-mail.
- ▶ Protects you from e-mail-based break-in attempts that exploit bugs in common e-mail programs, for example, Outlook, Eudora, and Pine.
- ▶ Blocks attachments based on their file names.

Sanitizer is the Perl program so it is very portable and fairly fast. It is also easy to install.

Sanitizer was written by Bjarni R. Einarsson and is sponsored by F-Prot (<http://www.f-prot.com>), a virus protection company. Sanitizer's Web site is at:

<http://mailtools.anomy.net>

Using SpamAssassin and Sanitizer to protect a site

The configuration we want protects an entire site with SpamAssassin and Sanitizer as shown in Figure 6-5. In this case, we should dedicate one of our blades to perform the filtering functions and simply relay the approved messages to the actual mail server.

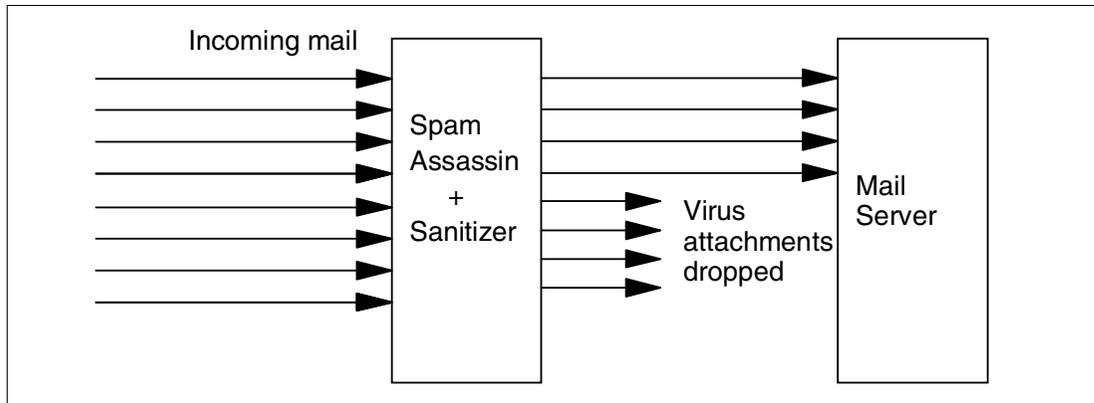


Figure 6-5 SpamAssassin and Sanitizer

This setup can be used with the open source mail server we discussed in earlier sections and with other types of mail servers.

The steps here are based on the document “Filtering Malware and Spam with Postfix”. You can find this document on the Web at:

<http://advosys.ca/papers/printable/postfix-filtering.html>

Required software

To set up SpamAssassin and Sanitizer as a site-wide mail filter against spam, you need the following components:

- ▶ Postfix:
<http://www.postfix.org>
- ▶ SpamAssassin:
<http://spamassassin.org>
- ▶ Anomy Sanitizer:
<http://mailtools.anomy.net/>
- ▶ MIME::Base64 Perl module
- ▶ MIME::QuotedPrint Perl module
- ▶ Mail::Audit Perl module

You can install the Perl modules by logging in to the Comprehensive Perl Archive Network (CPAN). We show you how to do this later. You may need to set up Perl to query CPAN. To do this, run the following command:

```
# perl -MCPAN -e shell
```

The initialization phase can be quite lengthy, but each step is well-explained and the program gives you the option of using the default answers to most (but not all) of the questions.

Creating a filter account

Create a real Linux account and groupname called *filter*. Specify the home directory to be `/var/spool/filter`:

```
# useradd -d /var/spool/filter filter
```

Installing required Perl modules

To download the modules using CPAN, type:

```
# perl -MCPAN -e shell
```

If you already configured CPAN, this immediately puts you into an interactive shell.

Type the following command sequences into this interactive shell (we do not include system responses because they are quite lengthy):

```
o conf prerequisites_policy ask
install MIME::Base64
install MIME::QuotedPrint
install Mail::Audit
quit
```

Step through the questions and install any prerequisite Perl software.

Installing Anomy

Anomy, like SpamAssassin, is a set of Perl programs. The prerequisite Perl modules were installed in the previous step.

Unpack Anomy in a directory. This creates a directory called *anomy*, with all the programs and modules inside. Putting Anomy in any directory should work. However, we recommend that you put these files in */usr/local/anomy*.

Configuring Anomy

In */usr/local/anomy*, create a file called *anomy.conf*. Example 6-21 shows the sample configuration that we use, borrowed from the Advosys document.

Example 6-21 Filtering configuration file for Anomy Sanitizer

```
# Example configuration file for Anomy Sanitizer
# From http://advosys.ca/papers/postfix-filtering.html
# Advosys Consulting Inc., Ottawa
# Works with Anomy Sanitizer revision 1.53
# Do not log to STDERR:
feat_log_stderr = 0
# Don't insert log in the message itself:
feat_log_inline = 0
# Advertisement to insert in each mail header:
header_info = X-Sanitizer: Advosys mail filter
header_url = 0
header_rev = 0
# Enable filename based policy decisions:
feat_files = 1
# Protect against buffer overflows and null values:
feat_lengths = 1
# Replace MIME boundaries with our own:
feat_boundaries = 1
# Fix invalid and ambiguous MIME boundaries, if possible:
feat_fixmime = 1
# Trust signed and/or encrypted messages:
feat_trust_pgp = 1
msg_pgp_warning = WARNING: Unsanitized content follows.\n
# Defang shell scripts:
feat_scripts = 0
# Defang active HTML:
```

```

feat_html = 1
# Defang UUEncoded files:
feat_uuencoded = 0
# Sanitize forwarded content too:
feat_forwards = 1
# Testing? Set to 1 for testing, 0 for production:
feat_testing = 0
# # Warn user about unscanned parts, etc.
feat_verbose = 1
# Force all parts (except text/html parts) to
# have file names.
feat_force_name = 1
# Disable web bugs:
feat_webbugs = 1
# Disable "score" based mail discarding:
score_panic = 0
score_bad = 0
msg_file_drop = \n*****\n
msg_file_drop += NOTE: An attachment named %FILENAME was deleted from
msg_file_drop += this message because it contained a windows executable
msg_file_drop += or other potentially dangerous file type.
msg_file_drop += Contact the system administrator for more information.
##
## File attachment name mangling rules:
##
# Specify the Anomy temp file and quarantine directory
file_name_tpl = /var/spool/filter/att-$F-$T.$$
# Number of rulesets we are defining:
file_list_rules = 2
file_default_policy = defang
# Delete probably nasty attachments:
file_list_1 = (?i)(winmail.dat)|
file_list_1 += (\.(exe|com|vb[se]|dll|ocx|cmd|bat|pif|lnk|hlp|ms[ip]|reg|sct|inf
file_list_1 += |asd|cab|sh[sh]|scr|cpl|chm|ws[fhc]|hta|vcd|vcf|eml|nws))$
file_list_1_policy = drop
file_list_1_scanner = 0
# Allow known "safe" file types and those that will be
# scanned by the user's desktop virus scanner:
file_list_2 = (?i)\.
# Word processor and document formats:
file_list_2 += (doc|dot|txt|rtf|pdf|ps|htm|[sp]?html?
# Spreadsheets:
file_list_2 += |xls|xlw|xlt|csv|wk[1-4]
# Presentation applications:
file_list_2 += |ppt|pps|pot
# Bitmap graphic files:
file_list_2 += |jpe?g|gif|png|tiff?|bmp|psd|pcx
# Vector graphics and diagramming:
file_list_2 += |vsd|drw|cdr|swf
# Multimedia:
file_list_2 += |mp3|avi|mpe?g|mov|ram?|mid|ogg
# Archives:
file_list_2 += |zip|gz|rar|tgz|bz2|tar
# Source code:
file_list_2 += |[ch](pp|\+|+)?|s|inc|asm|patch|java|php\d?|jsp|bas)
file_list_2_policy = accept
file_list_2_scanner = 0

```

```
# Any file type not listed above gets renamed to prevent
# ms outlook from auto-executing it.
```

You may want to customize this for your site. In our setting, we opted to drop all executable attachments.

Change the ownership of the entire anomy directory and change the permissions:

```
# chown -R root.filter /usr/local/anomy
# chmod 0750 /usr/local/anomy
```

Installing SpamAssassin

Since SpamAssassin is available as a Perl module, you can install it through CPAN. As we did earlier, bring up the Perl CPAN shell:

```
# perl -MCPAN -e shell
```

Install SpamAssassin using the following sequence:

```
o conf prerequisites_policy ask
install Mail::SpamAssassin
quit
```

Configuring SpamAssassin

Out of the box, SpamAssassin already contains a good set of rules. You can specify your own settings at `/etc/mail/spamassassin/local.cf`.

You should, however, create a *whitelist of e-mail addresses*. This is a list of legitimate senders, so they are not inadvertently blocked by SpamAssassin. Here's an example of a whitelist, which you insert into `/etc/mail/spamassassin/local.cf`:

```
whitelist_from sacha@yahoo.com # whitelist one specific sender
whitelist_from *@ibm.com        # whitelist an entire domain
whitelist_from *@securityfocus.com
```

Configuring Postfix

Before you attempt to configure Postfix to work with SpamAssassin and Anomy, make sure it is working correctly first. Test it in the configuration first before you make the changes for SpamAssassin and Anomy.

This is how Postfix is configured. Any message that Postfix receives is piped through Sanitizer and then through SpamAssassin. Postfix then takes the result and delivers it to the final destination.

Filter script

To do this, it must invoke an external script, as described in the `FILTER_README` document of Postfix. Example 6-22 shows the sample script `/usr/local/anomy/filter.sh`.

Example 6-22 Filtering script

```
#!/bin/sh
# filter.sh
# Simple filter to plug Anomy Sanitizer and SpamAssassin
# into the Postfix MTA
# From http://advosys.ca/papers/postfix-filtering.html
# Advosys Consulting Inc., Ottawa
# For use with:
```

```

# Postfix 20010228 or later
# Anomy Sanitizer revision 1.49 or later
# SpamAssassin 2.42 or later
# Note: Modify the file locations to match your particular
#       server and installation of SpamAssassin.
# File locations:
# (CHANGE AS REQUIRED TO MATCH YOUR SERVER)
# Sendmail below is a link to Postfix
INSPECT_DIR=/var/spool/filter
SENDMAIL="/usr/sbin/sendmail -i"
ANOMY=/usr/local/anomy
SANITIZER=/usr/local/anomy/bin/sanitizer.pl
ANOMY_CONF=/usr/local/anomy/anomy.conf
ANOMY_LOG=/dev/null
SPAMASSASSIN=/usr/bin/spamassassin
export ANOMY
# Exit codes from <syssexits.h>
EX_TEMPFAIL=75
EX_UNAVAILABLE=69
cd $INSPECT_DIR || { echo $INSPECT_DIR does not exist; exit $EX_TEMPFAIL; }
# Clean up when done or when aborting.
trap "rm -f out.$$" 0 1 2 3 15
cat | $SPAMASSASSIN -x | $SANITIZER \
    $ANOMY_CONF 2>>/dev/null > out.$$ || \
    { echo Message content rejected; exit $EX_UNAVAILABLE; }
$SENDMAIL "$@" < out.$$
exit $?

```

The script should have the following permissions:

```
-rwxr-x--- owner=root group=filter
```

Temporary directory for incoming mail messages

The script needs a temporary directory to store files. Create this directory and assign the appropriate permissions:

```

# mkdir /var/spool/filter
# chown root.filter /var/spool/filter
# chmod 0770 /var/spool/filter

```

Changing master.cf

Add the following lines to the end of `/etc/postfix/master.cf`:

```

filter  UNIX - n n - - pipe
        flags=Rq user=filter argv=/usr/local/anomy/filter.sh -f ${sender} -- ${recipient}

```

Change the line in `master.cf` that controls the Postfix daemon:

```
smtp  inet n - n - - smtpd -o content_filter=filter:dummy
```

To make the changes take effect immediately, type:

```
# postfix reload
```

Incoming and outgoing mail

Ideally, you want Postfix to filter only incoming mail. Unfortunately, Postfix cannot distinguish between incoming mail and outgoing mail. Considering how things are set up, the attachments to any e-mail we send are dropped!

There are two possible solutions for handling this. Postfix can run on a server with two network interfaces, one connected to the Internet and the other connected to the internal network. Figure 6-6 demonstrates this concept. You run two instances of Postfix, one listening to the external IP address and the other listening to the internal IP address.

You install two instances of Postfix, each one with its own configuration files and spool directory.

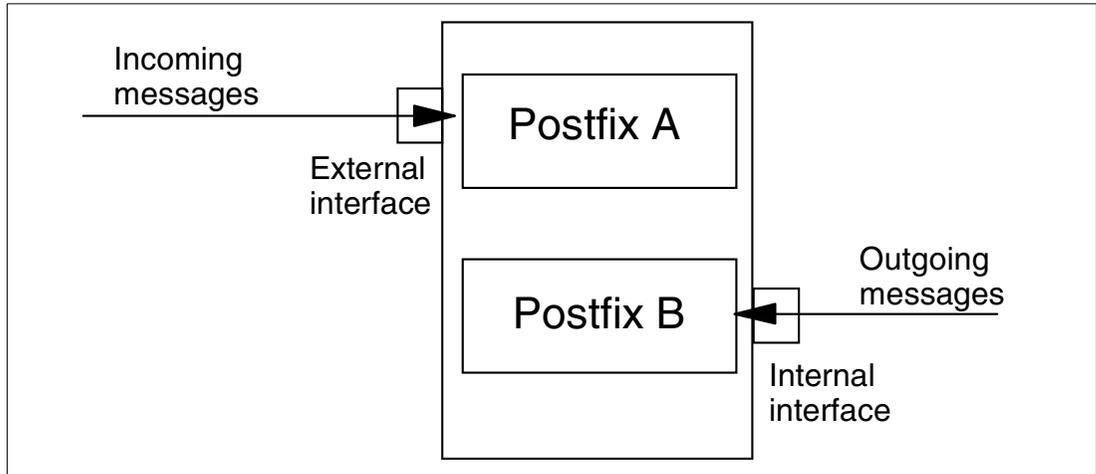


Figure 6-6 One server, two interfaces

This type of configuration is workable in a BladeCenter environment, but it may not be appropriate in all circumstances. For example, if one of the networks in BladeCenter is used for management, neither the external interface nor internal users should be able to access the interface.

Another solution is to use one blade as a pure mail relay and another as the actual mail server. Figure 6-7 illustrates this solution.

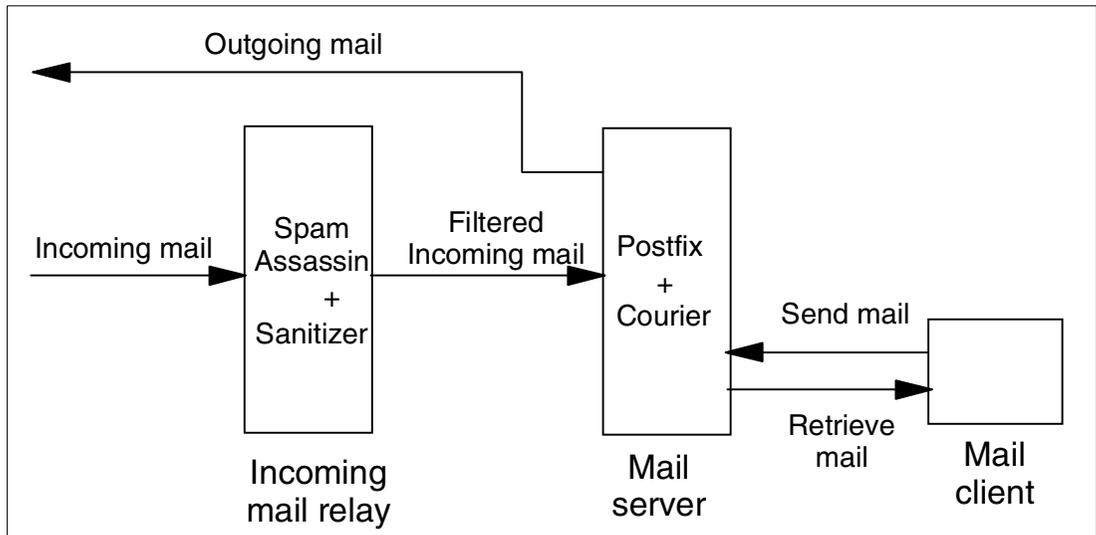


Figure 6-7 Relay and mail server

All incoming mail goes through the incoming mail relay. Outgoing mail originates from the mail server directly, either to another outgoing relay or to the Internet.

The mail server is configured normally. The incoming mail relay has the following line in its `/etc/postfix/main.cf` file:

```
relayhost = mailserver.yourdomain.com
```

Or it can point directly to the IP address of the mail server.

The DNS for the domain should point to the incoming mail relay as the preferred MX record for all incoming mail for the domain.

Alternatives

Spam is such a nuisance that the Freshmeat.net open source directory (<http://freshmeat.net>) lists 179 spam-related tools (as of this writing). Some alternatives are:

- ▶ **Spambouncer:** Similar to SpamAssassin. You can find Spambouncer on the Web at:
<http://www.spambouncer.org>
- ▶ **Obtuse:** A firewall kit with SMTP replacement and spam filtering tools. You can find Obtuse on the Web at:
<http://www.obtuse.com/smtpd.html>

At the moment, one promising approach to spam is *Bayesian filtering*. This is not a tool, but rather a technique implemented in many spam-filtering tools. Bayesian filtering uses a statistical method. It analyzes the characteristics of spam mail messages (frequency of occurrences of keywords together) and strings together these conditional probabilities to determine whether a message is spam. This technique can be incorporated as a module into SpamAssassin.

For more information on Bayesian filtering, see the following Web site:

<http://www.paulgraham.com/spam.html>

Likewise, for e-mail attachment virus scanning, several tools are available:

- ▶ AMaViS
<http://www.amavis.org>
- ▶ Inflex
<http://pldaniels.com/inflex/>
- ▶ Qmail-Scanner (also known as *scan4virus*)
<http://qmail-scanner.sourceforge.net/>
- ▶ MIMEDefang
<http://www.roaringpenguin.com/mimedefang>

6.2.8 Sendmail clusters on Linux

We looked in depth at creating reliable mail servers by combining Postfix, Courier, OpenLDAP, and Heartbeat. However, to really complete our picture of reliable e-mail, we have to paint the entire picture of Internet e-mail to include the actions of the MTAs and the DNS system.

The following study was conducted and co-authored by one of the principal authors of this redbook in an earlier test. For the complete text, see Highly Available/Scalable Sendmail Using Sendmail Clusters on Linux:

<http://www-1.ibm.com/servers/esdd/articles/sendmail/>

The tests were conducted on Sendmail only.

Clusters of servers running Sendmail or Postfix can deliver high performance and high availability at competitive prices. For experienced systems administrators, this has long been a commonly held practice.

We studied several configurations of Sendmail clusters on Linux and quantified their relative performance. We investigated and tested common performance tuning parameters in Sendmail's configuration and in the Linux operating system. We didn't have a shared disk for these tests, so we scoped the project to include SMTP routing and queuing only. This is a common configuration for a Sendmail cluster on the edge of a private network or as the front end for an internal mail store.

While our hardware resources were modest, we believe the relative differences make our results valuable to system architects who want to implement clusters of Linux-based Sendmail servers. Our results illustrate the relative importance of a Sendmail cluster's design features.

Summary results

There are many configuration options for Sendmail, LDAP and DNS. We consider only those that are important for this application. Unless otherwise stated, we used stock software and default settings. Of these options, we found a handful of factors that have dramatic effects on performance or that are required for scalability, such as LogLevel and QueueDirectory.

Ultimately, we found that even when Sendmail is correctly configured all the important factors led us to two facts:

- ▶ Sendmail is disk intensive, so the faster the disk, the faster Sendmail.
- ▶ Perceived performance may be impacted by factors out of your control. Remote DNSs fail, routes flap, queues fill up, and other third-party problems.

Findings

The conclusions of our study are explained in the following sections.

Clustered servers

We found that the best message throughput, approximately 100 messages per second, was by clustering two servers and adding a load balancer on the front end. This doubled the performance of the best single-server results, where we got approximately 50 messages per second. When we added a third server, we saw little or no improvement.

LogLevel

Because Sendmail logging is sometimes used like an audit trail, showing the ingress and egress of SMTP mail, it can be relatively expensive in terms of disk I/O. In some environments, it may be acceptable or desirable to turn this audit function off, delivering higher throughput. Even with the full logging turned on (LogLevel 9), we had acceptable performance by moving the log files onto faster file systems.

QueueDirectory

The QueueDirectory was an obvious point of contention. We found the best performance by using multiple queues and by switching QueueSortOrder to filename. Together, LogLevel and QueueDirectory account for most of our throughput gains.

Other configuration options

We also tested turning off Ident lookups, using SharedMem key and Delivery Mode for our workloads. These had little impact, but we assume that in real-world scenarios they might be more important.

Test scenarios

We evaluated several test scenarios:

- ▶ Single server
- ▶ Round-robin DNS
- ▶ Load Balancer
- ▶ MX-based failover

For the Load Balancer scenario, we tried both the switching appliance and a dedicated Linux server running balance software. We used a single host to find the optimum Sendmail configuration, stepping through variations in important configuration factors. Using the results of this testing, we derived an optimized configuration and used that in our different cluster configurations.

Round-robin DNS

Round-robin DNS is a straightforward way to multiplex incoming Internet SMTP traffic across machines. In its simplest form, several A records are entered for the one mail server host name. Each participating Sendmail server is configured to receive mail on behalf of this one host name. When a sender goes to deliver mail to the recipient, a DNS query is made. The results contain a list of all the A records for that host. By default, most MTA implementations take the first member of the list. Repeated queries for the same host name yield a rotating list of IP address (this is a feature of BIND/DNS). For example, if the name "us.ibm.com" is looked up on the Internet, the following IP address list is returned:

```
Name:    www.ibm.com
Addresses: 129.42.16.99, 129.42.17.99, 129.42.18.99, 129.42.19.99
Repeating the query returns:
Name:    www.ibm.com
Addresses: 129.42.19.99, 129.42.16.99, 129.42.17.99, 129.42.18.99
```

And again it returns:

```
Name:    www.ibm.com
Addresses: 129.42.17.99, 129.42.18.99, 129.42.19.99, 129.42.16.99
```

Figure 6-8 shows a round-robin DNS at work. All of the external interfaces of the Sendmail servers are directly connected to the Internet and published in DNS. Each machine acts as a SMTP router/buffer. It takes mail from the Internet and delivers it to a common mailbox server on a private network. This sort of arrangement is easy to set up, inexpensive, and if done correctly, trouble-free.

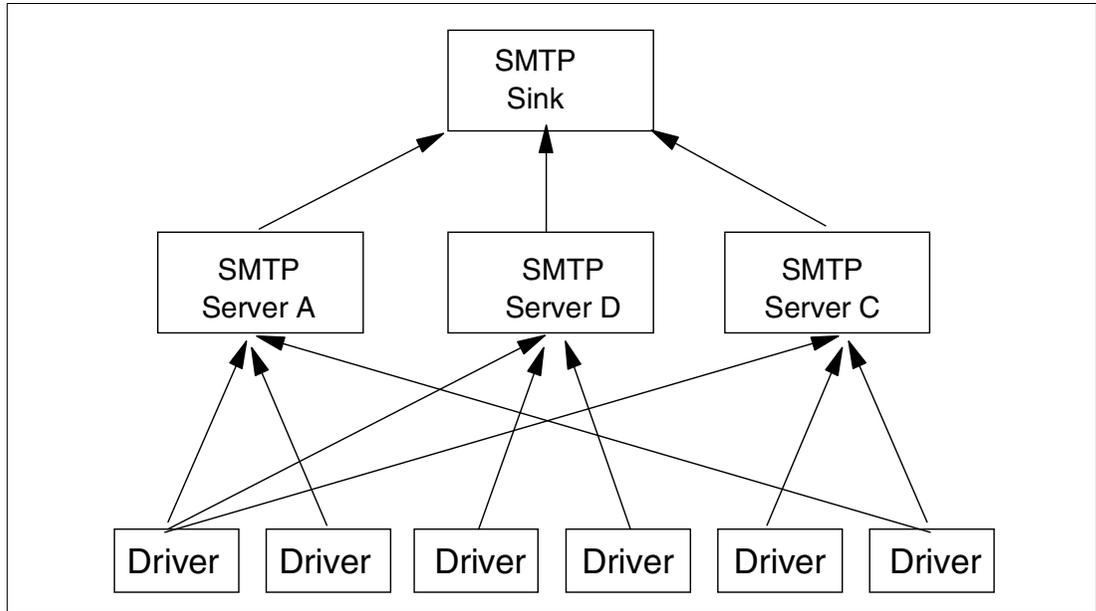


Figure 6-8 Three-way SMTP cluster using round-robin DNS

Figure 6-9 illustrates the problem of using round-robin DNS. Because of the way DNS works and the possibility of cached DNS records. If a host fails, mail agents may still try to contact it. At the least, mail may be delayed waiting for connections to time-out, be queued, and then resent after some backoff period.

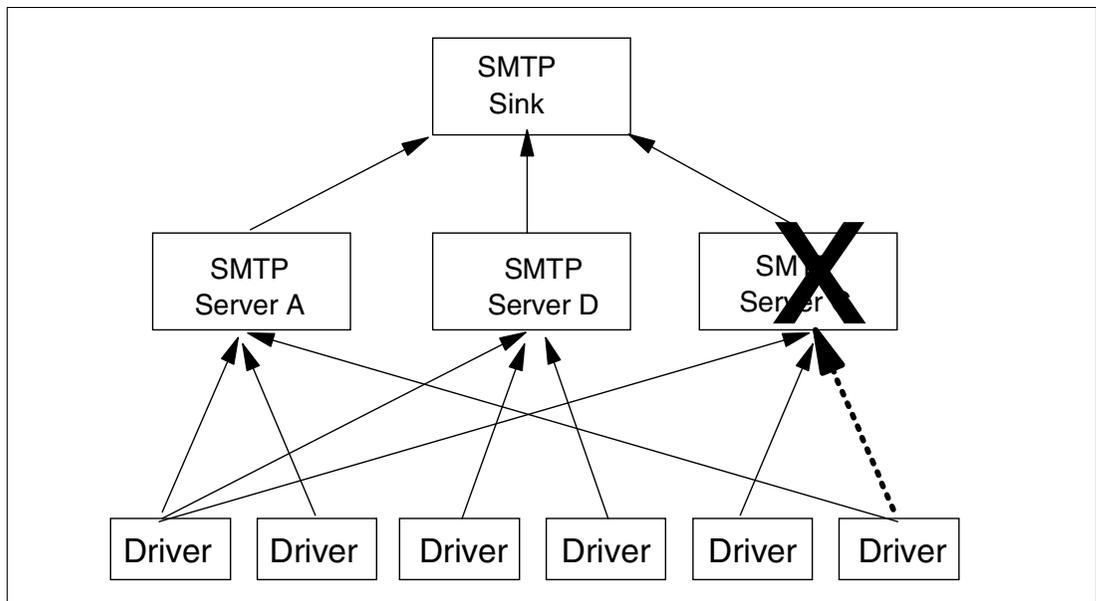


Figure 6-9 Round-robin DNS with one failed host

Load balancer

In recent years, the use of external workload directors has grown. These specialized devices can distribute load intelligently between several servers and handle failover and failback. The use of one such device allowed us to create a virtual mail server. Mail arriving for this virtual server is passed round-robin to each of the three real servers.

If for some reason one of the Sendmail servers should fail, the network switch stops sending new connections to that server (periodically checking to see if it has returned).

A benefit of this configuration is that it does not share the weakness of round-robin DNS solutions. Since only one IP address is published to the world, there is no reason to worry about cached entries when you add/remove members of the cluster.

Our experience with load balancing products has been generally positive. However, the equipment can be expensive and certainly adds another point of maintenance. To provide maximum protection and avoid single points of failure, two or more of the devices should be installed. Figure 6-10 shows an example of using a load balancer.

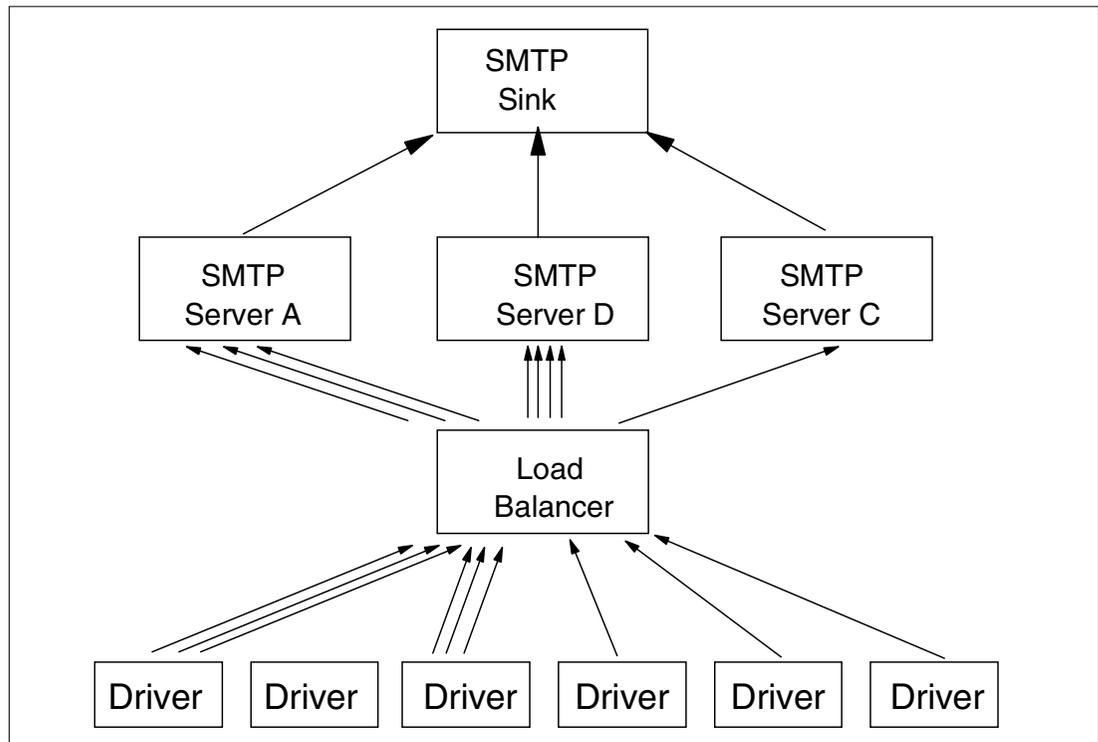


Figure 6-10 Using a load-balancer

MX-based failover

This is a traditional Sendmail/DNS configuration. A host name in a DNS can have one or more MX records. These records include a weighting factor. When other SMTP servers attempt to deliver mail to this host, they first look for an MX record and its relative weight. The MX record with the lowest weighting number is selected when there is more than one MX record. Mail is delivered to the first host that answers with the lowest weight/number.

In Figure 6-11, a domain has two MX records, one with a weight of 10 pointed at the Chicago office and one with a weight of 20 pointed at the New York office. Under normal circumstances, all the mail goes through the Chicago office, flowing over the corporate wide area network (WAN) for internal delivery.

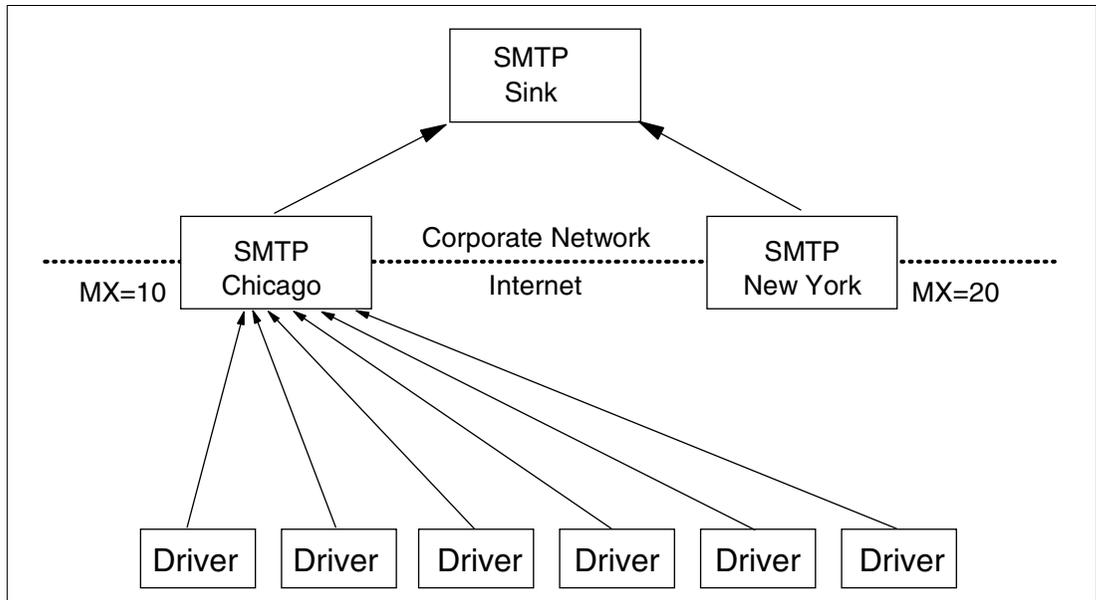


Figure 6-11 Two-way SMTP cluster with MX failover

If the Chicago office fails, or routes to the Chicago office fail, mail automatically flows via the New York office. Figure 6-12 illustrates this concept.

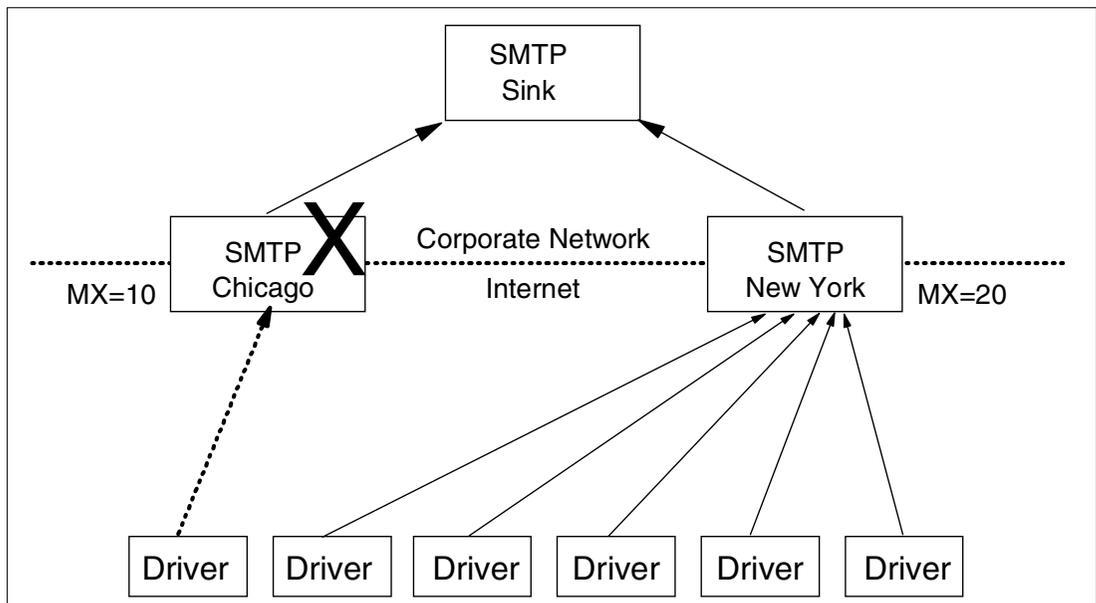


Figure 6-12 MX-based failover in action

If the Chicago office failed in the middle of an SMTP transaction, that transaction will fail (the dashed arrow). Because SMTP is a transaction, message integrity is assured. The sender times out, backs off, and resends the entire message later. If the Chicago office is still down, mail automatically flows through New York.

The main benefit of this method is that it's a very mature process, well documented, and understood. It also requires only minor configuration changes and no additional software or hardware. Unfortunately, the workload is not evenly distributed. Therefore, in practice under heavy load, the service may "flap" back and forth. MX solves availability problems, but not

scalability. The result is that you end up buying twice as much hardware to handle the possibility of a failure during peak traffic loads.

Hybrid solutions: The real world

Real world solutions tend to be hybrids drawing from all of the technologies described earlier. For example, Internet mail sent to IBM is delivered to one of three regional centers: Colorado, New York, or North Carolina. Through a combination of MX records for failover and round-robin DNS, IBM assures fast and reliable Internet mail. Here are the public DNS records:

```
us.ibm.com      preference = 10, mail exchanger = e22.nc.us.ibm.com
us.ibm.com      preference = 10, mail exchanger = e23.nc.us.ibm.com
us.ibm.com      preference = 10, mail exchanger = e24.nc.us.ibm.com
us.ibm.com      preference = 20, mail exchanger = e1.ny.us.ibm.com
us.ibm.com      preference = 10, mail exchanger = e2.ny.us.ibm.com
us.ibm.com      preference = 10, mail exchanger = e3.ny.us.ibm.com
us.ibm.com      preference = 10, mail exchanger = e4.ny.us.ibm.com
us.ibm.com      preference = 20, mail exchanger = e31.co.us.ibm.com
us.ibm.com      preference = 10, mail exchanger = e32.co.us.ibm.com
us.ibm.com      preference = 10, mail exchanger = e33.co.us.ibm.com
us.ibm.com      preference = 10, mail exchanger = e34.co.us.ibm.com
us.ibm.com      preference = 20, mail exchanger = e21.nc.us.ibm.com
```

Repeated DNS queries yield a rotating list of MX preferences:

```
us.ibm.com      preference = 10, mail exchanger = e2.ny.us.ibm.com
us.ibm.com      preference = 10, mail exchanger = e3.ny.us.ibm.com
us.ibm.com      preference = 10, mail exchanger = e4.ny.us.ibm.com
us.ibm.com      preference = 20, mail exchanger = e31.co.us.ibm.com
us.ibm.com      preference = 10, mail exchanger = e32.co.us.ibm.com
us.ibm.com      preference = 10, mail exchanger = e33.co.us.ibm.com
us.ibm.com      preference = 10, mail exchanger = e34.co.us.ibm.com
us.ibm.com      preference = 20, mail exchanger = e21.nc.us.ibm.com
us.ibm.com      preference = 10, mail exchanger = e22.nc.us.ibm.com
us.ibm.com      preference = 10, mail exchanger = e23.nc.us.ibm.com
us.ibm.com      preference = 10, mail exchanger = e24.nc.us.ibm.com
us.ibm.com      preference = 20, mail exchanger = e1.ny.us.ibm.com
```

Lab configuration

Our evaluations were all done using a *private switched network*. Each of the three SMTP servers was connected via 1000BT to fiber ports on a load balancing network switch. Each of the five SMTP load generators was connected to the same switch via a 100BT port. The switch is both a traditional layer two switch and a layer three/four workload balancer. In our configuration, all computers are in the same virtual local area network (VLAN) and in the same broadcast domain. We configured a virtual IP address on the switch. We also set up a policy for that virtual host that distributes connections to port SMTP (tcp/25) in a round-robin fashion between each of the three SMTP servers.

For administrative access, each SMTP server and one SMTP load generator were also attached via 100BT to the lab network. No testing was done over these links.

Each SMTP server had caching DNSs installed. They all had a replicated LDAP database (for some Sendmail maps). Figure 6-13 illustrates this configuration.

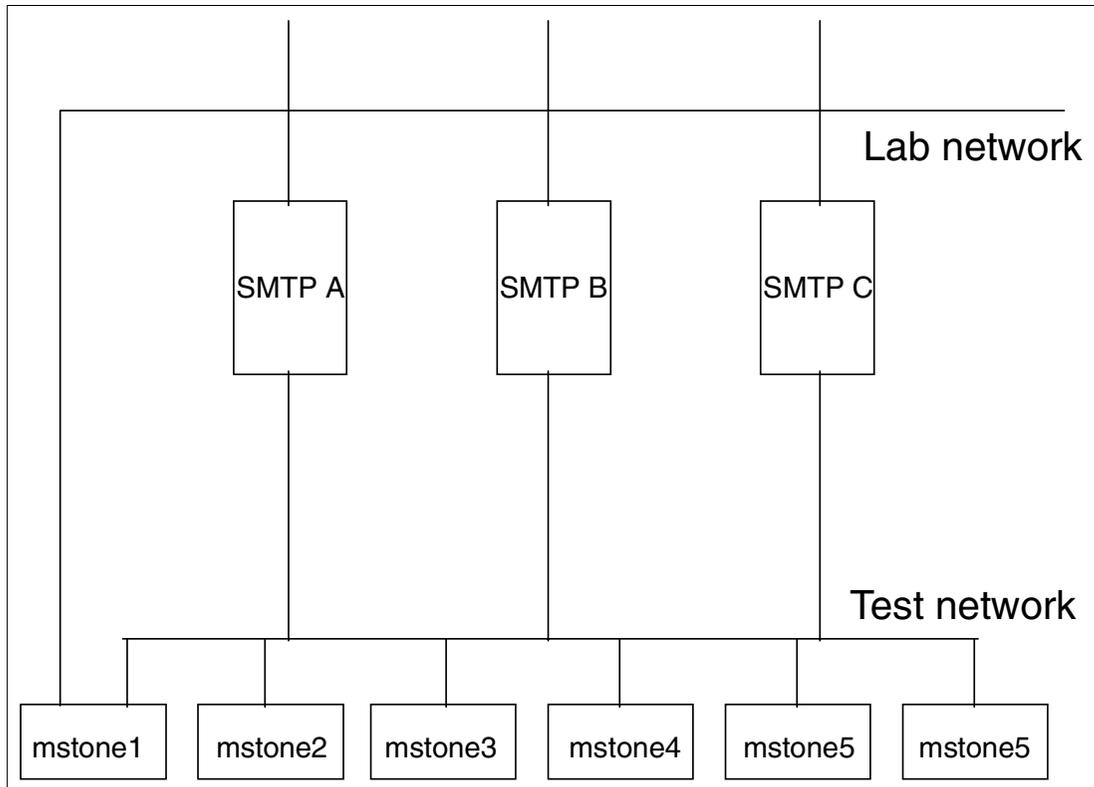


Figure 6-13 Sendmail cluster network

We used several techniques for proofing the network installation and deriving a baseline for maximum bandwidth. We ensured that there were no asymmetries. For the workloads we tested, actual network bandwidth was not really very significant. Anecdotally, it appeared that the TCP/IP stacks gave out long before the bandwidth did.

Building Sendmail with LDAP

Chapter 4, “Plumbing: Network infrastructure” on page 25, discusses setting up a centralized LDAP service. We realize that there may be cases where you want to use only LDAP as a stand alone e-mail directory. We have taken this opportunity to demonstrate an alternate method for installing and configuring OpenLDAP.

All our evaluations were done using the publicly available Sendmail version 8.12.0 beta 7. On Red Hat Linux, it builds without complaint. By default, it builds as an optimized “-O2” binary. Our only change was to add support for LDAP-based maps. We also found that we needed to force the Sendmail build to support the new shared memory feature. Here is the difference when using the build configuration file:

```
sendmail-8.12.0.Beta7/devtools/OS/Linux
1a2
> define('confENVDEF', '-DSM_CONF_SHM')
5c6,7
< define('confLIBS', '-ldl')
---
> define('confMAPDEF', '-DNIS -DMAP_REGEX -DLDAPMAP')
> define('confLIBS', '-L/usr/local/lib -llber -lldap -ldl')
```

Open LDAP installation and configuration notes

The LDAP server provides a Mail Local Address for the Sendmail service. For maximum speed with minimum network traffic, each mail server machine has a local LDAP server. We configured the servers as one master with two slaves. LDAP slaves maintain a read-only copy of the master database. All updates are pushed from the master server. Update traffic is non-existent in this test, so replication was not a performance consideration.

Building

OpenLDAP uses the well-known GNU configuration script to set up the source package. On our system, the package builds fine with the standard GNU recipe:

```
./configure
make
make install
```

We installed the binaries and configuration files under `/usr/local/etc/openldap`. The LDAP data files and runtime files are under `/usr/local/var`.

Configuration

Example 6-23 shows the annotated files from our installation.

Example 6-23 Master slapd.conf

```
# $OpenLDAP: pkg/ldap/servers/slapd/slapd.conf,v 1.8.8.6 2001/04/20 23:32:43 kurt Exp $
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
include      /usr/local/etc/openldap/schema/core.schema
include      /usr/local/etc/openldap/schema/cosine.schema
include      /usr/local/etc/openldap/schema/inetorgperson.schema
include      /usr/local/etc/openldap/schema/misc.schema
# LDAP directories use OID syntax to describe things. The syntax is described by these
# schema files. If the attribute type you need is not listed in these files, you're in a heap
# of trouble. The default slapd.conf includes only core.schema, so always check this.
# Define global ACLs to disable default read access.
# Do not enable referrals until AFTER you have a working directory
# service AND an understanding of referrals.
#referral    ldap://root.openldap.org
pidfile      /usr/local/var/slapd.pid
argsfile     /usr/local/var/slapd.args
repllogfile  /usr/local/var/slapd.repllog
#Locations for runtime information
# Load dynamic backend modules:
# modulepath /usr/local/libexec/openldap
# moduleload back_ldap.la
# moduleload back_ldbm.la
# moduleload back_passwd.la
# moduleload back_shell.la
#####
# ldbm database definitions
#####
database     ldbm
suffix       "dc=sequent,dc=com"
rootdn       "dc=ent,dc=sequent,dc=com"
# These two items define the Root Distinguished name and will be used by all the db
# utilites.
# Cleartext passwords, especially for the rootdn, should
# be avoid. See slappasswd(8) and slapd.conf(5) for details.
```

```

# Use of strong authentication encouraged.
rootpw      secret
#We're not very secure
# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd/tools. Mode 700 recommended.
directory   /usr/local/var/openldap-ldb
# Indices to maintain
index objectClass      eq
index mailLocalAddress pres,eq
index  cn               pres,eq
index  sn               pres,eq
index  mailroutingaddress pres,eq
# Indexing is always nice to have in a database. LDAP is no exception
replica host=slave3:389
        binddn="cn=Manager,dc=ent,dc=sequent,dc=com"
        bindmethod=simple credentials=secret
replica host=slave5:389
        bindmethod=simple credentials=secret
        binddn="cn=Manager,dc=ent,dc=sequent,dc=com"
#The replica entries point to our slave LDAP servers.

```

The slave is identical to the master, except for replication. Where the master has the replica directive, the slave has a directive indicating who can perform updates. Example 6-24 shows the slave configuration file.

Example 6-24 Slave slapd.conf

```

updatedn "cn=Manager,dc=ent,dc=sequent,dc=com"
Sample OID Notation for our schema
attributetype ( 2.16.840.1.113730.3.1.13
                NAME 'mailLocalAddress'
                DESC 'RFC822 email address of this recipient'
                EQUALITY caseIgnoreIA5Match
                SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

```

Notes on performance

During our experiments with Sendmail, we ran several tests using a set of systems to generate load and the MailStone. MailStone is a freely available performance measurement tool produced by Netscape, Inc. The MailStone tool is a collection of scripts and a mailclient binary.

Using the MailStone tool, we could describe a sophisticated work load with a mathematical distribution of number of recipients, message sizes, and mime attachments. The workload configurations are defined and interpreted on a test manager station. The work of generating the SMTP traffic is fanned out via remote shell and run on any number of mail client machines. We then set up multiple mail servers for our tests. We found the most performing set of options for multiple-server tests. The options are:

- ▶ Queue on RAM
- ▶ LogLevel 9 on RAM
- ▶ Shared Memory Enable
- ▶ Ident Disabled
- ▶ Delivery Mode set to Interactive

With these options, we demonstrated that we could build a high-performance, highly available Sendmail service. In our tests, the disk I/O was the most significant factor in the Sendmail system's overall performance. Configuring the server not to record logs, or to record logs to RAM disk, greatly improves performance. The mail queues are the other disk-intensive part of the Sendmail server process. Distributing the queues to multiple directories, putting the queue directories on the fastest available file system, or moving the queues to RAM disks, also improves performance significantly. Moving the mail queues to RAM disks may not be appropriate for many installations. In the case of an operating system crash or server hardware failure, the message delivery integrity of the Sendmail system is compromised.

We were surprised to learn that, for our workload, we could replace a commercial load balancing manager, with a dedicated Linux system running the user-land program `balance`. The economics of the Linux solution are compelling, but there may be operational advantages to using an appliance.

We saw doubling of performance when we went from one Sendmail server to two. However, adding a third server had no effect. Our load generation equipment was working hard, so it may be that we saturated the mail generation equipment. We also observed some TCP/IP problems under heavy load that may account for increases in connection error rate. Due to the limitation in our generation machines, we were unable to drive a load above the 10 million messages per day mark. We conclude that two IBM blade-based e-mail servers should be able to handle the mail load for the majority of businesses.

6.3 Instant messaging

In today's increasingly distributed world, organizations have embraced instant messaging as an important addition to e-mail and voice communications. Instant messaging allows colleagues working in different locations to quickly discuss business matters without having to pick up a telephone.

Instant messaging isn't just about exchanging short notes anymore. Today's robust applications now support extended features such as common whiteboards and quick file sharing.

6.3.1 Instant messaging's value to modern companies

Instant messaging has been called the next wave of corporate communication. Instant messaging does not offer new services or features that are not available in other technologies. It combines features in a highly useful and effective way. It provides a hybrid between the instant nature of talking on the telephone with e-mail's ability to hold multiple simultaneous, unrelated, conversations.

Most instant messaging systems provide two basic features:

- ▶ Buddy list management
- ▶ Instant chat

The buddy list is an interactive address book that lets you know when people are available for chat. Teleconferencing, an ancient tool of business, can be improved when used in conjunction with instant messaging by providing the ability to conduct private side chats.

6.3.2 Jabber

Jabber is an open Extensible Markup Language (XML) protocol for the real-time exchange of messages and presence between any two points on the Internet. Jabber provides the

standard functionality people expect of an instant messaging system: one-to-one chat, multi-user chat, the ability to subscribe to someone else's presence, and so on.

Jabber started as a small project by Jeremie Miller in 1998. Not long after, several core developers joined the project. This small team developed the *jabberd server*, Jabber clients for Windows and Linux, and gateways to the major instant messaging services such as AIM, ICQ, MSN, and Yahoo.

Late in 1999, Webb Interactive Services began sponsoring some of the core team. This helped to further speed up development. Most of the Jabber protocol was developed at this time. It evolved along with the jabberd server and early clients such as Winjab (which since deprecated in favor of Exodus) and Gabber. This early period of rapid change essentially came to an end in May 2000 with the release of jabberd 1.0.

Today, Jabber is the foundation for a variety of instant messaging applications. While the first application of Jabber technology is an asynchronous, extensible instant messaging platform, it has grown to support several applications because of its roots in XML.

Jabber servers are currently in use within a wide variety of environments, including:

- ▶ Small development teams
- ▶ Small to midsize company intranets
- ▶ Colleges and universities
- ▶ Web-based communities
- ▶ Internet service providers
- ▶ Large company intranets

Jabber clients

There are several Jabber clients, each one with a different range of capabilities. They cover the range of popular operating systems available today. At the minimum, these clients support basic chat. However, some have extended features such as group chat, file transfer, and common whiteboard that companies also find useful.

Not all Jabber clients need a Jabber server to work. They can also function in point-to-point mode or having one client contact another directly.

You can view the list of Jabber clients on the Web at:

<http://www.jabber.org/user/clientlist.php>

In this redbook, we look at Exodus for Windows and Coccinella for Linux.

Jabber servers

There are several Jabber server implementations with varying capabilities. At the very least, they support instant messaging, presence announcement, and profile storage. Additional features include such capabilities as SSL support, database integration, offline messaging, Microsoft Exchange and NetMeeting integration, and message archiving.

Some are open source projects. Several others are commercial implementations. In this redbook, we use the jabberd server.

For a full list of Jabber servers, see:

<http://www.jabber.org/admin/serverlist.php>

6.3.3 Running a Jabber server

The jabberd server is the original open-source server implementation of the Jabber protocol. It is the most popular software for deploying Jabber either inside a company or as a public instant messaging service.

Requirements

The following sections explain the requirements for running a Jabber server.

Hardware requirements

Hardware requirements depend on the number of users you want to support. jabberd has been tested to up to 10,000 simultaneous users. But remember that it was written as a proof-of-concept server, not as an industrial-strength server for large Internet service providers. For corporate intranets and small departments, with a user base of anywhere from 100 to 1,000, a Pentium®-class server with 512 MB RAM should be sufficient and well within the specifications of our blades.

In addition, a standard Linux server with default settings is limited to 1,024 simultaneous socket connections. You can handle more sockets by increasing the number of file descriptors under the proc settings.

Software requirements

Compiling jabberd requires make and, optionally, OpenSSL. These packages are part of the typical Red Hat Advanced Server installation.

Bandwidth requirements

Under normal usage, a Jabber server requires approximately 15 bits per second for each connected user. This means that a server with 1,000 concurrent users would consume about 15 Kb per second of bandwidth.

DNS requirements

jabberd works best if your server is defined with a fully qualified domain name (FQDN) in your DNS.

Firewall requirements (optional)

Jabber services use two ports:

- ▶ Port 5222 for client-to-server communications
- ▶ Port 5269 for server-to-server communications

Port 5223 is used for SSL connections. If you want Jabber clients to connect to your server, you need to ensure that TCP port 5222 is open. If you want users of your Jabber server to send messages to users on other Jabber servers, you need to ensure that TCP port 5269 is open.

If you plan to install gateways to other instant messaging systems, you need to open ports that are specific to those systems.

If you have a firewall between your Jabber server and any of the users of your server, make sure that your firewall timeout settings are appropriate for use with Jabber. Jabber users connect to the server using a persistent TCP socket on port 5222. Because the TCP socket is open as long as the user has a session on the server, firewall timeouts that are optimized for HTTP traffic may disconnect Jabber users prematurely.

Getting the software

We use version 1.4.2 version of the jabberd server. You can download this from the Web at:

<http://jabberd.jabberstudio.org/downloads/>

We build Jabber from source, which is jabber-1.4.2.tar.gz.

Compiling the server

Uncompress the source file in your working directory, and run the usual **configure** and **make** procedure as shown here:

```
% tar xzvf jabber-1.4.2.tar.gz
% cd jabber-1.4.2
% ./configure --enable-ssl
% make
```

Testing the server

Before you install the jabberd server, test to see if it works as explained in these steps:

1. Run the following command:

```
% ./jabber-1.4.2/jabberd/jabberd
```

You should see the output:

```
20020923T02:50:26: [notice] (-internal): initializing server
```

2. Open another terminal window and Telnet to port 5222 of your own server:

```
% telnet localhost 5222
```

It responds with the following messages:

```
Trying 127.0.0.1...
Connected to your-machine-name.
Escape character is '^]
```

3. Open an XML stream to start communicating with your jabberd server:

```
<stream:stream
  to='localhost'
  xmlns='jabber:client'
xmlns:stream='http://etherx.jabber.org/streams'>
```

You should receive this reply:

```
<?xml version='1.0'?> <stream:stream xmlns:stream='http://etherx.jabber.org/streams'
id='some-random-id' xmlns='jabber:client' from='localhost'>
```

4. Close the stream by sending the following message to the server:

```
<?xml version='1.0'?> <stream:stream xmlns:stream='http://etherx.jabber.org/streams'
id='some-random-id' xmlns='jabber:client' from='localhost'>
```

5. To shut down the jabberd server, simply kill the running jabberd process or press Ctrl+C on the window in which it is running.

Note: As you work with jabberd, you may see a message similar to the following example:

```
[notice] (update.jabber.org): bouncing a packet to jsm@update.jabber.org/1.4.2
```

This is Jabber attempting to look for updates and new versions of the server. This message typically appears when you run Jabber behind a firewall. This message is not critical. Jabber continues to operate. You can prevent the message from appearing by adding the following lines to jabber.xml:

```
<service id="update.jabber.org">
  <host>update.jabber.org</host>
  <null/>
</service>
```

For more information, see:

<http://www.pipetree.com/jabber/update.html>

Initial changes to the Jabber configuration file

The Jabber configuration file is jabber.xml. You can find a copy in the Jabber directory in which you compiled the server. There are several options in the configuration file, but you should only need to change one line:

```
<host><jabberd:cmdline flag="h">localhost</jabberd:cmdline>
```

Change this to the FQDN of your Jabber server:

```
<host><jabberd:cmdline flag="h">powx.lsp1.ibm.com</jabberd:cmdline>
```

By default, the jabberd program uses the jabber.xml file that is in the directory in which you compiled jabberd. If you want to use a Jabber configuration file that you placed elsewhere, you have to run Jabber with the `-c` option, followed by the location of the configuration file.

The spool directory

jabberd requires a directory in which to store files that it creates, most notably, the ID files of the users that you create. The location of this directory is specified in jabber.xml.

While you're still performing your tests, this is located in a subdirectory of the spool subdirectory of your Jabber directory. That subdirectory must have the name of your servers' FQDN, as defined in server.xml. You must create this directory:

```
% mkdir spool/powx.lsp1.ibm.com
```

Installing jabberd

The jabberd makefile does not come with an install option. If you want to achieve consistency in where you place your server applications, you must manually copy the program and the configuration file in the desired locations. Fortunately, this is an easy task because the jabberd server is self-contained.

Jabber also does not come with startup and shutdown scripts, so you have to create one.

Optional jabber.xml settings

The following sections discuss jabber.xml settings that you may find useful. You can find more information about these settings on the Web at:

<http://www.jabber.org/admin/adminguide.html>

Administrative privileges

Jabber administrators have the abilities to:

- ▶ View all users who are online
- ▶ Send out broadcast messages to all users who are logged in

The following lines set up administrative users:

```
<admin>
  <read>hamlet@shakespeare.com</read>
  <read>macbeth@shakespeare.com</read>
  <write>kinglear@shakespeare.com</write>
</admin>
```

Administrators with read privileges can only view all online users. Administrators with write privileges can send out broadcast messages, in addition to viewing users who are online.

To view all online users, an administrator sends the following message to the server:

```
<iq type='get' to='yourhostname/admin'>
  <query xmlns='jabber:iq:browse' />
</iq>
```

To broadcast a message, an administrator sends the following message:

```
<message to="yourserver.com/announce/online">
  <body>This is a broadcast message!</body>
</message>
```

Disabling server-to-server communications

To disable server-to-server communications, remove the following lines.

These lines are for the external DNS resolution:

```
<service id="dnsv">
  <host/>
  <load><dnsv>dnsv/dnsv.so</dnsv></load>
  <dnsv xmlns="jabber:config:dnsv">
    <resend service="_jabber._tcp">s2s</resend>
    <resend>s2s</resend>
  </dnsv>
</service>
```

These lines handle server-to-server communications:

```
<service id="s2s">
  <load><dialback>dialback/dialback.so</dialback></load>
  <dialback xmlns='jabber:config:dialback'>
    <ip port="5269"/>
  </dialback>
</service>
```

6.3.4 Using Jabber clients

Let's look at some Jabber clients now, specifically Coccinella for Linux and Jabber for Windows.

Coccinella

Coccinella is a whiteboard program written in Tcl/Tk that uses the Jabber protocol as the underlying transport. It was written by Mats Bengtsson. You can find it on the Web at:

<http://hem.fyristory.com/matben/>

Because it was written in Tcl/Tk, Coccinella is quite portable. Aside from Linux, it also runs on other UNIX variants, Windows, and Macintosh. It features a basic drawing area for sharing diagrams and supports extensions for audio and video. It also supports double-byte languages.

Since it's a graphical application, you need to run this in an X environment.

Requirements

Coccinella requires the Tcl and Tk libraries to be installed in your system. Most Linux distributions have these libraries, so this is not a problem.

For some reason, the version of Coccinella we used also required a sound card properly configured. It did not connect properly if it could not send a message to `/dev/mixer` or `/dev/dsp`.

Installing Coccinella

Download a copy of Coccinella from the Web at:

<http://hem.fyristory.com/matben/download>

Unpack the archive (`Whiteboard-0.94.3.tar.gz`) and enter the directory. You don't need to compile anything. Simply type the following text to invoke the program:

```
./Whiteboard.tcl
```

The welcome page opens.

Coccinella is fairly intuitive. You can step through the questions in the windows. It is important to know the FQDN of your Jabber server. Coccinella prompts you whether the user you're logging in is already registered on the server. If not, it gives you the opportunity to do so.

Logging in

After that, it's fairly simple. Click the **Connect** button and log in using your created user name to the server. Figure 6-14 shows the Login window.



Figure 6-14 Logging in to Coccinella

Adding a buddy

When you first begin, you won't have any other users in your roster. Click **Jabber-> Add new user...** The Add New User window (Figure 6-15) opens that asks your for the name you want to add. Use the format `user@jabberservername` to add a new user. You may want to add a nickname as well.



Figure 6-15 Adding a new user

This does not immediately add the user to your list. The other user must agree to be added. Instead this sends a message to the other user, asking them if you can add the name to the list. This happens on the Subscribe window (Figure 6-16).



Figure 6-16 *Subscribing*

If the other party agrees, the name appear in your roster. If you closed your roster window, click **Jabber-> Roster/Services**. The roster window should open with the names of the people in your list.

Chat in Coccinella

Click **Jabber->Chat**.... This opens a window that asks you the name of the party to whom you wish to talk. Type the name and click **OK**. A chat window (Figure 6-17) opens on both your desktop and the desktop of the called person.

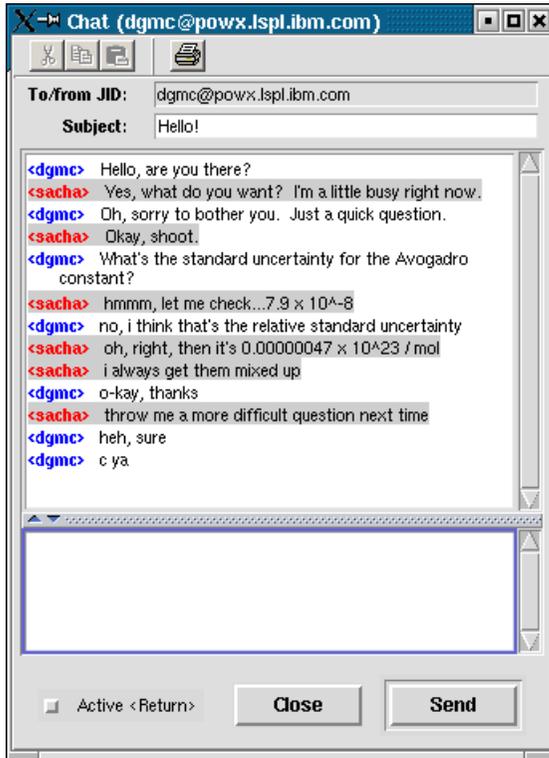


Figure 6-17 Chatting in Coccinella

Note: There's a bug in Coccinella that flags an error when you add the name of a user in the user@jabberserver format in the starting Chat dialog. To work around this (until it's fixed), type a deliberately wrong name in the Start Chat window. Then use the correct name in the To/from JID: field of the chat window.

Coccinella whiteboard

An attractive feature of Coccinella is its whiteboard function. You can make some drawings and send them to another user.

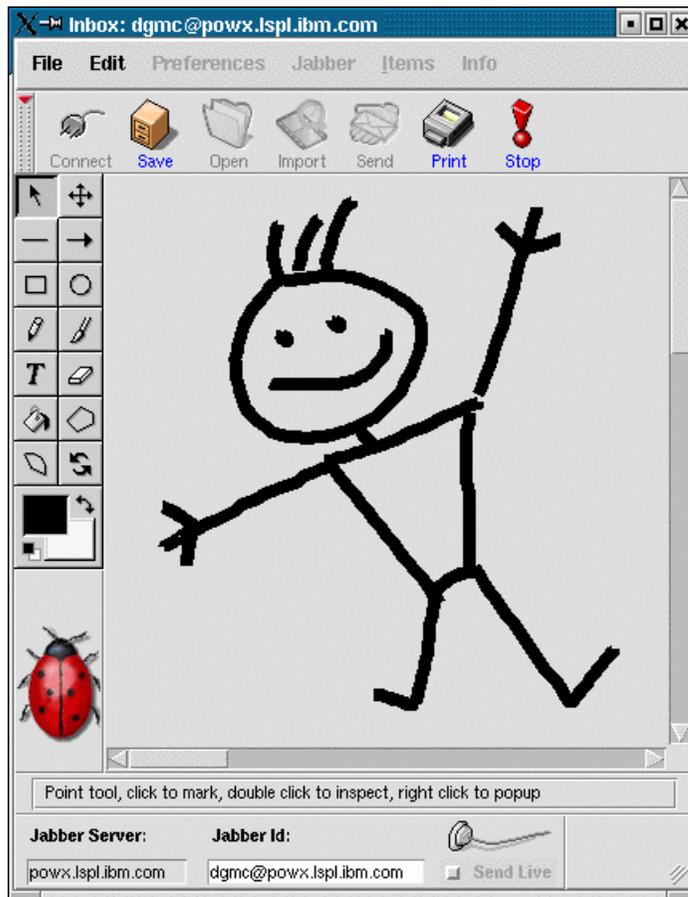


Figure 6-18 *Cocinella whiteboard*

You must understand that this is not a real-time whiteboard, where the figure appears on the other user's screen as you are drawing it. Instead, it is sent to the other user's inbox.

Thoughts on Coccinella

Coccinella was a very easy Jabber client to install and configure. What's particularly amazing about it is that it was written mostly in Tcl, a scripting language. It's a good showcase of how a functional Jabber application can be written using a relatively simple tool.

Coccinella still has a couple of minor bugs (pun not intended) mentioned above that may not appeal to less determined users. We feel that these bugs are easily remedied and will be so in subsequent releases.

Exodus

Exodus is a Jabber client that was developed as a successor to Winjab, the first Jabber client for Windows. Jabber is a very basic chat and presence client. It doesn't have too many fancy features, and by the same token, is quite easy to set up.

Installing Exodus

Download a copy of Exodus from the Web at:

<http://www.jabberstudio.org/projects/exodus/releases/>

Simply run the executable, which start an installation wizard. It's pretty straightforward.

Unfortunately, Exodus doesn't automatically place any icons in your desktop or Start menu. You can either create your own shortcuts for Exodus or invoke it directly from C:\Program Files\Exodus\exodus.exe.

Using Exodus

The first window you see upon starting Exodus is the login window (Figure 6-19). Fill this in as shown in the example.

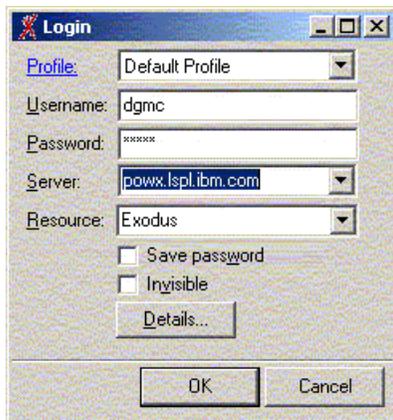


Figure 6-19 Exodus login window

The Exodus roster (Figure 6-20) immediately opens after you log on successfully.



Figure 6-20 Exodus roster

To start chatting, double-click a user name. Figure 6-21 shows an example chat session.

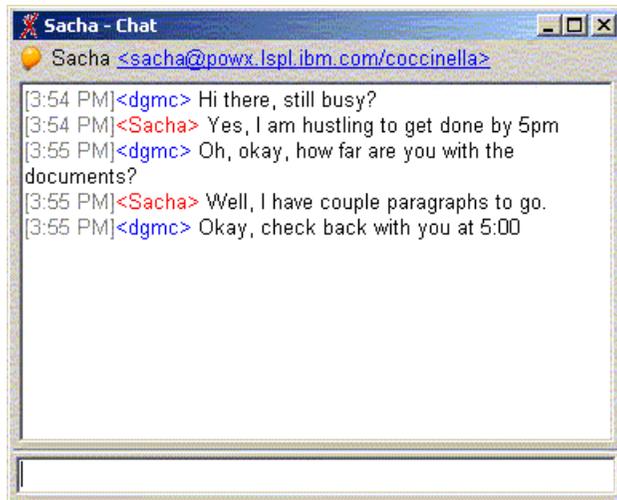


Figure 6-21 Exodus chat

Thoughts on Exodus

Exodus is a very basic chat client. It's not very fancy, but it does the job. For small organizations whose needs don't go beyond instant messaging, it's sufficient.

6.3.5 Considerations for using jabberd for an intranet

We've only scratched the surface of jabberd. There are a couple of other points to consider when you're using jabberd in an intranet.

Disabling registration

The default jabberd installation allows users to register in the user database as they please. This is not a good thing in an organization where identities are fixed, typically in some database or LDAP directory. You may then want to disable registration on jabberd.

To disable registration, comment out the following lines in jabber.xml:

```
<!--  
<register notify="yes">  
  <instructions>  
    Choose a username and password  
    to register with this server.  
  </instructions>  
  <name/>  
  <email/>  
</register>  
-->
```

Comment out these lines as well:

```
<!--  
<mod_register>./jvm/jvm.so</mod_register>  
-->
```

Centralized authentication schemes

If you disable registration, you obviously need another way to generate user accounts. The obvious choice is LDAP, but unfortunately, the integration is not mature yet. As of this writing, a design document for this project is being generated. The resources are on the Web at:

<http://xdblldap.tigris.org/design/ldap-jabber-integration.htm>

The following Web site shows code by which centralized authentication may be achieved:

<http://download.jabber.org/contrib/>

This includes:

- ▶ LDAP in various forms
- ▶ PAM
- ▶ SQL

However, none of these are officially part of the Jabber canon as yet.

Farming Jabber

For reliability and stability, Jabber can be implemented in a server farm. There are challenges, here, of course. Inter-server communication must be handled, as well as maintenance of state. Unfortunately, this is beyond the scope of the redbook at this time. Details of this are documented in the “Jabber Server Farming HOW-TO” by Ryan Eatmon. You can find this on the Web at:

<http://www.tldp.org/HOWTO/Jabber-Server-Farming-HOWTO/index.html>

Communicating with other instant messaging systems

While a company may choose to use jabberd internally, it may also need to communicate with users on other instant messaging networks. For example, instant messaging can be used as a customer support channel. Rather than maintain multiple instant messaging clients for its support personnel, a company can aggregate instant messaging communications internally with Jabber and use that to talk to other instant messaging systems such as AOL, Yahoo!, MSN, or ICQ. Modules that allow this communication are called *gateways* or *transports*.

Some important subprojects in this regard are:

- ▶ AIM/ICQ-Transport for Jabber
<http://aim-transport.jabberstudio.org/>
- ▶ JIT: Jabber ICQ Transport
<http://jit.jabberstudio.org/>
- ▶ MSN-Transport for Jabber
<http://msn-transport.jabberstudio.org/>
- ▶ Jabber Yahoo! Transport
<http://yahoo-transport.jabberstudio.org/>

You must realize that these other instant messaging systems are proprietary. Their owners may choose to “update” their protocols from time to time. The gateways, therefore, cannot be guaranteed to work on all configurations.

6.3.6 Extending Jabber

Jabber is really more than just an instant messaging platform. Think of it instead as a platform for exchanging XML messages between two points in real-time. The possibilities become much more diverse:

- ▶ **Machine-human communication:** You can program automated bots to keep track of events and notify you if certain triggers are activated, for example, when stock prices hit a certain level or when you receive an e-mail message.
- ▶ **Human-machine communication:** You can interface Jabber to a database and query it for information, such as telephone numbers or dictionaries
- ▶ **Machine-machine communication:** Instead of applications that write to TCP sockets, you can exchange information between machines using XML over Jabber. Programming becomes much more simplified.

The following Web page gives a list of programming interfaces for Jabber:

<http://www.jabber.org/developer/librarylist.php>

The major open-source programming languages are well-represented, including PHP, Perl, Python, and Java.

For a good view of the activities around Jabber, the central repository is currently located on the Web at:

<http://www.jabberstudio.org/project/>

Browse this site to see the projects and possibilities around Jabber.

For more information on extending Jabber programmatically, see *Programming Jabber: Extending XML Messaging* by D.J. Adams. A sample chapter is available on the Web at:

<http://www.oreilly.com/catalog/jabber/chapter/ch05.html>



Living spaces: Applications and portal server

Applications are like food to be consumed by guests of your home. You need a place to prepare and serve your application, as with food. In the world of the Web, this place is the application server. This chapter offers information that you can use to do everything from remodel your existing dining room, to build a world class kitchen.

7.1 Web applications

When you talk about *Web applications* today, the common understanding is that of an application that is deployed on a Web server and accessible via a Web client over the Internet or an intranet. There are even several tools for developing Web applications, such as PHP, Perl, JavaServer Pages (JSPs), Servlets, Python, Zope, and Nuke.

It's also important to look at the origins of the term Web applications. The term was popularized by the Java Servlet Specification released by JavaSoft in early 1997. When you talk about Web application servers today, the context that many people associate it with is with Java Web application servers.

According to the Java Servlet Specification 2.3, a Web application “is a collection of servlets, HTML pages, classes, and other resources that make up a complete application on a Web server.” Web application runs in a Java container within a Web server.

A Web application may consist of the following items:

- ▶ Servlets
- ▶ JSP
- ▶ Utility classes
- ▶ Static documents (HTML, images, sounds, etc.)
- ▶ Client side Java applets, beans, and classes
- ▶ Descriptive meta information, which ties all of the above elements together

7.1.1 Servlets

The basic unit of deployment of a Java Web application is the servlet. A servlet is a Java Web component that generates dynamic content. As with other Java components, servlets are compiled into bytecode that can be loaded and run by a Java-enabled Web server.

7.1.2 JavaBeans

JavaBeans (beans baked in Java) is a class framework designed to allow communication between Java components running in different Java virtual machines (JVMs). The JavaBean classes also provide a framework that makes it easy for a visual development tool to manipulate Java classes and components. JavaBeans are similar, but quite different, to traditional Java Applets. Beans provide several properties that set them apart from applets:

- ▶ **Introspection:** This allows a bean to query its own interface, and provides standard interface for development tools to inspect and managed Java components.
- ▶ **Events:** JavaBean events allow JavaBeans to communicate with each other.
- ▶ **Persistence:** This is not like object persistence. This feature allows design time development tool bean customizations to be saved, transported, and then retrieved.

7.1.3 JavaServer Pages

JavaServer Pages are a Java-based scripting technology. From a conceptual standpoint, JSP is similar to other so called “server side” scripting languages. JSP is actually an extension of the servlet specification. It provides a high level, extensible bridge between server-side Java, Java containers, Hypertext Markup Language (HTML), DHTML, and client-side Java. JSPs and their code are often commingled with client-side Java, or JavaScript, but the JSP execution model is radically different.

JavaServer Page code is Java code embedded within an HTML page. When a Web document with JSP code is served up by a Java-enabled Web server, the embedded code is first transformed into a Java servlet running within its own servlet container context. This behavior was designed to provide both very high scalability and security between JSP page instances. In order for this process to perform, it is broken up into two independent phases. These phases are *translation* and *request*.

The translation happens once. Translation is the act of parsing a JSP, creating the container, and compiling the resultant Java servlet. Transition of a page usually occurs upon its first request. Translation can also occur during the deployment of a JSP, but this feature depends on your application server. After translation is complete, the generated servlet is used to process any Hypertext Transfer Protocol (HTTP) requests to the Uniform Resource Locator (URL).

At the time of writing this document, JavaServer Pages Specification Version 2.0 was in its proposed final draft.

7.1.4 Containers

Containers, also called *servlet engines*, are Web server extensions that provide servlet functionality. A servlet container can be built directly into a Web server or installed as an add-on component. All servlet containers must support HTTP as a protocol for requests and responses. The minimum required version is HTTP/1.0, although HTTP/1.1 specification is now more common.

7.2 Tomcat

Tomcat is the reference implementation for the Java Servlet and JavaServer Page specifications. It is a fully functional servlet engine that is used in several production Web applications. Tomcat is maintained by the Apache Software Foundation (ASF) as part of its Jakarta project.

7.2.1 A brief history of Tomcat

Tomcat is the unification of two Java Web server projects, the Java Servlet Development Kit (JSDK) from Sun and Jserv from the Apache Java Group. During the early days of JSP 1.0 and Servlet 2.0, Sun had its own JSDK. Sun integrated this kit tightly with its own 100% pure Java Web Server product. At the same time, the developers of the open-source Apache Web server were working on Jserv, a JSP or servlet engine that was compatible with the latest servlet and JSP application programming interfaces (APIs) but would work with the very popular Apache.

The JSDK project focused on compliance to specifications, and the Apache JServ engine focused on delivering high performance. However, it soon became obvious that there was significant overlap between the two projects. Many users wanted to work with the a fully compliant Web application server, but were frustrated with performance problems of the JSDK. On the other hand, JServ users wanted closer adherence to reference standards.

To unify the two projects, Sun donated the source code of the servlet and JSP reference engine implementations to the ASF. The ASF created subgroup called the Jakarta project to focus on the servlet and JSP engines and to merge the two Web application server implementations. You can learn more about this project on the Web at:

<http://jakarta.apache.org>

Jakarta ended up handling all of the projects formerly under the auspices of the Apache Java group. All of the projects under the Jakarta are Java language open-source projects that support the Apache Software License. Tomcat is one of those projects.

Tomcat versions

The Tomcat 3.x series implemented the JSP 1.1 and Servlet 2.2 API specification. Tomcat 4.x was the first release to employ the new integrated, high-performance architecture. It will serve as the reference implementation for the JSP 1.2 and Servlet 2.3 API specification. Tomcat 5 implements the Servlet 2.4 and JavaServer Pages 2.0 specifications. It adds features that make it a useful platform for developing and deploying Web applications and Web services.

7.2.2 Diving into Tomcat

It's fairly easy and straightforward to get started with Tomcat. Pre-compiled binaries of Tomcat in RPM format are available for download from the Jakarta project Web site. Since Tomcat already has built-in Web server functionality, you can immediately start working after you install it.

Required files

In this part of the redbook, we work with Tomcat 4.1.24. Go to the following Web site:

<http://jakarta.apache.org/builds/jakarta-tomcat-4.0/release/v4.1.24/rpms/>

Download the following files from the site:

- ▶ tomcat4-4.1.24-full.2jpp.noarch.rpm
- ▶ tomcat4-4.1.24-full.2jpp.src.rpm
- ▶ tomcat4-admin-webapps-4.1.24-full.2jpp.noarch.rpm
- ▶ tomcat4-admin-webapps-4.1.24-le.2jpp.noarch.rpm
- ▶ tomcat4-webapps-4.1.24-full.2jpp.noarch.rpm
- ▶ tomcat4-webapps-4.1.24-le.2jpp.noarch.rpm

You also need Java Developer Kit. This should be included in the Red Hat Advanced Server distribution. Otherwise, you can download a copy from:

<http://www-106.ibm.com/developerworks/java/jdk/index.html>

Installation

Perform these steps as the root user:

1. If the Java Developer Kit is not yet installed, install it using the following command:

```
/bin/rpm -ivh IBMJava2-SDK-1.3.1-5
```

This installs the JDK into the /opt/IBMJava2-13/ directory.

2. Install Tomcat. Enter the following command:

```
/bin/rpm -ivh jakarta-tomcat-4.1.24-full.2jpp.noarch.rpm
```

This installs Tomcat in the /var/tomcat4/... directory. Configuration files are in /etc/tomcat4.

3. Add some environment variables that will be required by the JDK and Tomcat. These variables are:

```
JAVA_HOME  
CATALINA_HOME
```

We set these values in the /etc/profile file so that they are loaded by the system at startup and every time a user logs in.

4. Add the following lines to /etc/profile:

```
JAVA_HOME=/opt/IBMJava2-131
CATALINA_HOME=/var/tomcat4
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC JAVA_HOME CATALINA_HOME
```

At this point, Tomcat is already installed into the system, but it is not active yet. You have to start it manually. If you're new to Tomcat and Java Web application servers, you may want to hold off on that until you read through the next steps to have a better user experience with the software. Tomcat created several directories when it was installed. Each of them has a specific function. Table 7-1 lists the directories and their descriptions.

Table 7-1 Tomcat directory structure

Directory name	Description
/var/tomcat4/bin	Contains startup/shutdown... scripts.
/var/tomcat4/conf (linked to /etc/tomcat4/)	Contains various configuration files, including server.xml (Tomcat's main configuration file) and web.xml that sets the default values for the various Web applications deployed in Tomcat.
/var/tomcat4/doc	Contains miscellaneous documents regarding Tomcat.
/var/tomcat4/lib	Contains various Java Archive (JAR) files that are used by Tomcat. On UNIX, any file in this directory is appended to Tomcat's classpath.
/var/tomcat4/logs	This is where Tomcat places its log files.
/var/tomcat4/src	The servlet APIs source files. These are only the empty interfaces and abstract classes that should be implemented by any servlet container.
/var/tomcat4/webapps	Contains sample Web applications.

Tomcat automatically creates the following directories when you start it:

- ▶ **Work:** Automatically generated by Tomcat. This is where Tomcat places intermediate files (such as compiled JSP files) during its work. If you delete this directory while Tomcat is running, you will not be able to execute JSP pages.
- ▶ **Classes:** You can create this directory to add additional classes to the classpath. Any class that you add to this directory will find its place in Tomcat's classpath.

Starting and stopping Tomcat

The RPM binary installation of Tomcat already includes startup and shutdown scripts. You can access these scripts using standard System V commands.

To start Tomcat, type the following command:

```
# service tomcat4 start
```

To shut down Tomcat, type the following command:

```
# service tomcat4 stop
```

You can start Tomcat automatically whenever you boot your system by marking it in Red Hat's system services menu:

```
# chkconfig --level 345 tomcat4 on
```

Configuring Tomcat for servlet and JSP development

As Tomcat is installed, you can't work with the Tomcat Web application server. The binary file from the Jakarta Web site does not include any servlets, JSPs, or even plain HTML pages. If

you jump ahead, start Tomcat, and point your Web browser at your server, you will most likely receive an error message.

Later, you see how to properly deploy a Java Web application on Tomcat. Deployment is pretty straightforward since you simply drop a Web application archive (WAR) file into the Tomcat working directory. In the meantime, if you're just starting out, you want to configure your Tomcat server for development purposes.

You must log in as root to edit the following files:

1. Enable the root context in the `/etc/tomcat4/conf/server.xml`.

The `/etc/tomcat4/conf/server.xml` file is the main configuration file of Tomcat.

The root context is the default Web application in Tomcat, essentially the root directory for your entire Web application server. It is convenient to use when you are learning about servlets and JSPs. It is disabled by default in Tomcat 4.1.12 and later versions.

To enable it, uncomment the following line in `/etc/tomcat4/conf/server.xml`:

```
<Context path="" docBase="ROOT" debug="0"/>
```

2. Enable the invoker servlet.

The invoker servlet lets you run servlets without making changes to the `WEB-INF/web.xml` file in your Web application. With the invoker servlet enabled, you can drop your servlet into `WEB-INF/classes` and use the following URL:

```
http://host/servlet/ServletName
```

The invoker servlet is extremely convenient when learning and even during initial development, but you do not want it on at deployment time.

Up until Tomcat 4.1.12, the invoker was enabled by default. Because of a security flaw, which allowed Web browsers to use the servlet to see the source code of JSP pages, it has been disabled.

To enable the invoker servlet, uncomment the following servlet-mapping element in `/etc/tomcat4/conf/web.xml`:

```
<servlet-mapping>
<servlet-name>invoker</servlet-name>
<url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
```

3. Turn on servlet reloading.

In a Tomcat production environment, servlets are loaded by Tomcat when they are first invoked. From then on, Tomcat assumes that the servlets will not be changed. If any changes are made to the servlets, Tomcat must be restarted.

This situation, of course, is not ideal for a development environment. We want Tomcat to check the modification dates of the class files of requested servlets and reload the ones that have changed. In a development environment, it's convenient to have. In a production environment, this degrades performance.

To turn on servlet reloading, edit `/etc/tomcat4/conf/server.xml`. Find the following comment:

```
<!-- Define properties for each web application. This is only needed if you want to set
non-default properties, or have web application document roots in places other than the
virtual hosts's appBase directory. -->
```

Insert the following line after it:

```
<DefaultContext reloadable="true"/>
```

Testing Tomcat

At this point, Tomcat should be ready to go. Before you try any servlets or JSPs, check whether Tomcat starts properly.

To start the Tomcat service, enter the following command:

```
# service tomcat4 start
```

Barring any errors in the server.xml file, your Tomcat server should start up properly.

To stop the Tomcat service later, type:

```
# service tomcat4 stop
```

Test a simple HTML page. Put a simple HTML page into the `/var/tomcat4/webapps/ROOT/` directory. This directory should be created by Tomcat when you start it:

```
<HTML>
<HEAD><TITLE>Hello, world!</TITLE></HEAD>
<BODY>
<H1>Hello, world!</H1>
</BODY>
</HTML>
```

Access this page through this URL:

```
http://hostname:8080/yourwebpage.html
```

To test a simple JSP page, place the following sample JSP code into the `/var/tomcat4/webapps/ROOT/` directory:

```
<HTML>
<HEAD><TITLE>Hello with JSP</TITLE></HEAD>
<BODY>
<H1>
<%
if (request.getParameter("name") == null) {
out.println("Hello, world!");
} else {
out.println("Hello, " + request.getParameter("name"));
}%>
</H1>
</BODY></HTML>
```

You can access this page through the following URL:

```
http://hostname:8080/yourjspage.jsp
```

A simple servlet development environment

The previous section explains how to deploy a static HTML page and a JavaServer Page into the Tomcat server. This section shows you how to deploy simple servlets.

At the most basic form, you place servlets into a `WEB-INF/classes` directory where Tomcat can access them. Unlike simple HTML pages and JSPs that Tomcat automatically serves in their native forms, you have to compile servlets into Java class files.

If you're just starting, you need to set up a basic development environment to test your servlets. You may want to set this up on your own development environment instead of on BladeCenter itself. Follow these steps:

1. Set up a Tomcat environment on your client machine similar to the steps explained in the previous sections.

2. Create a development directory where you will write your servlet and JSP source code, for example: `/home/user/servlet-devel`.
3. Set your CLASSPATH to include the servlet libraries. Because servlets and JSP are not part of the Java2SE platform, you have to identify the servlet classes to the compiler. To do this, modify `/etc/profile` and add the following lines:

```
CLASSPATH=/var/tomcat4/common/lib/servlet.jar
export CLASSPATH
```

Now, every time you log on to the system, the environment variable is automatically loaded.

4. To activate the environment variable, log off and log on again.

Some simple servlets

In this section, we write some simple servlets to deploy to our Tomcat server:

1. Create the Java source file for the servlet. In your `/home/user/servlet-devel` directory, create the following file named *HelloWorld.java*:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out=res.getWriter();
        out.println("<HTML>");
        out.println("<HEAD><TITLE>Hello, world</TITLE></HEAD>");
        out.println("<BODY>");
        out.println("Hello, world!");
        out.println("</BODY></HTML>");
    }
}
```

2. Compile the servlet:

```
% javac HelloWorld.java
```

This should create a Java class file called `HelloWorld.class`.

3. Copy the Java class file to the `WEB-INF/classes` directory:

```
cp /home/user/servlet-devel/HelloWorld.class /var/tomcat4/webapps/ROOT/WEB-INF/servlets
```

4. Access the servlet from the URL:

```
http://hostname:8080/servlets/HelloWorld
```

If you enabled servlet reloading, you should see your servlet output displayed. If you did not, you may need to restart your Tomcat server.

Additional resources

Marty Hall, author of *Core Servlets and JavaServer Pages* and *More Servlets and JavaServer Pages*, maintains Web sites for both these books at:

- ▶ <http://www.coreservlets.com>
- ▶ <http://www.moreservlets.com>

You can download the complete text for *Core Servlets and JavaServer Pages* as a trial offering from his Web sites. Marty also has several other excellent resources for free download, such as presentation materials and source code.

Lajos Moczar maintains a series of short guides to Tomcat and related technologies on his Web site at:

<http://www.galatea.com/flashguides/>

See his links on Tomcat at:

<http://www.galatea.com/flashguides/tomcat-overview.xml>

The O'Reilly Web site has published several short articles on Tomcat by James Goodwill. Some were written two years ago, but are still relevant today. We recommend that you cross-check against newer documentation from the Java and Tomcat Web sites. In particular, look for "Installing and Configuring Tomcat", which you can find on the O'Reilly site at:

<http://www.onjava.com/pub/a/onjava/2001/03/29/tomcat.html>

7.2.3 Java Web applications

A Java Web application is rooted in a specific path within a Web server. For example, a catalog application may be located at:

<http://www.mycorp.com/catalog>

All requests that start with this prefix are routed to what we call a *ServletContext*.

ServletContext

The ServletContext is defined in the server.xml file of Tomcat. It basically represents all the information about the Web application. For example, our application <http://www.mycorp.com/catalog> is represented by the following ServletContext in our Tomcat server's server.xml file:

```
<Context path="/catalog" docBase="catalog" debug="0" reloadable="true" />
```

The first parameter, `path="/catalog"`, tells the servlet container that all requests with `/catalog` appended to the server's URL belong to the catalog Web application. The second parameter, `docBase="catalog"`, tells the servlet container that the Web application exists in the `/catalog` directory.

Directory structure

A Web application exists within a structured hierarchy of directories. The top level of this hierarchy serves as the document root for files that are part of the application. For example, for a Web application with the context path `/catalog` in a Web container, the `index.html` file at the base of the Web application hierarchy can be served to satisfy a request from `/catalog/index.html`.

The directory structure looks this:

- ▶ **/catalog**: This is the root directory of the Web application. All JSP, HTML, and graphics files are stored here.
- ▶ **/catalog/WEB-INF**: This directory contains all resources related to the application that are not in the document root of the application. No files contained in this directory can be served directly to a client. See the description for WEB-INF in the following section.
- ▶ **/catalog/WEB-INF/classes**: This directory is where servlet and utility classes are located.
- ▶ **/catalog/WEB-INF/lib**: This directory contains JAR files that the Web application depends on. For example, this is where you place a JAR file that contains a JDBC driver.

Here is a sample listing of files of a typical Web application:

```
/catalog/index.html
/catalog/howto.jsp
/catalog/feedback.jsp
/catalog/images/banner.gif
/catalog/images/jumping.gif
/catalog/WEB-INF/web.xml
/catalog/WEB-INF/lib/jspbean.jar
/catalog/WEB-INF/classes/com/mycorp/servlets/MyServlet.class
/catalog/WEB-INF/classes/com/mycorp/util/MyUtils.class
```

The WEB-INF directory

The WEB-INF directory is a special directory that exists within the application hierarchy. It contains all things related to the application that are not in the document root of the application. The WEB-INF node is not part of the public document tree of the application and no file contained in the WEB-INF directory may be served directly to a client by the container.

However, the contents of the WEB-INF directory are visible to servlet code. If an application developer needs access, from servlet code to application-specific configuration information that they do not want to expose to the Web client, they may place it under this directory.

The contents of the WEB-INF directory are:

- ▶ **/WEB-INF/web.xml deployment descriptor**
- ▶ **/WEB-INF/classes/ directory for servlet and utility classes:** The classes in this directory must be available to the application class loader.
- ▶ **/WEB-INF/lib/*.jar area for Java Archive (JAR) files:** These files contain servlets, beans, and other utility classes useful to the Web application. The Web application class loader must be able to load classes from any of these archive files.

The Web application classloader must load classes from the WEB-INF/ classes directory first and then from library JARs in the WEB-INF/lib directory.

Web Application Archive (WAR) file

Consider the case when you have a large and complex application. It consists of dozens of resource files such as HTML pages, graphics, servlets, Java libraries, and JSPs. If you were to deploy them to several Tomcat servers, you may very well make a mistake in uploading the files. Luckily, Java Web applications can be packaged into a single compact file called a *Web Application Archive file*.

Web applications can be packaged and signed into a WAR format file using the standard Java Archive tools. For example, in the case of our small application above, we package them with the command:

```
% jar cvf catalog.jar /catalog/*
```

This creates a catalog.jar file, that is placed in /var/tomcat4/webapps and accessed via <http://host:8080/catalog>.

Note: When packaged into such a form, a META-INF directory is sometimes included. META-INF contains information useful to Java Archive tools. This directory must not be directly served as content by the container in response to a Web client's request.

7.2.4 A Quick example: Jetspeed

To get a feel of Java applications packaged in this form, let's look at a complete Web application, Jetspeed. Jetspeed is a Java-based portal server. We look at its configuration and inner workings in 7.3.1, "Jetspeed" on page 171. However, it's instructive to look at how it's accessed in its native format.

Installing Jetspeed

To install Jetspeed, follow these steps:

1. Download a copy of the latest Jetspeed WAR file from:

<http://jakarta.apache.org/builds/jakarta-jetspeed/release/v1.4b4/>

2. Unpack the jetspeed-1.4-b4.zip file:

```
% tar xzvf jetspeed-1.43b.tar.gz
```

This should yield the following files:

```
jetspeed-1.4-b4/  
jetspeed-1.4-b4/INSTALL  
jetspeed-1.4-b4/jetspeed.war  
jetspeed-1.4-b4/LICENSE  
jetspeed-1.4-b4/NOTES  
jetspeed-1.4-b4/README  
jetspeed-1.4-b4/WARNING
```

3. Copy the jetspeed.war file to your Tomcat Web applications directory:

```
# cp jetspeed.war /var/tomcat4/webapps
```

4. Restart the Tomcat Web server:

```
# service tomcat4 restart
```

The Jetspeed portal application should now be available. Point your Web browser to:

<http://hostname:8080/jetspeed>

Inside the jetspeed.war file

If you're working from your development directory, you can unpack the contents by typing the command:

```
# jar xvf jetspeed.war
```

However, if you're working from the Tomcat server, you may notice that Tomcat also automatically unpacks jetspeed.war in /var/tomcat4/webapps/ when you first invoke it. You can examine the structure of the Jetspeed Web application by looking into this directory.

Notice the directory structure of this Web application for now. Compare it with the structure defined for Java Web applications in general.

Additional resources

For more information, read "Java Web Applications" by James Goodwill. You can find this document on the Web at:

<http://www.onjava.com/pub/a/onjava/2001/03/15/tomcat.html>

7.2.5 The deployment descriptor: web.xml

At the heart of all Web applications is a deployment descriptor. It is an Extensible Markup Language (XML) file that describes configuration information for the entire Web application. It is usually named *web.xml* and is located in the WEB-INF/ directory.

For our application the location of the web.xml file is in the /catalog/WEB-INF/ directory. The information that is contained in the deployment descriptor includes:

- ▶ ServletContext Init parameters
- ▶ Localized content
- ▶ Session configuration
- ▶ Servlet/JSP definitions
- ▶ Servlet/JSP mappings
- ▶ Mime Type mappings
- ▶ Welcome file list
- ▶ Error pages
- ▶ Security

The deployment descriptor has undergone changes with each new version of the Java Servlet Specification. The version of Tomcat we are using, 4.1.24, follows the specifications of Java Servlet Specification 2.3.

As the names suggests, web.xml is an XML file and is, therefore, governed by a Document Type Descriptor (DTD). You can find this DTD on the Web at:

http://java.sun.com/dtd/web-app_2_3.dtd

Refer to the Java Servlet Specification 2.3 for a complete description of all the elements in that DTD.

An example

The following code is a short simple example of a Web application deployment descriptor:

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Appli-
cation 2.3//EN" "http://java.sun.com/j2ee/dtds/web-app_2_3.dtd">
<web-app>
  <display-name>A Simple Application</display-name>
  <context-param>
    <param-name>Webmaster</param-name>
    <param-value>webmaster@mycorp.com</param-value>
  </context-param>
  <servlet>
    <servlet-name>catalog</servlet-name>
    <servlet-class>com.mycorp.CatalogServlet</servlet-class>
    <init-param>
      <param-name>catalog</param-name>
      <param-value>Spring</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>catalog</servlet-name>
    <url-pattern>/catalog/*</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>30</session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
```

```

        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
    </welcome-file-list>
    <error-page>
        <error-code>404</error-code>
        <location>/404.html</location>
    </error-page>
</web-app>

```

The example dissected

The `<web-app></web-app>` element is the root of the deployment descriptor for the entire Web application. It contains all the other elements that describe the application.

The `<display-name></display-name>` element contains a short name that is intended to be displayed by tools.

The `<context-param></context-param>` element contains the declaration of a Web application's servlet context initialization parameters. The description elements here include any information that the WAR file producer wants to provide to the one deploying the WAR file. In our example, the names Webmaster and the e-mail address webmaster@mycorp.com are being provided.

The `<servlet></servlet>` element contains the declarative data of a servlet.

The `<servlet-name></servlet-name>` element contains the canonical name of a servlet, which is unique within the Web application. In this case, the servlet we are describing is catalog.

The `<servlet-class></servlet-class>` element contains the fully qualified class name of the servlet.

The `<init-param></init-param>` element contains a name/value pair as the initialization param of the servlet. In our example, we want catalog=Spring.

The `<servlet-mapping></servlet-mapping>` element defines a mapping between a servlet and a URL pattern. In this example, any calls to host://hostname/catalog/* are redirected to the servlet catalog.

The `<session-config></session-config>` element defines the session parameters for this Web application. In our example, we want the session timeout to be set to 30 minutes, as indicated in the `<session-timeout></session-timeout>` element.

The `<welcome-file-list></welcome-file-list>` element contains a list of all the filenames that may be used as the initial page for the application, given in order. The files are marked with the `<welcome-file></welcome-file>` elements. For example, if a user goes to `http://localhost/catalog`, the Web application looks first for `index.jsp` and displays that. If `index.jsp` is not found, then it looks for `index.html`, and lastly, for `index.htm`.

The `<error-page></error-page>` element maps any HTTP errors to a customized error page. In our example, if a requested page is not found, a case of the well-known 404 error, the application displays `404.html` instead of the default error page.

Additional resources

Supplement your knowledge with the following articles from the O'Reilly Network:

- ▶ "Java Web Applications"
<http://www.onjava.com/pub/a/onjava/2001/03/15/tomcat.html>
- ▶ "Deploying Web Applications to Tomcat"
<http://www.onjava.com/pub/a/onjava/2001/04/19/tomcat.html>
- ▶ "Creating a Web Application with Ant and Tomcat4"
<http://www.onjava.com/pub/a/onjava/2003/01/08/tomcat4.html>

Ant is "make" tool for Java applications. We touch on it briefly in 7.3.1, "Jetspeed" on page 171. For a more comprehensive treatment, read the chapter "Building the Web Application with Ant" in the redbook *Linux Application Development Using WebSphere Studio 5*, SG24-6431.

7.2.6 Understanding Tomcat's configuration file

In working with Tomcat in the preceding chapters, we had to make some slight modifications to the `server.xml` file, which is Tomcat's primary configuration file. So far, we have been working only at defining the `ServerContext` to point Tomcat to the right directory whenever we reference an application.

To enjoy the full benefit of Tomcat, we'll have to delve deeper into the options that are available to us in the `server.xml` file. We use this information later to configure Tomcat for SSL, integrating Tomcat with Apache, and finally clustering with Tomcat.

A simple `server.xml` file

Let's look at a sample `server.xml` file, using the one that came with our default Tomcat installation. It may look complicated. However, we explain the example in detail in the following sections.

```
<Server port="8005" shutdown="SHUTDOWN" debug="0">
  <Service name="Tomcat-Standalone">
    <Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
      port="8080" minProcessors="5" maxProcessors="75"
      enableLookups="true" redirectPort="8443"
      acceptCount="10" debug="0" connectionTimeout="60000"/>
    <Engine name="Standalone" defaultHost="localhost" debug="0">
      <Logger className="org.apache.catalina.logger.FileLogger"
        prefix="catalina_log." suffix=".txt"
        timestamp="true"/>
      <Realm className="org.apache.catalina.realm.MemoryRealm" />
      <Host name="localhost" debug="0" appBase="webapps" unpackWARs="true">
        <Valve className="org.apache.catalina.valves.AccessLogValve"
          directory="logs" prefix="localhost_access_log." suffix=".txt"
          pattern="common"/>
        <Logger className="org.apache.catalina.logger.FileLogger"
          directory="logs" prefix="localhost_log." suffix=".txt"
          timestamp="true"/>
        <Context path="/examples" docBase="examples" debug="0"
          reloadable="true">
          <Logger className="org.apache.catalina.logger.FileLogger"
            prefix="localhost_examples_log." suffix=".txt"
            timestamp="true"/>
        </Context>
      </Host>
    </Engine>
  </Service>
</Server>
```

```

    </Engine>
  </Service>
</Server>

```

Anatomy of a server.xml file

Graphically, Figure 7-1 shows how the server.xml file looks. The server.xml file is essentially a description of all the services that a Tomcat server will provide: the resources it uses, the Web applications it contains, how these applications will be invoked, and so on. You can then think of a server.xml file as a attributes within containers, very much like a multi-layered Christmas present.

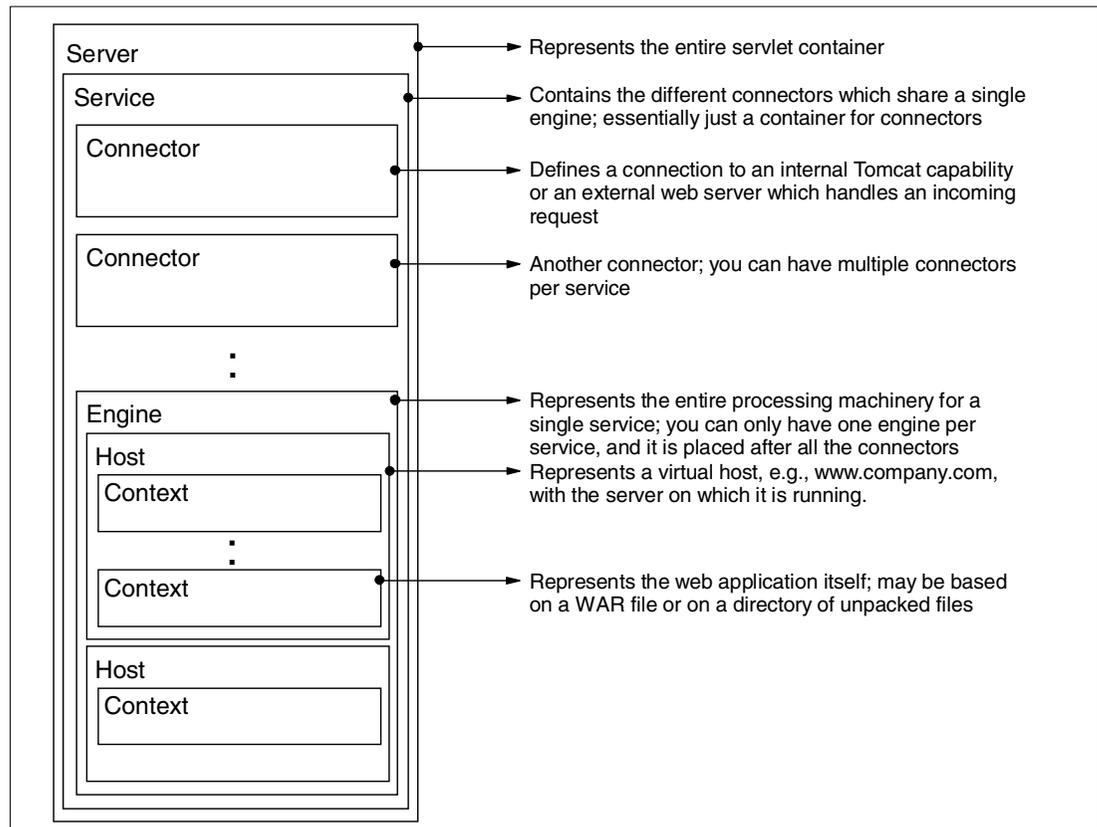


Figure 7-1 Tomcat server.xml anatomy

Now let's take our simple example apart, piece by piece, to see what it's made of.

The <Server></Server> element

A <Server></Server> element represents the entire Catalina servlet container. Therefore, it must be the single outermost element in the conf/server.xml configuration file. Its attributes represent the characteristics of the servlet container as a whole.

There are three attributes under <Server></Server>:

- ▶ **className:** This is Java class name of the implementation to use and it is almost always org.apache.catalina.Server interface. If no class name is specified, this standard implementation is used. You can safely omit this attribute.
- ▶ **port:** This is the TCP/IP port number on which this server waits for a shutdown command. This connection must be initiated from the same server computer that is running this instance of Tomcat. This attribute must be present. In our example, Tomcat waits for a shutdown command from port 8005.

- ▶ **shutdown:** This command string that must be received via a TCP/IP connection to the specified port number, in order to shut down Tomcat. This attribute must be present. In our example, the command is "SHUTDOWN".

The debug attribute is an additional attribute of the `org.apache.catalina.core.StandardServer`. It indicates the level of debugging detail logged by this server to the associated logger, which we discuss in “The `<Logger></Logger>` element” on page 158.

The debug attribute is available to all Tomcat elements. It states the debug level to use when logging messages to a defined Logger as shown in the following example. Higher numbers generate more detailed output. If not specified, the default debugging detail level is zero.

In this example, this is:

```
<Server port="8005" shutdown="SHUTDOWN" debug="0">
```

A `<Server></Server>` can contain several `<Service></Service>` elements.

The `<Service></Service>` element

The `<Service></Service>` is just a container for all the connectors that do the actual work of processing incoming requests. A `<Service></Service>` may contain several `<Connector>` elements but only one `<Engine>` element. `<Connector>` and `<Engine>` elements are described in the following sections.

The `<Service></Service>` element has two attributes:

- ▶ **className:** Java class name of the implementation to use. This class must implement the `org.apache.catalina.Service` interface. If no class name is specified, the standard implementation is used.
- ▶ **name:** The display name of this service, which is included in log messages if you use standard Catalina components. The name of each service that is associated with a particular server must be unique.

Of course, there is also the ubiquitous debug attribute.

In our example, this is:

```
<Service name="Tomcat-Standalone">
```

The `<Connector></Connector>` element

The Tomcat server essentially takes in incoming HTTP requests and processes them. Instead of handling these HTTP requests internally, Tomcat takes the modular approach and passes them on to connectors. The connector is the component that actually manages requests and responses to and from a calling client.

There are two types of connectors: connectors that allow browsers to connect directly to the Tomcat and connectors that do it through a Web server. The Tomcat documentation gives the complete list, but you only need two connectors at most.

In our example, we use the currently recommended Coyote HTTP/1.1 connector. The Coyote HTTP/1.1 enables Tomcat to function as a stand-alone Web server, in addition to its ability to execute servlets and JSP pages. A particular instance of this component listens for connections on a specific TCP port number on the server.

```
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
  port="8080" minProcessors="5" maxProcessors="75"
  enableLookups="true" redirectPort="8443"
  acceptCount="10" debug="0" connectionTimeout="60000"/>
```

Let's look at the attributes of this connector:

- ▶ **port:** This is the TCP port number on which this connector creates a server socket and awaits incoming connections. Your operating system allows only one server application to listen to a particular port number on a particular IP address. In our example, we listen at port 8080.
- ▶ **minProcessors:** This is the number of request processing threads that are created when this connector is first started. This attribute should be set to a value smaller than that set for maxProcessors. The default value is 5, which is what we are explicitly setting in our example.
- ▶ **maxProcessors:** This is the maximum number of request processing threads to be created by this connector, which determines the maximum number of simultaneous requests that can be handled. If not specified, this attribute is set to 20. In our example, we set this to 75.
- ▶ **enableLookups:** If this is set to *true*, it performs DNS lookups to return the actual host name of the remote client. If this is set to *false*, it skips the DNS lookup and returns the IP address in String form instead. By default, DNS lookups are disabled. In our case, we have set it to perform the lookup. Keep in mind that this degrades performance of the server.
- ▶ **redirectPort:** If the connector receives an SSL request, Tomcat automatically redirects the request to the port number specified here, which is port 8443 in this case.
- ▶ **acceptCount:** The maximum queue length for incoming connection requests when all possible request processing threads are in use. Any requests received when the queue is full are refused. The default value is 10, which we explicitly set.
- ▶ **connectionTimeout:** The number of milliseconds this connector waits, after accepting a connection, for the request URI line to be presented. The default value is 60000 (for example 60 seconds).

At server startup time, this connector creates several request processing threads based on the value of minProcessors. Each incoming request requires a thread for the duration of that request. If more simultaneous requests are received than can be handled by the currently available request processing threads, additional threads are created up to the configured maximum, based on the value of maxProcessors. If still more simultaneous requests are received, they are stacked up inside the server socket created by the connector, up to the configured maximum (the value of the acceptCount attribute).

Any further simultaneous requests receive "connection refused" errors, until resources are available to process them.

One or more such connectors can be configured as part of a single service. Each forwards to the associated engine to perform request processing and create the response.

For the complete list of options associated with the Coyote HTTP/1.1 connector, see the following Web site:

<http://jakarta.apache.org/tomcat/tomcat-4.1-doc/config/coyote.html>

The <Engine></Engine> element

The <Engine></Engine> element represents the entire request processing machinery associated with a particular Tomcat service. It receives and processes all requests from one or more connectors. It also returns the completed response to the connector for ultimate transmission back to the client.

Exactly one `<Engine></Engine>` element *must* be nested inside a service element, following all of the corresponding connector elements associated with this service.

The attributes of the `<Engine></Engine>` element are:

- ▶ **className**: This is the Java class name of the implementation to use. This class must implement the `org.apache.catalina.Engine` interface. If it is not specified, the standard value is used.
- ▶ **defaultHost**: This is the default host name, which identifies the host that will process requests directed to host names on this server, but which are not configured in this configuration file. This name *must* match the name attributes of one of the host elements defined within the engine.
- ▶ **jvmRoute**: This optional identifier is used for clustering and load balancing only.
- ▶ **name**: This is the logical name of this engine. It is used in log and error messages.

In our example, this is:

```
<Engine name="Standalone" defaultHost="localhost" debug="0">
```

Note: The `defaultHost` value must refer to a `<Host></Host>` element.

The `<Host></Host>` element

The `<Host></Host>` elements define host names to which the service responds. The names must also be registered with your domain name system so that they map to the IP address of your server.

You need to define at least one host, which is the default, as defined by the `<Engine></Engine>` element. One of the hosts associated with each engine *must* have a name matching the `defaultHost` attribute of that engine.

The standard attributes of the `<Host></Host>` element are:

- ▶ **appBase**: This is the pathname of a directory that may contain Web applications to be deployed on this virtual host. You may specify an absolute pathname for this directory, or a pathname that is relative to the `$CATALINA_HOME` directory.
- ▶ **autoDeploy**: This flag value indicates whether Web applications from this host should be automatically deployed by the host configurator. The flag's value defaults to true.
- ▶ **className**: This is the Java class name of the implementation to use. This class must implement the `org.apache.catalina.Host` interface. If not specified, the standard value, `org.apache.catalina.core.StandardHost` will be used. The class `org.apache.catalina.core.StandardHost` has its own set optional of attributes
- ▶ **name**: This is the network name of this virtual host, as registered in your domain name system. One of the hosts nested within an engine *must* have a name that matches the `defaultHost` setting for that engine.

To return to our example, the `<Host></Host>` element defined for our service is:

```
<Host name="localhost" debug="0" appBase="webapps" unpackWARs="true">
```

The name attribute is "localhost", which we set as the `defaultHost` in our `<Engine></Engine>` element. The application base is `webapps`. If this `server.xml` file is deployed on the server we set up, this means that applications can be deployed in `/var/tomcat4/webapps`, the `$CATALINA_HOME` directory being `/var/tomcat4`. The other attributes are maintained as the default.

What about the `unpackWARs` attribute that is also in our `<Host></Host>` element? Where did that come from?

Remember that we said that the default host class, `org.apache.catalina.core.StandardHost`, had its own set of optional attributes? `unpackWARs` is one such attribute. Set to `true`, it means that Web applications that are placed in the `appBase` directory as WAR files are automatically unpacked into a corresponding disk directory structure.

The available attributes for `org.apache.catalina.core.StandardHost` are listed in Tomcat documentation.

Virtual hosting with Tomcat

What happens if you want to map the names "www.mycompany.com" and "company.com" to the same service? The tedious and error-prone way of doing that is to create `<Host></Host>` element entries for each name, both bearing the same attributes and elements.

A quicker way of doing it is through the `<Alias></Alias>` element, a sub-element of `<Host></Host>`. We use `<Alias></Alias>` if we want both network names to map to the same virtual host, running the same applications.

A sample snippet in our `server.xml` file is shown here:

```
<Host name="www.mycompany.com" ...>
  ...
  <Alias>mycompany.com</Alias>
  ...
</Host>
```

The <Context></Context> element

The `<Context></Context>` element represents the Web application that is run within a particular virtual host. Remember that each Web application is based on a WAR file or a corresponding directory containing the corresponding unpacked contents.

The Web application used to process each HTTP request is selected by Tomcat based on matching the longest possible prefix of the Request URI against the context path of each defined context. After it is selected, that context selects an appropriate servlet to process the incoming request, according to the servlet mappings defined in the Web application deployment descriptor file, the `web.xml` which we are now familiar with.

You may define as many `<Context></Context>` elements as you choose. All of them are nested within a `<Host></Host>` element in `server.xml`. Each such context *must* have a unique context path, which is defined by the `path` attribute. In addition, you *must* define a context with a context path equal to a zero-length string. This context becomes the default Web application for this virtual host. It is used to process all requests that do not match any other context's context path.

Let's look at what we set in our example:

```
<Context path="/examples" docBase="examples" debug="0"
  reloadable="true">
```

There are several attributes related to context. In our particular example, we have set:

- ▶ **Path:** The context path of this Web application, which is matched against the beginning of each request URI to select the appropriate Web application for processing. All of the context paths within a particular host must be unique.
- ▶ **Docbase:** This is the document base (also known as the *context root*) directory for this Web application, or the pathname to the WAR file (if this Web application is being

executed directly from the WAR file). You may specify an absolute pathname for this directory or WAR file, or a pathname that is relative to the appBase directory of the owning host.

- ▶ **Reloadable:** Set to true if you want Catalina to monitor classes in /WEB-INF/classes/ and /WEB-INF/lib for changes, and automatically reload the Web application if a change is detected. This feature is useful during application development. However, it requires significant runtime overhead and is not recommended for use on deployed production applications.

Given our example, any reference to http://hostname:8080/examples is then mapped to elements in /var/tomcat4/webapps/examples. Since we set the attribute Reloadable to be *true*, all the class files are automatically read in by Tomcat if they are changed. This entails a performance hit.

The <Logger></Logger> element

In our example, one of the nested elements in the <Context></Context> element is <Logger></Logger>:

```
<Context path="/examples" docBase="examples" debug="0"
  reloadable="true">
  <Logger className="org.apache.catalina.logger.FileLogger"
    prefix="localhost_examples_log." suffix=".txt"
    timestamp="true"/>
</Context>
```

Look at the <Host></Host> container. It also has an associated <Logger></Logger> element.

```
<Logger className="org.apache.catalina.logger.FileLogger"
  directory="logs" prefix="localhost_log." suffix=".txt"
  timestamp="true"/>
```

The <Logger></Logger> element represents a destination for logging, debugging, and error messages for a Tomcat container. As such, it applies to <Context></Context> and <Host></Host> and can be used for <Engine></Engine>.

Loggers associated with an <Engine></Engine> or a <Host></Host> are automatically inherited by lower-level containers, unless explicitly overridden.

Let's look at some of the attributes we use for our <Logger></Logger> element:

- ▶ **ClassName:** There are several possible logger class implementations to use. In this example, we use org.apache.catalina.logger.FileLogger, which records all logged messages to disk files in a specified directory. Log files are created from a configured prefix, the current date in YYYY-MM-DD format, and a configured suffix. Other options for className are: the Standard Error Logger (org.apache.catalina.logger.SystemErrLogger) and the Standard Output Logger (org.apache.catalina.logger.SystemOutLogger).
- ▶ **Directory:** This is the absolute or relative pathname of a directory in which log files created by this logger is placed. We defined it as *logs* in the <Host></Host> container, so the logger entry in the <Context></Context> also inherits this location. For our setup, these logs are placed in /var/tomcat4/logs.
- ▶ **Prefix:** This is the prefix of the log file, which is "localhost_examples_log" as well as "localhost_log" in this case.
- ▶ **Suffix:** This is the suffix of the log file, which is ".txt" in this case.
- ▶ **Timestamp:** Setting timestamp to *true* causes all logged messages to be date- and time-stamped. Sample log files generated by these settings are "localhost_examples_log.2003-05-03.txt" and "localhost_log.2003-05-03.txt".

The <Valve></Valve> element

A <Valve></Valve> element represents a component that is inserted into the request processing pipeline for the associated Tomcat container, whether it be engine, host, or context.

There are several valve implementations, each one with a different function:

- ▶ **Access Log Valve:** This creates log files in the same format as those created by standard Web servers. These logs can later be analyzed by standard log analysis tools to track page hit counts, user session activity, and so on. This is the valve that we are using in our example. As you can see, it shares characteristics with the file logger implementations:

```
<Valve className="org.apache.catalina.valves.AccessLogValve"
  directory="logs" prefix="localhost_access_log." Suffix=".txt"
  Pattern="common"/>
```

- ▶ **Remote Address Filter valve:** This valve allows you to compare the IP address of the client that submitted a request against a pattern list. It either allows the request to continue or refuses to process the request from this client.
- ▶ **Remote Host Filter:** This filter allows you to compare the host name of the client that submitted this request against a pattern list. It either allows the request to continue or refuses to process the request from this client.
- ▶ **Request Dumper Valve:** This valve is a useful tool in debugging interactions with a client application or browser that is sending HTTP requests to your Tomcat-based server.
- ▶ **Single Sign On Valve:** This valve is used when you want to give users the ability to sign on to any one of the Web applications associated with your virtual host, and then have their identity recognized by all other Web applications on the same virtual host.

Additional resources

We've gone through a detailed exposition of a short and fairly simple server.xml file of Tomcat. Keep in mind that there are several other possible option that can be set. We've taken you through some of the more important ones and provided you a glimpse of the other possibilities.

The *Tomcat Server Configuration Reference* provides a detailed explanation of the options and rules. As you configure your Web application server, you should frequently reference this document. You can find this document online at:

<http://jakarta.apache.org/tomcat/tomcat-4.1-doc/index.html>

See also James Goodwill's document "Demystifying Tomcat's server.xml file" on the Web at:

<http://www.onjava.com/pub/a/onjava/2002/07/31/tomcat.html>

7.2.7 Using the Tomcat Web Application Manager

Tomcat has an application manager that greatly simplifies the task of installing and managing a Web application. The application manager allows you to deploy a new Web application or undeploy an existing one without having to restart the entire Tomcat container.

The Web application manager allows you to:

- ▶ Deploy a new Web application, on a specified context path, from the uploaded contents of a WAR file.
- ▶ Install a new Web application, which can be anywhere on the server's disks.

- ▶ List the currently deployed Web applications and the sessions that are currently active for those Web applications.
- ▶ Reload an existing Web application to reflect changes in the contents of /WEB-INF/classes or /WEB-INF/lib.
- ▶ List the operating system and JVM property values.
- ▶ List the available global Java Naming and Directory Interface (JNDI) resources, for use in deployment tools that are preparing <ResourceLink> elements nested in a <Context> deployment description.
- ▶ List the available security roles defined in the user database.
- ▶ Remove an installed Web application.
- ▶ Start a stopped application (making it available again).
- ▶ Stop an existing application (so that it becomes unavailable), but do not undeploy it.
- ▶ Undeploy a deployed Web application and delete its document base directory.

The Tomcat application manager can be accessed three ways:

- ▶ Through a Web interface you use in your browser. This is the method we concentrate on in this chapter.
- ▶ As a minimal version using HTTP requests only which can work with customized scripts set up by system administrators. Commands are given as part of the request URI, and responses are in the form of simple text that can be easily parsed and processed.
- ▶ As a convenient set of task definitions for the Ant build tool.

Installing the Tomcat Web Application Manager

The Tomcat Web Application Manager can be downloaded as an RPM-installable file, tomcat4-admin-webapps-4.1.24-full.2jpp.noarch.rpm from the Jakarta Tomcat binaries directory. To install the application manager, enter the following command:

```
# rpm -ivh tomcat4-admin-webapps-4.1.24-full.jpp.noarch.rpm
```

This installs the necessary files and updates the server.xml file to permit a privileged user to manipulate the container.

Configuring the Tomcat Web Application Manager

Before you can access the Tomcat Web Application Manager, you must create a user account with administrative rights over the Tomcat server. For security reasons, the application manager does not have such an administrative account enabled by default.

1. To create the account, edit /etc/tomcat4/tomcat-users.xml. The default contents of the file are:

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="role1"/>
  <role rolename="tomcat"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="role1" password="tomcat" roles="role1"/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
</tomcat-users>
```

2. Log in as root and change this to:

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="manager"/>
```

```

<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="admin" password="password" roles="manager"/>
<user username="tomcat" password="tomcat" roles="tomcat"/>
<user username="role1" password="tomcat" roles="role1"/>
<user username="both" password="tomcat" roles="tomcat,role1"/>
</tomcat-users>

```

Don't forget to select a better user name and password for your administrator.

3. Restart the Tomcat server:

```
# service tomcat4 restart
```

4. Point your Web browser to:

```
http://hostname:8080/manager/html
```

You are prompted for your user name and password. If you followed our sample tomcat-users.xml file, the user name is "admin" and the password is "password". After giving the right credentials, you should see the page shown in Figure 7-2.

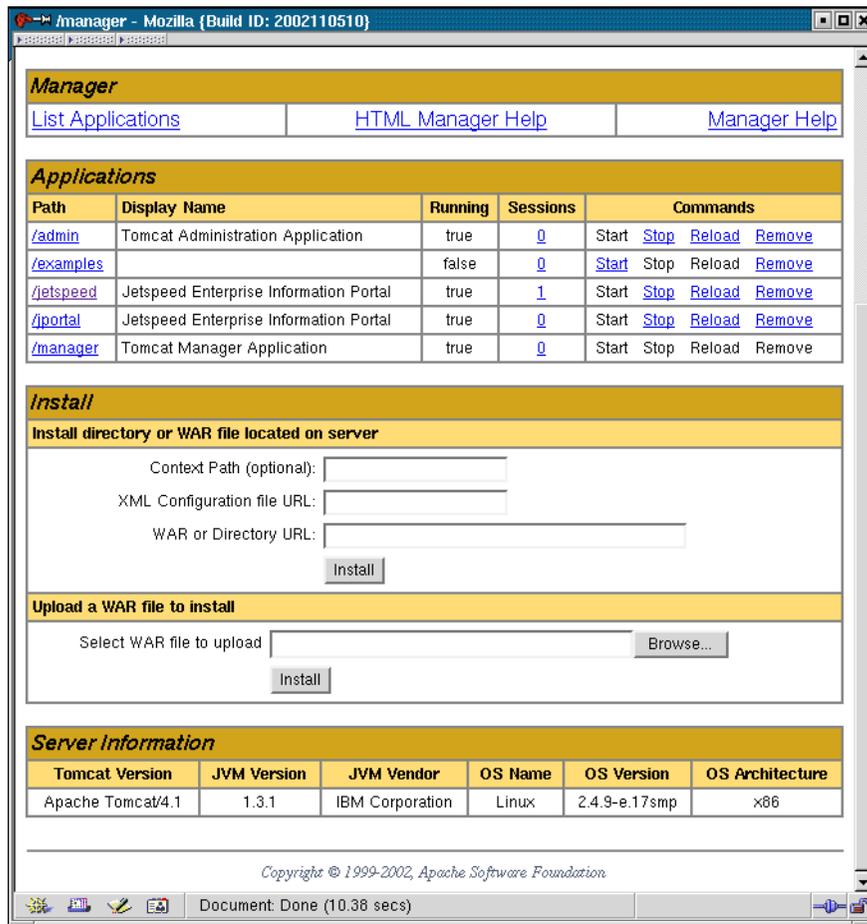


Figure 7-2 Tomcat Web Administration Manager

Using the Tomcat Web Application Manager

The main interface is divided into five sections:

- ▶ **Message:** Displays success and failure messages
- ▶ **Manager:** General manager operations such as list and help
- ▶ **Applications:** List of Web applications and commands

- ▶ **Install:** Installing Web applications
- ▶ **Server Information:** Information about the Tomcat server

Message

The message section displays information about the success or failure of the last Web application manager command you performed. If it succeeded, you see OK which may be followed by a success message. If it failed, you see FAIL followed by an error message.

Manager

The Manager section has three links:

- ▶ **List Applications:** Redisplay a list of Web applications
- ▶ **HTML Manager Help:** A link to the Web application manager help pages
- ▶ **Manager Help:** A link to the comprehensive manager application HOW-TO

Applications

The Applications section lists all the Web applications installed on the Tomcat server along with status information. It also provides links for managing them. For each Web application, the following items are displayed:

- ▶ **Path:** The Web application context path, specifically, how the Web application is accessed via a browser.
- ▶ **Display name:** The display name for the Web application if it has one configured in its "web.xml" file.
- ▶ **Running:** Whether the Web application is running and available (true) or not running and unavailable (false).
- ▶ **Sessions:** The number of active sessions for remote users of this Web application. The number of sessions is a link that, when submitted, displays more details about session usage by the Web application in the Message box.
- ▶ **Commands:** Lists all commands, which can be performed on the Web application. Only those commands that can be performed are listed as a link that can be submitted. No commands can be performed on the manager Web application itself. The following commands can be performed:
 - **Start:** Start a Web application that was stopped.
 - **Stop:** Stop a Web application that is currently running and make it unavailable.
 - **Reload:** Reload the Web application so that new JAR files in /WEB-INF/lib/ or new classes in /WEB-INF/classes/ can be used.
 - **Remove:** Stop and then remove this Web application from the server.

Install

Web applications can be installed using files or directories located on the Tomcat server or you can upload a WAR file to the server. To install an application, fill in the appropriate fields for the type of install you want to do. Then click the **Install** button to submit it.

The different methods for installing an application are explained on the following page.

Server Information

This section of the Web application manager displays information about Tomcat, the operating system of the server Tomcat is hosted on, and the JVM in which Tomcat is running.

Installing applications

There are several ways to install applications using the Tomcat Web Application Manager.

Installing a Directory or WAR by URL

If a Web application already resides on the server, you can use this method. In the WAR or Directory URL field, enter the name of the application. The field takes in the URL naming format. For example, if your application is located in the directory `/home/user/webapp1`, in the WAR or Directory URL field, you enter:

```
file:/home/user/webapp1
```

As another example, if you had the application as a WAR file `webapp2.war` in the directory `/home/user`, you enter:

```
jar:file:/home/user/webapp2.war
```

If the Context Path field is left empty, the context path defaults to the name of the directory or the WAR file. In our examples, the default context paths are automatically the `/webapp1` and `/webapp2` directory.

If we want Context Path to be something else, we enter that name into the Context Path field. For example, if we want the application in the directory `/home/user/webapp1` to map to `/mywebapp`, you have the following values:

- ▶ Context Path: `/mywebapp`
- ▶ WAR or Directory URL: `file:/home/user/webapp1`

Installing a directory or URL in the host appBase

The host appBase is specified in Tomcat's `server.xml` file. The host appBase that we use in our examples is `/var/tomcat4/webapps`. You can install a Web application directory or WAR file directory in that directory and use the Tomcat Web Application Manager to recognize it. As before, if no Context Path is specified, the directory name or the WAR file name is used as the path.

For example, if we place our Web application in a subdirectory named `webapp1` in `/var/tomcat4/webapps`, the values for WAR or Directory URL is `webapp1`. No other path information is necessary.

If we had a WAR file named `webapp2.war`, the value for WAR or Directory URL is `webapp2.war`.

Installing using a Context configuration XML file

This is a more advanced installation method that requires you to understand the workings of Tomcat's configuration file `server.xml`. In effect, you create a description of the application in a *context configuration XML file*. The file itself is a snippet that fits within the `server.xml` configuration file.

Here is an example of a context configuration XML file:

```
<Context path="/webapp1" docBase="/home/user/webapp1"
        debug="0">
<!-- Link to the user database we will get roles from -->
<ResourceLink name="users" global="UserDatabase"
        type="org.apache.catalina.UserDatabase"/>
</Context>
```

If you use this method, you do not place any entry in the Context-Path field. Use of the WAR or Directory URL is optional. All information required to describe the file can already be found in the context configuration XML file.

Uploading a WAR file to install

Use this method if your deployment server is different from your development system. In this method, you upload the WAR file from your local system. It is installed automatically in the appBase of the host. The Context Path is simply the name of the WAR file.

The file uploaded must be a WAR file. The upload fails if it is not a WAR file.

7.2.8 SSL with Tomcat

This section takes you through the steps of using Tomcat to create an SSL session. Note that this step is only necessary when running Tomcat as a stand-alone Web server.

Enabling Tomcat with SSL

Enabling Tomcat for SSL involves three basic steps.

Step 1: Download the Java Secure Socket Extensions (JSSE)

Download the Java Secure Socket Extensions package, version 1.0.2 or later, from:

<http://java.sun.com/products/jsse>

The JSSE is a zip file that contains three JAR files. Make JSSE an installed extension by copying all three JAR files (jcert.jar, jnet.jar, and jsse.jar) into your /opt/IBMJava2-131/jre/lib/ext directory.

If you are running JDK 1.4, these classes are integrated directly into the JDK, so you can skip this entire step.

Step 2: Prepare the certificate Keystore

Tomcat currently operates only on Java's standard "Java KeyStore" format. This format is created by the keytool command-line utility. This tool is included in the JDK.

To create a new keystore from scratch, containing a single self-signed certificate, type the following command from a terminal command line:

```
# $JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA -keystore /etc/keystore  
-storepass changeit
```

This command creates a new file called /etc/keystore. The password to access this file is "changeit".

You are prompted for general information about this certificate, such as company, contact name, and so on. This information is displayed to users who attempt to access a secure page in your application, so make sure that the information provided here matches what they expect.

Finally, you are prompted for the key password, which is the password specifically for this certificate (as opposed to any other certificates stored in the same keystore file). You *must* use the same password here as was used for the keystore password itself. Currently, the keytool prompt tells you that pressing the Enter key does this for you automatically.

If everything is successful, you now have a keystore file with a certificate that can be used by your server.

Step 3: Edit the Tomcat configuration file

The final step is to configure your secure socket in the server.xml file. An example `<Connector>` element for an SSL connector is included in the default server.xml file installed with Tomcat. It looks something like this example:

```
<!-- Define an SSL HTTP/1.1 Connector on port 8443 -->
<!--
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
           port="8443" minProcessors="5" maxProcessors="75"
           enableLookups="true"
           acceptCount="100" debug="0" scheme="https" secure="true"
           useURIValidationHack="false" disableUploadTimeout="true">
  <Factory className="org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
          clientAuth="false" protocol="TLS"
          keystoreFile="/etc/keystore" keystorePass="changeit"/>
</Connector>
-->
```

This statement is commented out by default, so you need to remove the comment tags around it. You must also add the `keystoreFile` and `keystorePass` properties.

Testing the setup

Restart Tomcat and point your Web browser to it. Go to:

`http://localhost:8443/`

Your browser is prompted to accept the unrecognized certificate.

Re-examining the server.xml file

Note that is the same type of `<Connector>`/`</Connector>` that we dealt with in the previous chapter. Some new attributes are introduced here:

- ▶ **scheme:** Set this attribute to the name of the protocol that you want to have returned by calls. The default for this is *HTTP*. Since we are working with SSL, it is *HTTPS*.
- ▶ **secure:** Set this attribute to *true* if you are dealing with secure connections. The default property is *false*.

The port attribute

The port attribute (default value is 8443) is the TCP/IP port number on which Tomcat listens for secure connections. You can change this to any port number you want (such as to the default port for HTTPS communications, which is 443).

The <Factory>/Factory> element

You also notice a `<Factory>`/`</Factory>` element nested inside the `<Connector>`/`</Connector>`. Where did this come from? This is where the *socket factory* used by Tomcat, whenever it needs a socket on the corresponding port number, is configured. The `<Factory>`/`</Factory>` element is present only when setting up a connector for SSL.

For the full list of attributes for the `<Factory>`/`</Factory>` element, see the Tomcat Server Configuration Reference at:

<http://jakarta.apache.org/tomcat/tomcat-4.1-doc/index.html>

Installing certificates from a Certificate Authority

In our previous example, we generated our own certificates. In real life, the certificates are most likely provided by a Certificate Authority. The following steps explain how to install a certificate.

Step 1: Create a local Certificate Signing Request (CSR)

To obtain a Certificate from the Certificate Authority of your choice, create a Certificate Signing Request. The CSR is used by the Certificate Authority to create a certificate that identifies your Web site as *secure*. To create a CSR, follow these steps:

1. Create a local certificate:

```
# keytool -genkey -alias tomcat -keyalg RSA -keystore <your_keystore_filename>
```

Note: In some cases, you must enter the domain of your Web site, such as `www.myside.org`, in the first- and lastname field to create a working certificate.

2. Create the CSR with the following command:

```
keytool -certreq -keyalg RSA -alias tomcat -file certreq.csr -keystore  
<your_keystore_filename>
```

3. You now have a file called `certreq.csr`. The file is encoded in PEM format. You can submit it to the Certificate Authority (look at the documentation of the Certificate Authority Web site on how to do this). In return, you receive a certificate.

Step 2: Importing the certificate

Now that you have your certificate, you can import it into your local keystore. First you must import a *Chain Certificate* or *Root Certificate* into your keystore. After that you can proceed with importing your Certificate.

1. Download a Chain Certificate from the Certificate Authority you obtained the Certificate from:

- For Verisign.com, go to:

<http://www.verisign.com/support/install/intermediate.html>

- For Trustcenter.de, go to:

<http://www.trustcenter.de/certservices/cacerts/en/en.htm#server>

- For Thawte.com, go to:

<http://www.thawte.com/certs/trustmap.html>

2. Import the Chain Certificate into your keystore:

```
# keytool -import -alias root -keystore <your_keystore_filename> -trustcacerts -file  
<filename_of_the_chain_certificate>
```

3. Import your new certificate, which must be in the X.509 format:

```
# keytool -import -alias tomcat -keystore <your_keystore_filename> -trustcacerts -file  
<your_certificate_filename>
```

When to use SSL

Setting up SSL with Tomcat is only necessary when running it as a stand-alone Web server. When running Tomcat primarily as a Servlet/JSP container behind another Web server, such as Apache or Microsoft IIS, it is usually the primary Web server that handles the SSL connections from users. Typically, this server negotiates all SSL-related functionality. Then it passes on any requests destined for the Tomcat container only after decrypting those requests.

Likewise, Tomcat returns cleartext responses that are encrypted before they are returned to the user's browser. In this environment, Tomcat knows that communications between the primary Web server and the client are taking place over a secure connection, but it does not participate in the encryption or decryption itself.

Additional resources

The Galatea Flashguides presents several short tutorials on SSL for various versions of Tomcat. See the following Web sites:

<http://www.galatea.com/flashguides/tomcat-ssl-4-unix.xml>

<http://www.galatea.com/flashguides/tomcat-ssl-unix.xml>

7.2.9 Integrating Tomcat and Apache

While Tomcat can serve static documents like HTML pages and graphics, it does not perform nearly as well as Apache does for this task. Tomcat's primary purpose is to be an application server. At the same time, Apache doesn't have any Web application capabilities.

To have the best of both worlds, we can use Apache and Tomcat together. Apache processes all incoming requests and passes any requests to Tomcat via connectors.

Currently three connectors are available for Apache-Tomcat integration:

- ▶ **JK** is a replacement to the deprecated `mod_jserv`, the old connector for Apache and Tomcat. JK is a completely new Tomcat-Apache plug-in that handles the communication between Tomcat and Apache.
- ▶ **JK2** is the revised version of JK. The native part was completely restructured and the configuration is now simplified.
- ▶ **mod_webapp** is a new way to integrate Apache and Tomcat, but it is still new and should not be used for production environments.

We use `mod_jk` as our connector in our example.

Required files

For this section, you need the following packages:

- ▶ `apache-1.3.27-2`
- ▶ `tomcat4-4.1.24-full.2jpp`
- ▶ `tomcat4-webapps-4.1.24-full.2jpp` (optional)
- ▶ `IBMJava2-SDK-1.3.1-5`

You also need to download `mod_jk-ap13-1.2.2-1jpp.i386.rpm` from the Web at:

<http://jakarta.apache.org/builds/jakarta-tomcat-connectors/jk/release/v1.2.2/rpms>

Prerequisites

For now, let's install Tomcat and Apache on the same server. As a prerequisite then, install Apache, Tomcat, and the Java SDK on the server.

You should test Apache and Tomcat, but make sure to shut them down before you proceed with the reconfiguration in the following sections.

Installing mod_jk

Install the `mod_jk` connector from the RPM file. This copies the `mod_jk.so` library into the modules library of Apache. It also adds the configuration files `mod_jk.conf` and the `workers.properties` file into `/etc/httpd/conf`.

Installing the `mod_jk` connector updates the `/etc/httpd/conf/httpd.conf` configuration file of Apache.

Reviewing the httpd.conf file

After installation, the following lines are added at the end of the httpd.conf file:

```
<IfDefine HAVE_JK>
    LoadModule jk_module modules/mod_jk.so
    AddModule mod_jk.c
    Include /etc/httpd/conf/mod_jk.conf
</IfDefine>
```

You can leave this as is. However, when you start the Apache Web server, it may throw a warning indicating that mod_jk was already loaded. This is not serious, but if you don't want to see it, you can comment out the AddModule line.

mod_jk.conf

The file /etc/httpd/conf/mod_jk.conf is installed by the RPM. This is actually a list of Apache directives that are read into httpd.conf when Apache first starts. This file is important for integration because it tells Apache how to use Tomcat.

Let's examine this file a little closer.

The top three lines tell Apache where the workers.properties file and mod_jk log files are:

```
JKWorkersFile /etc/httpd/conf/workers.properties
JKLogFile     /var/log/httpd/mod_jk.log
JKLogLevel    error
```

They also tell Apache what level of incidents to log.

Next, you see the following directives:

```
JKMount /*.jsp ajp13
JKMount /servlet/* ajp13
```

This tells Apache to pass any JSP and servlet requests ordinarily in the Tomcat root context (http://hostname:8080/index.jsp and http://hostname:8080/servlet/helloworld in our previous examples with Tomcat only) to the Tomcat server. With this activated, a request to http://hostname/index.jsp or http://hostname/servlet/helloworld is redirected by Apache to Tomcat.

You also see the following lines:

```
Alias /examples "/var/tomcat3/webapps/examples"
<Directory "/var/tomcat3/webapps/examples">
    Options Indexes FollowSymLinks
</Directory>
JKMount /examples/servlet/* ajp13
JKMount /examples/*.jsp ajp13
<Location "/examples/WEB-INF/">
    AllowOverride None
    deny from all
</Location>
```

These lines reference the directory that the tomcat4-webapps RPM installs (if you chose to install them). You have to change the lines that refer to tomcat3 to tomcat4 to reflect the location of the files. These lines tell Apache how to handle any requests to http://hostname/examples:

- ▶ **Alias /examples "/var/tomcat3/webapps/examples"**: Any request made to http://hostname/examples is read from /var/tomcat4/webapps/examples. This covers everything except JSPs and files in the servlet subdirectory.

- ▶ **JkMount /examples/servlet/* ajp13:** Any request made to http://hostname/examples with a JSP extension are redirected to the Tomcat server.
- ▶ **JkMount /examples/*.jsp ajp13:** All requests made to http://hostname/examples/servlet are redirected to the Tomcat server.

The other directives are standard Apache directives designed to protect the directories. The rest of the file should be self-explanatory.

workers.properties

workers.properties controls the behavior of the actual connector between Apache and Tomcat. The default file configures for both ajp12 and ajp13. You can comment out ajp12, since we do not use it.

The contents of our workers.properties file is:

```
workers.tomcat_home=/var/tomcat4
workers.java_home=/opt/IBMJava2-131
ps=/
worker.list=ajp13
worker.ajp13.port=8009
worker.ajp13.host=localhost
worker.ajp13.type=ajp13
worker.ajp13.lbfactor=1
worker.loadbalancer.type=lb
worker.loadbalancer.balanced_workers=ajp13
worker.inprocess.type=jni
worker.inprocess.class_path=$(workers.tomcat_home)$(ps)lib$(ps)tomcat.jar
worker.inprocess.cmd_line=start
worker.inprocess.jvm_lib=$(workers.java_home)$(ps)jre$(ps)bin$(ps)classic$(ps)libjvm.so
worker.inprocess.stdout=$(workers.tomcat_home)$(ps)logs$(ps)inprocess.stdout
worker.inprocess.stderr=$(workers.tomcat_home)$(ps)logs$(ps)inpr
```

You have to edit the default file to reflect the changes we made.

server.xml

Finally, you have to edit /var/tomcat4/conf/server.xml.

Comment out JMX MBeans support:

```
<!-- Comment these entries out to disable JMX MBeans support -->
<!-- You may also configure custom components (e.g. Valves/Realms) by
      including your own mbean-descriptor file(s), and setting the
      "descriptors" attribute to point to a ';' seperated list of paths
      (in the ClassLoader sense) of files to add to the default list.
      e.g. descriptors="/com/myfirm/mypackage/mbean-descriptor.xml"
-->
<!--
<Listener className="org.apache.catalina.mbeans.ServerLifecycleListener"
          debug="0"/>
<Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener"
          debug="0"/>
-->
```

Also comment out the standard HTTP connector. We won't connect through port 8080 of this server anymore.

```
<!-- Define a non-SSL Coyote HTTP/1.1 Connector on port 8080 -->
<!--
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
```

```

        port="8080" minProcessors="5" maxProcessors="75"
        enableLookups="true" redirectPort="8443"
        acceptCount="100" debug="0" connectionTimeout="20000"
        useURIValidationHack="false" disableUploadTimeout="true" />
-->

```

Comment out the JK2 section, since we are not using it:

```

<!-- Define a Coyote/JK2 AJP 1.3 Connector on port 8009 -->
<!--
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
        port="8009" minProcessors="5" maxProcessors="75"
        enableLookups="true" redirectPort="8443"
        acceptCount="10" debug="0" connectionTimeout="0"
        useURIValidationHack="false"
        protocolHandlerClassName="org.apache.jk.server.JkCoyoteHandler"/>
-->

```

Finally, uncomment the JK section, which we are using:

```

<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector className="org.apache ajp.tomcat4.Ajp13Connector"
        port="8009" minProcessors="5" maxProcessors="75"
        acceptCount="10" debug="0"/>

```

Starting the servers

Start first the Tomcat server, followed by the Apache server. Then, launch a Web browser and attempt to access:

```
http://hostname/examples/index.jsp
```

If you see an error that says that Apache is unable to communicate with Tomcat, check the port address in your workers.properties. Compare it with the one defined in server.xml. They should be the same. If not, just change the values so both agree.

Additional resources

This section was based on the work by Pascal Chong and available on the Web at:

http://linux-sxs.org/internet_serving/tomcat-apache.html

You may also want to try load-balancing and clustering with Tomcat and using mod_jk2.

For more information, check out the following Web sites:

- ▶ "Configuring Apache and Tomcat 4 with JK 1.2" by James Goodwill
<http://www.onjava.com/pub/a/onjava/2002/11/20/tomcat.html>
- ▶ "Integrating Apache and Tomcat4 for UNIX" from the Galatea Flashguides
<http://www.galatea.com/flashguides/apache-tomcat-4-unix.xml>
- ▶ "Enabling SSL in Tomcat/Apache in UNIX" from Galatea Flashguides
<http://www.galatea.com/flashguides/apache-tomcat-ssl-unix.xml>
- ▶ "Tomcat 4 Clustering HOWTO" by Filip Hanik
<http://cvs.apache.org/~fhanik/index.html>
- ▶ "In-session Memory Replication" by Filip Hanik
<http://www.filip.net/tomcat/tomcat-javagroups.html>

- ▶ “Apache 1.3.23 + Tomcat 4.0.2 + Load Balancing” by Pascal Forget
<http://www.ubears.com/tomcat/>

7.3 Portals

When the Internet started to catch on in 1995, real value started to be created when companies put their information online. Many companies had the goal of organizing all their information, accessible from a single point. The term *portal* was adopted to describe this type of Web application. A portal is the first point of access to a Web server. Functionally, it is designed to present information in a concise and centered way, thereby making the Internet or intranet easier to use. Popular examples are Yahoo! or Netscape.

A portal makes network resources (applications, databases and so forth) available to end users. With modern portals, users can access the portal via a Web browser, WAP-phone, pager, or any other device.

7.3.1 Jetspeed

In “The <Connector></Connector> element” on page 154, we used Jetspeed as a sample Web application. We also started it running very quickly within a Tomcat environment. In this section, we look at greater detail into what it is and what features it provides.

What is Jetspeed?

Jetspeed is an open source implementation of an Enterprise Information Portal, using Java and XML. Jetspeed aggregates information from multiple sources and provides a common structure and framework for these data sources and applications.

Jetspeed helps you build portal applications quickly. It is a tool for both portal developers and user interface designers. Currently the focus is on providing developers with a set of tools that facilitates building the base for the portal. With Jetspeed, you can quickly build an XML portal and syndicate your own content.

Jetspeed features

Jetspeed has several features that make it appealing as an implementation for portals. Here is a quick summary of these features:

- ▶ Standardized Java Portlet API specification (standardization in process)
- ▶ Template-based layouts
- ▶ Supports remote XML content feeds
- ▶ Custom default home page configuration
- ▶ Database user authentication (with Lightweight Directory Access Protocol (LDAP) in beta)
- ▶ In-memory cache for quick page rendering
- ▶ Rich Site Summary (RSS) support for syndicated content
- ▶ Integration with Cocoon, WebMacro, and Velocity so that you can develop with the newest XML/XSL technology
- ▶ Wireless Markup Language (WML) support
- ▶ XML based configuration registry of portlets
- ▶ WAR support
- ▶ Synchronization with Avantgo

- ▶ Fully portable across all platforms that support JDK 1.2 and Servlet 2.2
- ▶ Profiler Service to access portal pages based on user, security (groups, roles, access control lists), media types, and language
- ▶ Persistence Service available to all portlets to easily store state per user, page, and portlet
- ▶ Skins so that users can choose colors and display attributes
- ▶ Customizer for selecting portlets and defining layouts for individual pages
- ▶ Portlet structure markup language (PSML) for defining layouts
- ▶ User, group, role and permission administration via Jetspeed security portlets
- ▶ Role-based security access to portlets

Getting started with Jetspeed

Jetspeed requires a Web application container to function. In 7.2.4, “A Quick example: Jetspeed” on page 149, we used Jetspeed as a sample Web application. Refer back to that section to learn how to install Jetspeed. Figure 7-3 shows the default portal application page for Jetspeed.

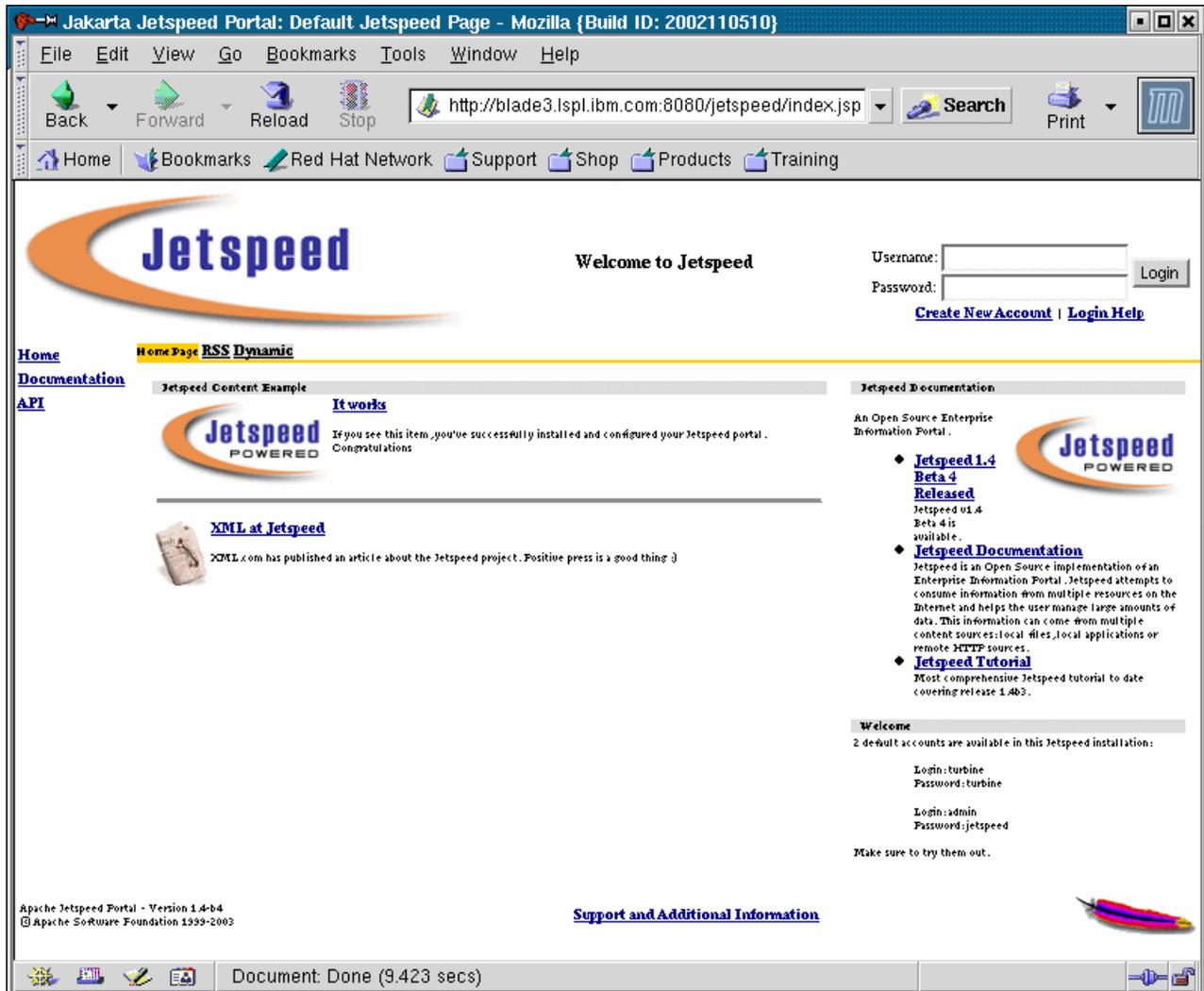


Figure 7-3 Jetspeed default portal application

Using Jetspeed: A user's perspective

The default Jetspeed installation gives you a working example portal. It's really meant to be a showcase of what Jetspeed offers. There are two user accounts with the user names and passwords conveniently listed in the lower right corner of the sample page. There are also some sample applications that you can access.

Let's take Jetspeed for a trial run, using the default settings.

To begin, log on as the user turbine (password: turbine). This brings you to the portal that was created for that user as shown in Figure 7-4.

The screenshot shows a Mozilla browser window displaying the Jakarta Jetspeed Portal. The browser title is "Jakarta Jetspeed Portal: Default Jetspeed page - Mozilla {Build ID: 2002110510}". The address bar shows "http://blade3.ispl.ibm.com:8080/jetspeed/portal;jsessionid". The page features the Jetspeed logo and a welcome message for "Tommy Turbine". A navigation menu on the left includes "Home", "Documentation", and "API". The main content area is titled "Welcome to Jetspeed" and includes a "Hello World in Velocity" portlet with a text input field and an "Update" button. Below this is a "Stock Portfolio" portlet with a table of stock data. Other portlets include "xmlhack" with developer news, "Turbine" with a Java Web Application Framework description, and "DatabaseBrowserTest" with a table of coffee data.

Symbols	Price	Change	Volume
MSFT	25.63	-0.26	45131528
IBM	88.70	-1.28	7752300
ORCL	12.53	+0.11	34512080
SUNW	4.24	0.00	47426756

COFFEE_ID	COFFEE_NAME	SUPPLIER_ID	PRICE	SALES	TOTAL
1	ColombianGrade	5	7.59	1	2

Figure 7-4 Jetspeed portal for 'turbine'

As you can see, the page is broken down into sections:

- ▶ A header, on top
- ▶ A footer, at the bottom
- ▶ A navigator, on the left (this can be placed elsewhere)
- ▶ The main portal area occupying the major part of the page

The main portal area is divided into blocks of information. Each block is a portlet, essentially, a small application that performs a task and presents the output within the framework

provided by the Jetspeed portal. The portlet can be practically anything from a database query, a web-mail client, or news feed to a stock ticker.

The portlets have a uniform structure around them. Each is bounded by a set of controls to customize, print, magnify, close, minimize, or maximize the portlet.

As a portal user, you have a great degree of flexibility to customize the look-and-feel of your site. You can change the layout of your portal, you can rearrange the placement of your portlets, and you can add and delete portlets. The changes are persistent across your session. Jetspeed remembers your configuration preferences.

For example, let's say you choose to have a different layout for your portal. Click the **pencil icon** in the yellow box near the top left corner of the page. This brings you to the layout configuration page (Figure 7-5).

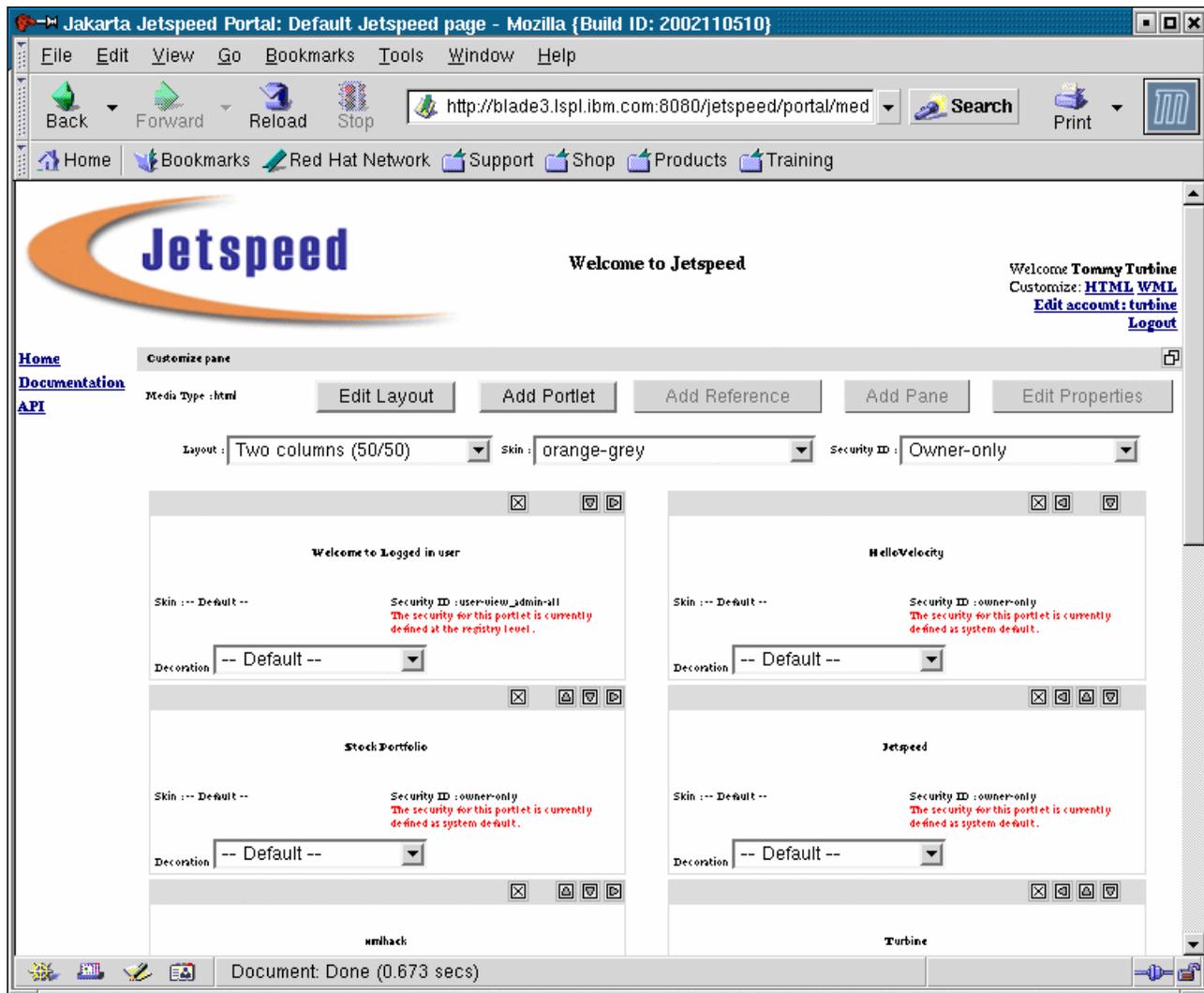


Figure 7-5 Layout configuration page

Select a new layout, for example **Two Columns (75/25)**, and a new skin, such as **orange-red-3d-icons**. Your new preferences are then displayed for you to preview them as shown in the example in Figure 7-6.

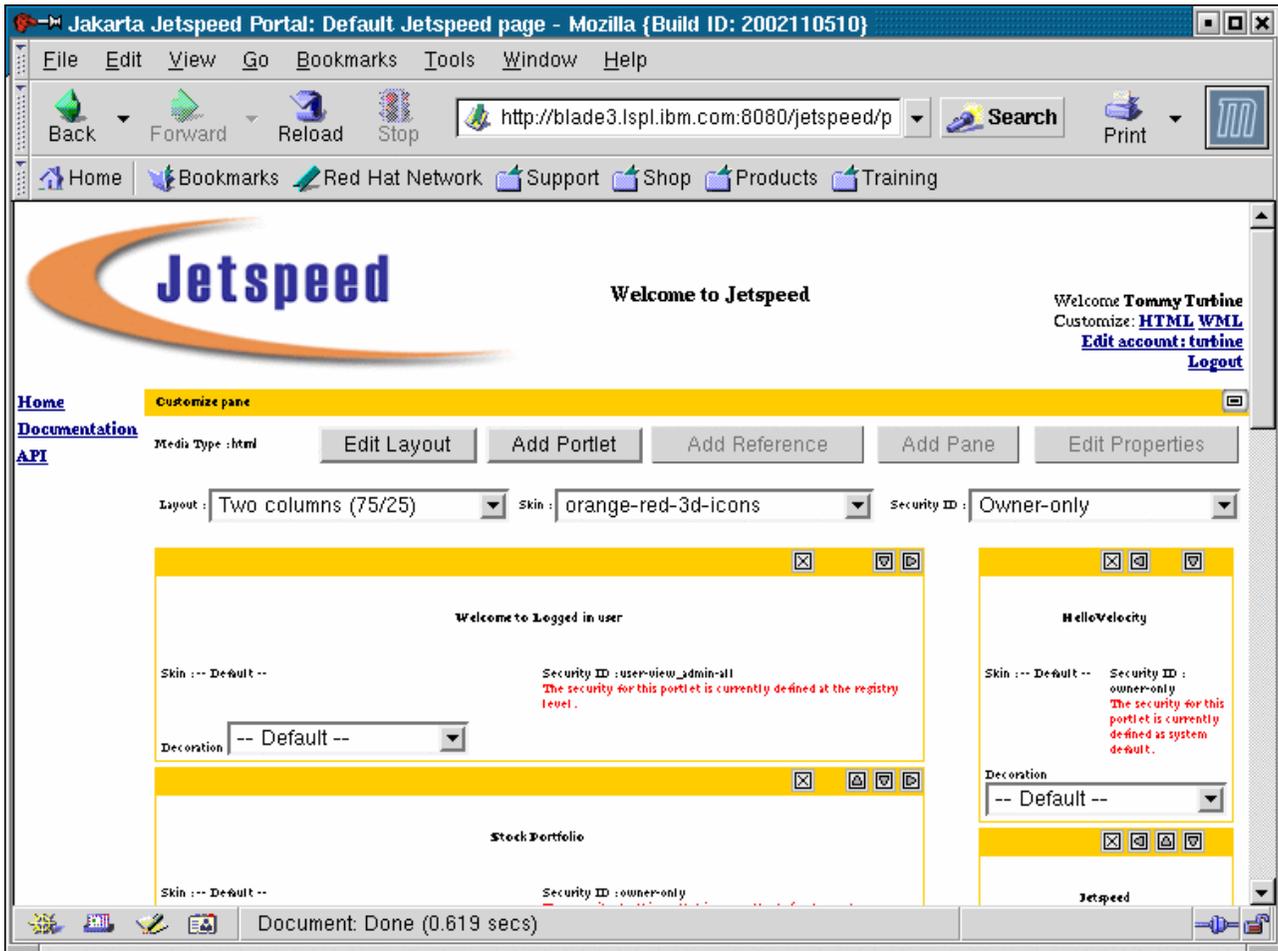


Figure 7-6 New layout preview

Then go to the bottom of the page and click **Save and apply**. When you return to your main page, the new choices are reflected.

You can delete portlets easily. Simply click the **X** on the control panel around your portlet, and it goes away.

To add portlets, go back to the Customize Pane and click **Add Portlet**. This gives you a list of all the portlets that are registered with your portal server as shown in Figure 7-7.

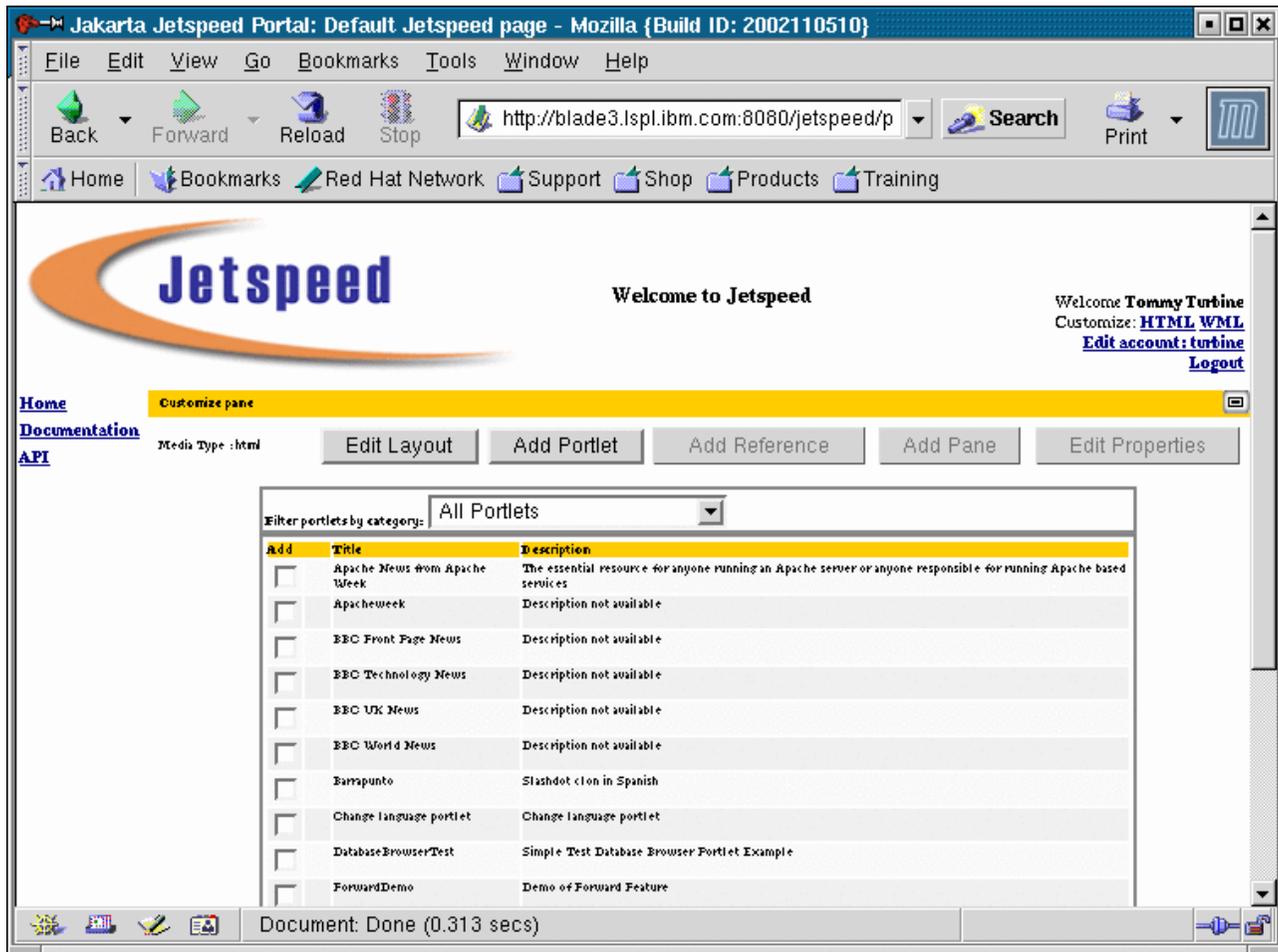


Figure 7-7 Jetspeed 'Add Portlet' UI

Check the portlets you want to add. There may be more than one page of portlets listed, so you can click Next to move to the next page, or filter the categories of portlets you want to see. You can also reposition portlets by clicking on the directional arrows. After you are satisfied with your selection, click **Apply**. When you return to your main page, you see your changes reflected accordingly.

Jetspeed: The developer's perspective

If you look at Jetspeed from a developer's perspective, you realize its advantages if you think about it as a framework. As a developer, you would develop a portlet that plugs into Jetspeed. The portlet uses the authentication mechanisms and the controls that Jetspeed provides. If you're working on a large portal site, with potentially hundreds of views to present to users, portlets make life so much easier.

More on portlets

A portlet is a Web component managed by a container and generates dynamic content. Portlets are platform-independent Java classes, similar to servlets. But while servlets usually interact directly with Web clients, portlets interact with Web clients indirectly through portals through a portlet container, like Jetspeed.

Portlets are specialized servlets that plug into and run in portals. Portlets are designed to be combinable in the larger context of a portal page. They rely on the portal infrastructure to function.

Portlets have different modes:

- ▶ View Mode for standard user interaction
- ▶ Customize (Edit) Mode for editing the portlet data, customizing the portlet content
- ▶ Maximized to view the portlet in full screen with no other portlets in view
- ▶ Minimized to only show the title bar
- ▶ Closed to close the portlet and remove it from the page

Jetspeed portlet tutorial

There is an excellent tutorial on the Web that covers several aspects of Jetspeed, including how to customize the portal, how to add new layouts, and how to develop different types of portlets. The tutorial itself is a work in progress, but it has sufficient information to get you through major portions of development. You find this tutorial at:

<http://www.bluesunrise.com/jetspeed-docs/JetspeedTutorial.htm>

Tips on working with the tutorial

If you want to work with the tutorial, you must have the following prerequisites to set it up:

- ▶ A working installation of Tomcat
- ▶ A working installation of Jetspeed
- ▶ The Ant build tool
- ▶ The jportal tutorial file, which you can get from:

<http://www.bluesunrise.com/jetspeed-docs/jportal.jar>

Installing the tutorial is not nearly as straightforward as we hoped. We had to make some minor adjustments for it to work properly. We outline the steps we took in the following sections.

Installing Ant

Ant is a Java-based build tool. It's similar to the `make` utility, but with some improvements. You can download Ant from the Web at:

<http://ant.apache.org>

You need it to work with the Jetspeed tutorial. Our file was `apache-ant-1.5.3-1-bin.tar.gz`.

1. In a temporary directory, unpack the archive:

```
tar xvf apache-ant-1.5.3-1-bin.tar.gz
```

This creates a directory called `apache-ant-1.5.3-1`.

2. Move this directory to `/usr/local/ant`. This was our choice to make it available to all the users of our development machine. Only the `bin` and `lib` directories of ant are really needed for you to work with ant, but we chose to put in all the files.

```
mv apache-ant-1.5.3-1 /usr/local/ant
```

3. Set up the environment. Edit `/etc/profile` and add `/usr/local/ant/bin` to the `PATH`.
4. Add a line `ANT_HOME=/usr/local/ant` in `/etc/profile`.
5. Still in `/etc/profile`, add `CLASSPATH` pointing to the location of the Tomcat `servlet.jar` file. Don't forget to export any environment variables you set.

For example, we added the following lines to our `/etc/profile`:

```
if ! echo $PATH | /bin/grep -q "/usr/local/ant/bin" ; then
    PATH="$PATH:/usr/local/ant/bin"
fi
ANT_HOME=/usr/local/ant
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC JAVA_HOME CATALINA_HOME ANT_HOME
CLASSPATH
```

Setting up the tutorial

By default, the tutorial assumes that you are working with a Jetspeed that you compiled from source. We thought it was easier to work with the Jetspeed binary. Here are the steps we used to start with the tutorial:

1. Copy the `jportal.jar` file into a working directory. We recommend that you create a directory for it, instead of unpacking it with other files. Unpack the `jportal.jar` file.

```
jar xf jportal.jar
```

2. Edit the main `build.properties` file. This is the file in the top directory of the `jportal` archive you unpacked. These are the contents of our `build.properties` file:

```
jetspeed_home=/var/tomcat4/webapps/jetspeed
catalina_home=/var/tomcat4
portlet_app_name=jportal
company=com.bluesunrise.jportal
jetspeed_jar=/WEB-INF/lib/jetspeed-1.4-b4.jar
jetspeed_war=/jetspeed.war
jetspeed_lib=/WEB-INF/lib/
jetspeed_conf=/WEB-INF/conf/
```

3. Make a copy `/var/tomcat4/webapps/jetspeed.war` to `/var/tomcat4/webapps/jetspeed/`. The `/var/tomcat4/webapps/jetspeed/` directory should be automatically unpacked by Tomcat from `jetspeed.war` the first time you access `http://hostname:8080/jetspeed`. If this is not the case, you may have to unpack it manually.

Why do you have to do this? Unfortunately, it's a quirk in how the ant build files were written. We found it much easier to just make a copy of `jetspeed.war` in that directory than edit the build files:

```
cp /var/tomcat4/webapps/jetspeed.war /var/tomcat4/webapps/jetspeed
```

4. Edit the file `torque3/build.xml`.

Find the line that says:

```
<property name="torque.lib.dir" value="${jetspeed_home}/lib"/>
```

Replace it with:

```
<property name="torque.lib.dir" value="${jetspeed_home}/WEB-INF/lib"/>
```

Find the line that says:

```
<unjar src="${jetspeed_home}/lib/torque-3.0.jar" dest="templates.scratch"/>
```

Replace it with:

```
<unjar src="${jetspeed_home}/WEB-INF/lib/torque-3.0.jar" dest="templates.scratch"/>
```

And that's it. You can run ant against the `build.properties` and `build.xml` file in your `jportal` directory. If you want to try it, you can simply type `ant` and it performs a preliminary compile.

When compiling Java applications using ant, you can be an ordinary user. But to deploy the application, you have to do it as *root*.

Developing a sample portlet

To show you the portlet development cycle, let's develop and deploy a sample portlet. We do this without the ant utility to simplify matters for this small example.

Setting up a Jetspeed development environment

The classes representing the Java portlet API are automatically unpacked from the jetspeed.was file when you first invoke Jetspeed from Tomcat. For developing portlets, it's better to work with the Jetspeed source archive. Download jetspeed-1.4-b4-src.zip from:

<http://jakarta.apache.org/builds/jakarta-jetspeed/release/v1.4b4/>

Unpack the file into your working directory.

Set up a CLASSPATH to point to the required portlet libraries:

```
export
CLASSPATH=/home/user/jetspeed-1.4-b4/WEB-INF/lib/jetspeed-1.4-b4.jar:/home/user/jetspeed-1.4-b4/WEB-INF/lib/turbine-2.2.jar:/home/user/jetspeed-1.4-b4/WEB-INF/lib/ecs-1.4.1.jar
```

If you want, you can also place this in your .bash_profile file so that it's automatically set every time you log on.

A "Hello, world!" portlet

This is the obligatory "Hello, world" program for portlets:

```
import org.apache.jetspeed.portal.portlets.AbstractInstancePortlet;
import org.apache.turbine.util.RunData;
import org.apache.ecs.ConcreteElement;
import org.apache.ecs.StringElement;
public class HelloWorldPortlet extends AbstractInstancePortlet
    public ConcreteElement getContent (RunData runData)
    {
        return (new StringElement ("Hello World!"));
    }
}
```

Save it as HelloWorldPortlet.java. Then, compile it:

```
javac HelloWorldPortlet.java
```

This produces a file called HelloWorldPortlet.class.

Deploying the portlet

To deploy the portlet, you must follow these steps:

1. Put the portlet class file into the CLASSPATH searched by Jetspeed. The simplest way to do this is by putting it in the WEB-INF/classes subdirectory of the Jetspeed directory. You may have to create this directory.

In our case, we enter:

```
mkdir /var/tomcat4/webapps/jetspeed/WEB-INF/classes
cp /home/user/HelloWorldPortlet.class /var/tomcat4/webapps/Jetspeed/WEB-INF/classes
```

2. You must create a registry fragment file in WEB-INF/conf so that Jetspeed knows about your portlet. The registry fragment file is any file with a .xreg extension.

Our portlet registry entry looks something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<registry>
    <portlet-entry name="HelloWorld" hidden="false" type="instance"
application="false">
```

```

<meta-info>
  <title>HelloWorld</title>
  <description>My first portlet</description>
</meta-info>
<classname>HelloWorldPortlet</classname>
<media-type ref="html"/>
</portlet-entry>
</registry>

```

3. Bring up the Jetspeed portal server, log in as a user, and customize your portlet. When you get to the “Add Portlet” menu, you see “HelloWorld” and its corresponding description in the list of available portlets.

Authenticating Jetspeed with LDAP

Jetspeed comes with a built-in “Hypersonic” database for managing user accounts. For larger sites, though, you may want to use a centralized database. For this purpose, Jetspeed supports several external databases including DB2®, MySQL, and PostgreSQL.

It’s also possible to authenticate against LDAP. This feature was added in Jetspeed 1.3-b2. This allows you to use your LDAP server as a central authentication for your Jetspeed portal and for your e-mail server, Samba server, etc.

We take you through the steps required to set up LDAP authentication with Jetspeed.

Setting up an LDAP server

First, you must set up a working LDAP server. We use the OpenLDAP server for this function. For simplicity and clarity, we work with a fresh copy of OpenLDAP, with no pre-existing entries, instead of integrating it with our existing entries. Review 4.3, “LDAP” on page 34, if you need some help here:

1. The Jetspeed source files ships with the LDAP schemas specific to Jetspeed. Copy these to the schema directories of OpenLDAP:

```
cp /home/user/jetspeed-1.4-b4/src/ldap/jetspeed.schema /etc/openldap/schema
```

2. Modify the /etc/slapd.conf file to include the following configuration file:

```

include      /etc/openldap/schema/core.schema
include      /etc/openldap/schema/cosine.schema
include      /etc/openldap/schema/inetorgperson.schema
include      /etc/openldap/schema/nis.schema
include      /etc/openldap/schema/redhat/rfc822-MailMember.schema
include      /etc/openldap/schema/redhat/autofs.schema
include      /etc/openldap/schema/redhat/kerberosobject.schema
include      /etc/openldap/schema/jetspeed.schema
database ldbm
suffix "ou=jetspeed,o=apache"
rootdn "cn=ldapadmin,ou=jetspeed,o=apache"
rootpw secret

```

3. Start your OpenLDAP server in the usual manner:

```
service ldap start
```

Populating your OpenLDAP server

The Jetspeed source directory contains an LDIF file that you can use to initially populate your OpenLDAP server. This file is found in the src/ldap subdirectory of the Jetspeed directory as jetspeed.ldif.

Jetspeed has an interesting quirk. All the text files are written in DOS format. If you use jetspeed.ldif as the input to ldapadd, only the first entry is read in. You have to convert jetspeed.ldif to UNIX format. You can use the dos2unix command:

```
dos2unix jetspeed-1.4-b4/src/ldap/jetspeed.ldif
```

After doing this, you can populate your LDAP directory:

```
ldapadd -D "cn=ldapadmin,ou=jetspeed,o=apache" -W -x < jetspeed.ldif
```

Configuring Jetspeed to use the LDAP server

The Jetspeed src/ldap directory contains a sample configuration file for Jetspeed's security properties, using LDAP as the authentication system.

Make a backup of the JetspeedSecurity.properties file in /var/tomcat4/webapps/jetspeed/WEB-INF/conf, and use the sample file to overwrite JetspeedSecurity.properties:

```
cp /var/tomcat4/webapps/jetspeed/WEB-INF/conf/JetspeedSecurity.properties
/var/tomcat4/webapps/jetspeed/WEB-INF/conf/JetspeedSecurity.properties.bak
cp /home/user/jetspeed-1.4-b4/src/ldap/LDAP-JetspeedSecurity.properties
/var/tomcat4/webapps/jetspeed/WEB-INF/conf/JetspeedSecurity.properties
```

Your JetspeedSecurity.properties should now read something like this example:

```
services.JetspeedSecurity.classname =
org.apache.jetspeed.services.security.JetspeedDBSecurityService
services.SecurityCache.classname = org.apache.jetspeed.services.security.SecurityCacheImpl
services.JetspeedSecurity.programmatic.cascade.delete = false
services.PortalAuthentication.classname =
org.apache.jetspeed.services.security.ldap.LDAPAuthentication
services.PortalAccessController.classname =
org.apache.jetspeed.services.security.registry.RegistryAccessController
services.UserManagement.classname =
org.apache.jetspeed.services.security.ldap.LDAPUserManagement
services.JetspeedSecurity.user.class = org.apache.jetspeed.om.security.ldap.LDAPUser
services.RoleManagement.classname =
org.apache.jetspeed.services.security.ldap.LDAPRoleManagement
services.JetspeedSecurity.role.class = org.apache.jetspeed.om.security.ldap.LDAPRole
services.GroupManagement.classname =
org.apache.jetspeed.services.security.ldap.LDAPGroupManagement
services.JetspeedSecurity.group.class = org.apache.jetspeed.om.security.ldap.LDAPGroup
services.PermissionManagement.classname =
org.apache.jetspeed.services.security.ldap.LDAPPermissionManagement
services.JetspeedSecurity.permission.class =
org.apache.jetspeed.om.security.ldap.LDAPPermission
services.ldap.classname=org.apache.jetspeed.services.ldap.LDAPService
services.ldap.host=localhost
services.ldap.port=389
services.ldap.sslport=636
services.ldap.basedn=ou/jetspeed%o/apache
services.ldap.managerdn=cn/ldapadmin%ou/jetspeed%o/apache
services.ldap.password=secret
services.ldap.anonymousbind=false
services.ldap.securityauthentication=simple
services.ldap.contextcache=false
# services.ldap.securityprotocol=ssl
# services.ldap.socketfactory=javax.net.ssl.SSLSocketFactory
# services.ldap.jndiprovider=com.sun.jndi.ldap.LdapCtxFactory
# services.ldap.saslclientpckgs=
services.ldap.limit=0
```

```
services.ldap.timeout=0
services.ldap.version=3
LocalWords:SHA
```

Open your Jetspeed portal. Your site should open. It won't be obvious just by looking at the portal that it's using the LDAP directory for authentication. To test this out, create a user and check the LDAP directory (using `gq`) to see if the user account was added.

Notes about LDAP authentication on Jetspeed

One of the drawbacks of LDAP authentication on Jetspeed is how it stores the user passwords. According to the documentation for Jetspeed 1.4-b4, the only currently supported authentication mechanism is UNIX crypt. In our tests, however, Jetspeed user passwords were stored using Cleartext authentication (as viewed from `gq`).

This can pose a problem in integrating other applications, which are similarly limited to a different type of cryptographic algorithm.



Cabinetry: Open source databases

Food is stored in refrigerators, cabinets, and pantries. These are all customized locations within a home that keep important goods safe, and organized. Maybe data is like bread, cheese, and meat. They all need to be kept in special compartments so they don't spoil.

Relational databases are a critical part of today's portals and Web-based applications. They facilitate the management of both user data and system data. They allow easy searches and easy manipulation.

For a long time, relational databases were offered only by commercial vendors. In recent years, we've seen the entry of many free open-source alternatives. While they may offer less features than their commercial equivalents, oftentimes these features are sufficient for the purposes of small- and medium-sized enterprises.

8.1 PostgreSQL, MySQL, and others

The open source community is working on several relational database projects. Currently the most popular ones are PostgreSQL and MySQL.

MySQL and PostgreSQL are included in the Red Hat Advanced Server distribution. Both are packaged for RPM installation and have extensions for Perl, PHP, Java database connectivity (JDBC), and open database connectivity (ODBC).

8.1.1 PostgreSQL

PostgreSQL began as an enhancement of the Postgres database management system, a next-generation database management system (DBMS) research prototype. Postgres was a project at the University of California, Berkeley, by Professor Michael Stonebraker and his graduate students. The actual PostgreSQL project was written by Andrew Yu and Jolly Chen in 1986.

PostgreSQL development is performed by a team of developers who all subscribe to the PostgreSQL development mailing list. The current coordinator is Marc G. Fournier. This team is now responsible for all development of PostgreSQL.

PostgreSQL has most features present in large commercial DBMSs, such as transactions, subselects, triggers, views, foreign key referential integrity, and sophisticated locking. Other features are user-defined types, inheritance, rules, and multi-version concurrency control to reduce lock contention.

PostgreSQL has fairly good performance. As with all tools, it is faster for some things, slower for others. In comparison to MySQL or leaner database systems, it is slower on inserts and updates because of transaction overhead.

For more information about PostgreSQL, see:

<http://www.postgresql.org>

8.1.2 MySQL

MySQL started as an extension to another open source database, mSQL. The original developers of MySQL, T.c.X. DataKonsultAB (and now known as MySQL AB) originally intended to use mSQL to connect to their database tables, but concluded that it was not fast or flexible enough. They wrote a new Structured Query Language (SQL) interface that was compatible with the application programming interfaces (APIs) of mSQL.

Although it is open source, MySQL is owned by MySQL AB. MySQL AB is the company that develops, supports, and markets the MySQL database server globally.

MySQL maximizes speed and customizability. As a database management system, it was designed for speed, compactness, stability, and ease of deployment. The separation of the core server from the storage engine makes it possible to run MySQL under strict transaction control or with ultrafast transactionless disk access, whichever is most appropriate for the situation.

To learn more about MySQL, see:

<http://www.mysql.com>

8.1.3 PostgreSQL versus MySQL

This is a subject of much debate. In the past, the general rule of thumb was that if your database operations were primarily reads and you wanted to maximize speed, you used MySQL. If your database operations involved a lot of read-write transactions and you wanted row-level locking and rollback, you chose PostgreSQL.

However, as both competing development teams improved their products, the distinctions have blurred. PostgreSQL speed has improved, and MySQL now better supports atomic transactions.

For a technical perspective to the debate, see the following documents:

- ▶ A Featurewise Comparison of PostgreSQL and MySQL
http://www.mysql.com/doc/M/y/MySQL-PostgreSQL_features.html
- ▶ Why not MySQL?
<http://openacs.org/philosophy/why-not-mysql.html>

Is one really better than the other? Ultimately, it depends on what you use the tool for.

For our purposes, we chose to work with MySQL to demonstrate its database replication features (a feature which PostgreSQL has, but to a less mature degree at the time of this writing).

8.1.4 Other open source databases

The open source database world is not limited to PostgreSQL or MySQL. There are others as well, although they are not as well known. Here is a sampling:

- ▶ Firebird
<http://firebird.sourceforge.net/>
- ▶ Interbase
<http://freshmeat.net/projects/interbase>
- ▶ SQLite
<http://freshmeat.net/projects/sqlite/>
- ▶ GNU SQL Server
<http://freshmeat.net/projects/gnysqlserver>

There are several more and Christopher Browne has compiled a short survey in his articles:

- ▶ “SQL Databases for Linux”
<http://freshmeat.net/articles/view/305/>
- ▶ “Non-SQL Databases for Linux”
<http://freshmeat.net/articles/view/307/>

8.2 Working with MySQL

MySQL already comes with an extensive and excellent documentation at:

<http://www.mysql.com/documentation/mysql/bychapter/index.html>

This manual shows you how to compile, upgrade, install, manage and optimize the MySQL database. It also gives a good tutorial on SQL.

Because of this, we have chosen to focus only on some Red Hat starting specifics as well as MySQL availability.

8.2.1 Required MySQL RPM packages

At the minimum, you need to install the following MySQL packages:

- ▶ `mysql-3.23.56-1.72`
- ▶ `mysqlclient9-3.23.22-6`
- ▶ `mysql-server-3.23.56-1.72`

As you can see, the MySQL client and the MySQL server are shipped separately. This means that you can install only the MySQL client on a remote machine and access a server over the network. However, you *must* install the MySQL client on the machine where you want to run MySQL server.

MySQL is not configured to start immediately. Therefore, you need to run `ntsysv` to have enable MySQL at bootup.

8.2.2 Starting MySQL the first time

You only need to perform these steps when you install and set up MySQL for the very first time:

1. Log in as the root user and execute the MySQL startup script:

```
# service mysqld start
```

This creates the required MySQL directories and the MySQL user account.

2. Access the MySQL database and check the settings. You can use the MySQL client. At the command prompt, invoke the environment:

```
# mysql
```

This takes you into the MySQL client environment (Figure 8-1).

```
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 3.23.56-log
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```

Figure 8-1 MySQL client environment

3. To see what databases are created, type the following command in the MySQL environment:

```
show databases;
```

The semicolon is the command terminator in MySQL, so you must include it.

The `mysql` database shown is the system settings of the entire MySQL database server that you are running. The `test` database is a dummy database used for testing purposes. See Figure 8-2.

```
mysql> show databases;
+-----+
| Database |
+-----+
| mysql   |
| test    |
+-----+
2 rows in set (0.00 sec)
mysql>
```

Figure 8-2 MySQL session

4. To examine the contents of the mysql database, follow the command sequence shown in Figure 8-3.

```
mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
mysql> show tables;
+-----+
| Tables_in_mysql |
+-----+
| columns_priv    |
| db              |
| func            |
| host            |
| tables_priv     |
| user            |
+-----+
6 rows in set (0.00 sec)
mysql>
```

Figure 8-3 Examining the contents of a database

5. To exit from the MySQL client, type either `exit`, `quit`, or `\q`.

8.2.3 Securing MySQL

When we invoked the `mysql` client, it did not prompt us for a user name or password. It immediately gave us access to the database as the `root` user. In fact, any user who runs the `mysql` client can have this same level of access. This is very bad security!

Unfortunately, this is the case because of the choices that Red Hat made. You have to lock down your database by adding a password to the `root` account on the database. To change the passwords on your newly established MySQL server, follow the sequence shown in Figure 8-4.

```
# mysql -u root mysql
mysql> SET PASSWORD FOR root@localhost=PASSWORD('new_password');
```

Figure 8-4 Changing the passwords

From here on, every time you log on to MySQL using the client, you must supply the `-u` and the `-p` flags:

```
# mysql -u root -p
```

This prompts you for the password for the user `root` before it allows you to enter the interactive MySQL environment.

8.3 MySQL replication

Starting with version 3.23.15, MySQL supports one-way replication internally. One server acts as the master, while the others act as slaves. Updates to the master are automatically updated in the slaves. Since this is one-way replication. However, updates to slaves are not replicated in the master.

The master server keeps a binary log of updates and an index file to binary logs to keep track of log rotation. The slave, upon connecting, informs the master where it left off since the last successfully propagated update, catches up on the updates, and then blocks and waits for the master to notify it of the new updates.

8.3.1 Uses of replication

The one-way replication of MySQL is a good first step toward a robust and highly-available database. It's not perfect, since some important things are missing. For example, if a master dies, there's no way to automatically elevate a slave to become the new master.

Nevertheless, this replication features is still quite useful in a number of situations. The following sections highlight two examples.

Improving MySQL performance

Most Web-based databases are oriented toward read and query operations instead of write operations. You can improve performance of your overall database by setting up a cluster of MySQL servers. All updates are performed on the master, but the read operations can be distributed across the slaves.

Non-disruptive backups

Instead of bringing the entire database down for backup, you can disconnect a slave server from the replication cluster and perform the backup from there. When you reconnect it to the master database, the slave catches up on all the transactions that occurred during the backup.

8.3.2 Setting up replication

Setting up servers for replication is well-documented in the MySQL manual. We repeat the steps here in condensed format.

To set up replication, set up two blades, both as MySQL servers. In our laboratory setup, we installed these on `blade1` and `blade2`. We start with fresh servers, and no databases are set up on either server as yet.

Replication Master: blade1

First, we set up the replication master. We set up a replication user on the master. We create a user named `repl` that can access the master from any host:

1. From the mysql client environment, enter:

```
mysql> GRANT FILE ON *.* TO repl@"%" IDENTIFIED BY 'password';
```

Our repl@"%" parameter is a wildcard. It essentially allows a user repl from any machine, as identified by a password, to replicate using this server as master.

2. Edit the MySQL server configuration file, /etc/my.cnf. It should include two additional lines:

```
log-bin
server-id=1
```

The log-bin directive tells the server to keep a binary log of transactions. The server-id is a unique identifier for all servers in the MySQL cluster. We use a server-id of 1 to designate our master.

3. Restart the replication master.
4. From the mysql client, enter the following command:

```
SHOW MASTER STATUS;
```

This should give you a report similar to the one in Figure 8-5.

```
mysql> show master status;
+-----+-----+-----+-----+
| File           | Position | Binlog_do_db | Binlog_ignore_db |
+-----+-----+-----+-----+
| blade1-bin.005 | 73       |               |                   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Figure 8-5 Output of SHOW MASTER STATUS

5. Note this information for now. We use this to set up our slave server later.

Replication Slave: blade2

Next, we set up the replication slave:

1. Edit the MySQL slave configuration file /etc/my.cnf. It does not need to have the log-bin directive, but it does need a server-id directive (different from the server-id used for the master):

```
server-id=2
```

2. Restart the MySQL slave.
3. Run the following commands on the slave from the mysql client:

```
mysql> CHANGE MASTER TO MASTER_HOST=blade1.lsp1.ibm.com
MASTER_USER='repl',
MASTER_PASSWORD='password',
MASTER_LOG_FILE='blade1-bin.005',
MASTER_LOG_POS=73;
```

The values for MASTER_LOG_FILE and MASTER_LOG_POS are taken from the output of the SHOW MASTER STATUS command earlier, when we were working on the master.

4. Start the slave threads:

```
mysql> START SLAVE;
```

Testing our setup

Now, let's test our setup:

1. On the master server, create a test table and populate it with entries. Figure 8-6 shows an example of our session.

```
mysql> use test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
mysql> create table users (fname char(20), lname char(20));
Query OK, 0 rows affected (0.02 sec)
mysql> describe users;
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| fname | char(20)  | YES  |     | NULL    |       |
| lname | char(20)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
mysql> insert into users values ('Gary', 'Gygax');
Query OK, 1 row affected (0.00 sec)
mysql> insert into users values ('Nichelle', 'Nichols');
Query OK, 1 row affected (0.00 sec)
mysql> insert into users values ('Stephen', 'Hawking');
Query OK, 1 row affected (0.00 sec)
mysql> insert into users values ('Al', 'Gore');
Query OK, 1 row affected (0.00 sec)
mysql> select * from users;
+-----+-----+
| fname  | lname  |
+-----+-----+
| Gary   | Gygax  |
| Nichelle | Nichols |
| Stephen | Hawking |
| Al     | Gore   |
+-----+-----+
4 rows in set (0.00 sec)
```

Figure 8-6 Populating a database

2. Connect to the slave SQL server.

3. From a mysql client, check if the table you created in the master was also created automatically in the slave. Figure 8-7 shows a sample of our session.

```
mysql> use test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| users          |
+-----+
1 row in set (0.00 sec)
mysql> select * from users;
+-----+-----+
| fname | lname |
+-----+-----+
| Gary  | Gygax |
| Nichelle | Nichols |
| Stephen | Hawking |
| Al    | Gore  |
+-----+-----+
4 rows in set (0.00 sec)
```

Figure 8-7 Testing the setup

As you can see, all the data was automatically replicated.

8.4 Using MySQL replication

Currently, it is only practical to do one-way replication since there is no conflict resolution in MySQL similar to LDAP's replication tools. This set of features allows MySQL to be configured for:

- ▶ Horizontal scaling
- ▶ High availability
- ▶ Live backups
- ▶ Dedicated reporting servers

8.4.1 Load balancing MySQL queries with a workload manager

To improve performance, you can use a workload manager to distribute queries. A workload manager is a layer two or layer three redirector. It can be an appliance such as Cisco's LocalDirector, or it can be delivered using software on a dedicated server.

The three Web clients (standard browsers) shown in Figure 8-8 are connecting to a HTTP based application server. The application server queries needed data, using SQL, from the workload manager. The workload manager selects the next MySQL server to use in a round-robin fashion. The MySQL server is queried, with the results flowing back first to the workload manager, then the application server, and finally the Web client. The workload manager skips any host that is failed or not running MySQL.

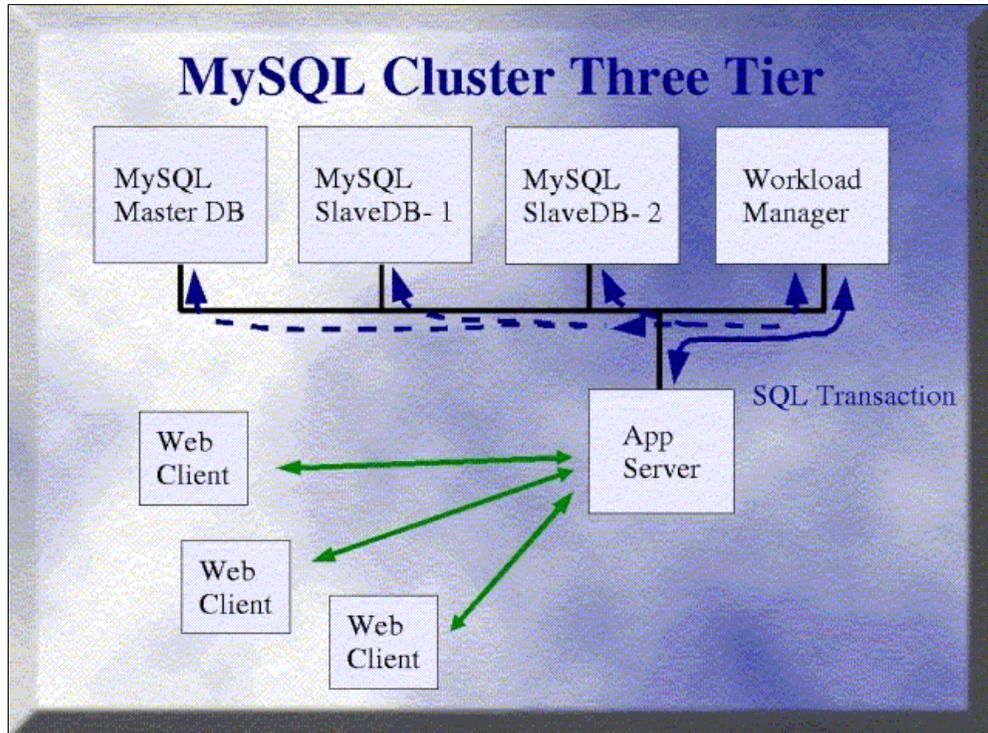


Figure 8-8 MySQL cluster

8.4.2 Application logic versus cluster logic

Many important Internet applications have high-level concepts of failover, distributed processing, and other clustering behavior. Depending on a project's requirements, it may be possible to put all of the high availability and scaling logic directly in the application server.

For third-party applications, it may not be possible to add cluster logic (because you do not have access to the source). Given the relative maturity and low expense of the cluster technology, the price/value proposition for moving the cluster logic into the application starts to look thin.

8.4.3 Example: Using application logic

This sample code shown in Example 8-1 is adapted from an article by Michael Tanoviceanu in *PHPBuilder*. It shows how a PHP database query can be modified to take into account multiple servers.

Example 8-1 Code using application logic

```
<?php
function db_connect_plus(){
    $username = "username";
    $password = "password";
    $primary = "blade1.1spl.ibm.com";
    $secondary = "blade2.1spl.ibm.com";
    $timeout = 15; // timeout in seconds
/* Attempt to connect to the database on the primary server at port 3306
and with a timeout of 15 seconds.
```

```

    If it works out okay, generate a link statement using primary as our actual database
server. */
    if($fp = fsockopen($primary, 3306, &$errno, &$errstr, $timeout)){
        fclose($fp);
        return $link = mysql_connect($primary, $username, $password);
    /* If that doesn't work out, try it for the secondary server. */
    } elseif($fp = fsockopen($secondary, 3306, &$errno, &$errstr, $timeout)){
        fclose($fp);
        return $link = mysql_connect($secondary, $username, $password);
    } else
    /* Oops, something wrong! */
    return 0;
?>

```

In this code, we use **fsockopen** to check if the MySQL server is still responding. We have to go through this route because the MySQL query statement unfortunately does not have a configurable timeout value.

You can find article by Michael Tanoviceanu in *PHPBuilder* on the Web at:

<http://www.phpbuilder.com/columns/tanoviceanu20000912.php3>

8.4.4 Horizontal scaling and MySQL replication

In Figure 8-9, we simplify things by looking closer at the relationship between the application server and the database server cluster. The application server is a single, very busy SQL client. From the MySQL servers' perspective, all the queries come from the workload manager. For the master database, it appears that all the updates come from the application server, which to it looks like an SQL Client.

Here we assume the database is relatively small (<2 GB) and that updates are a small fraction of the transactions (<= 10%). We also assume that nothing is shared. This is commonly described as a *shared-nothing-server-farm*.

Updates are made only to the master database. The two arrows stemming from MySQL Master DB on the left show one-way MySQL replication. Any updates made to the Master DB are automatically forwarded to the Slave DB instances. If a slave is temporarily unavailable, it automatically catches the next time it successfully syncs with the master. For upcoming versions of MySQL, more sophisticated forms of replication are being proposed such as quorum votes and dynamic master selection.

Growing processing power for the cluster is simply a matter of adding more MySQL slave instance machines and configuring the workload manager/IP redirector to include them in the pool. The scaling results are, in theory, linear up through the point where the shared network media or the workload manager run out of resources. In practice, using 10 modest servers should deliver almost a straight 10 times the performance over any one of the nodes.

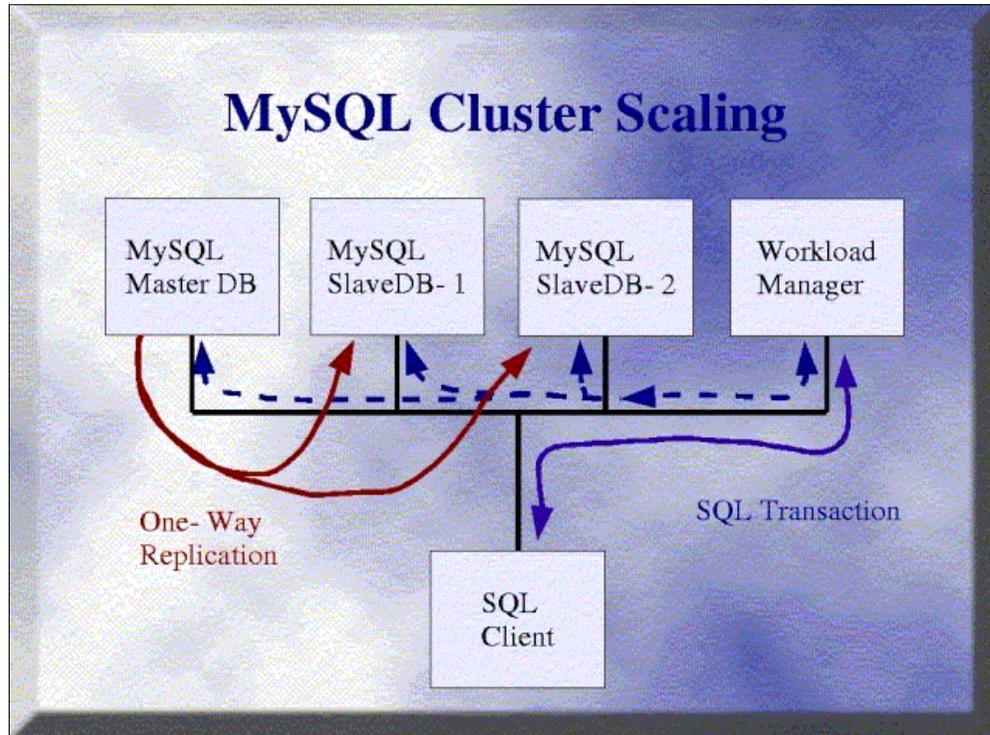


Figure 8-9 Scaling with a MySQL cluster

8.4.5 High availability

One of the potential applications for a MySQL cluster is high availability. Given a master/slave replicated pair and a workload manager, it's possible to provide uninterrupted service to a MySQL database. Most of the redirector/load balance solutions include support for automatic failover when there is a failure. MySQL's replication likewise notices a failed slave and queues any updates for later. When the slave is brought back online, the workload manager begins to include it for new work, and the MySQL master automatically syncs up.

A production system is likely to use redundant workload managers, hubs, network connections, etc. as well. For clarity, Figure 8-10 shows the flow of both the replication and SQL transactions when a node (SlaveDB-2) fails.

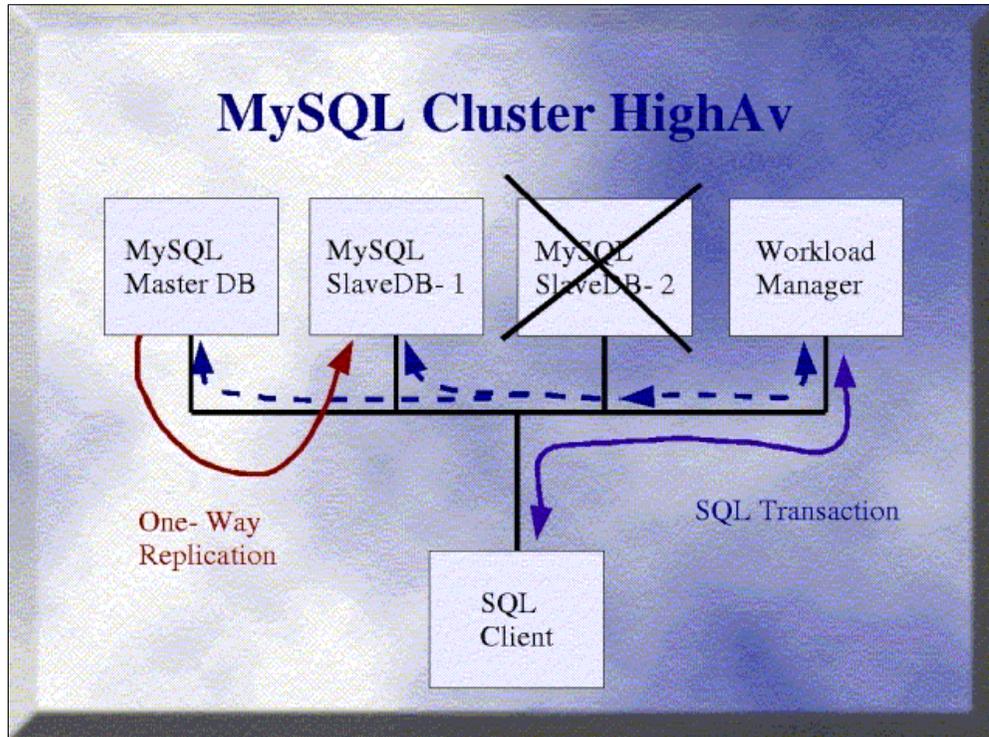


Figure 8-10 High availability

8.5 What if the master fails?

The situations that we've explored make the optimistic assumption that slave MySQL servers fail, but not the master MySQL servers. Unfortunately, that's not always true. Master servers can also fail.

It's not such a problem if a slave server fails. If it does, the SQL client or the workload manager can simply skip the failed server when it performs a read database query.

But what if the master fails? Can we perform updates to the slave and have these changes reflected in the master when it comes back up? Fortunately, we can.

8.5.1 Setting up a mutual master-slave relationship

One way to address the master failure scenario is by creating a mutual master-slave relationship between two servers. In our scenario, we set up blade1 as our master and blade2 as our slave. At the same time, we set up blade1 to be the slave of blade2.

Using the setup that we already have with blade1 and blade2, the changes we expect to make are outlined in the following sections.

On the slave server, blade2

Complete the following steps:

1. On blade2, set up a replication user on the master. Create a user named *rep1* that can access the master from any host. From the mysql client environment, enter:

```
mysql> GRANT FILE ON *.* TO rep1@"%" IDENTIFIED BY 'password';
```

- On blade2, add the following lines to /etc/my.cnf:


```
log-bin
log-slave-updates
```
- Check the master status using the following command:


```
SHOW MASTER STATUS
```

Note the information here since we will feed it into blade2 later.
- Restart the MySQL server.

On the master server, blade1

No significant changes are necessary on blade1. The only change that is required is to tell blade1 who its master is going to be:

```
mysql> CHANGE MASTER TO MASTER_HOST=blade2.lsp1.ibm.com
MASTER_USER='rep1',
MASTER_PASSWORD='password',
MASTER_LOG_FILE='blade2-bin.001',
MASTER_LOG_POS=445;
```

Then, restart the MySQL server.

What we just did

What we've actually done is to create a reverse relationship between blade1 and blade2. Any changes made in blade1 are reflected in blade2. And any changes made in blade2 are reflected in blade1.

The directive log-slave-updates tells the slave to log the updates done by the slave SQL thread to the slave's binary log. It requires that the slave be started with binary logging enabled, using the log-bin directive.

8.5.2 Chaining servers

We can extend our two-server MySQL setup to include several more nodes. In this case, we chain the servers, as shown in Figure 8-11. Server A updates Server B. Server B updates Server C. And Server C updates Server A.

Thanks to server IDs, which are encoded in the binary log events, Server A will know when the event it reads was originally created by Server A, so Server A will not execute it and there will be no infinite loop.

Issues with this approach

This circular setup only works if you perform non-conflicting updates between the tables. If you insert data in Servers A and C, never insert a row in Server A that may have a conflicting key with a row insert in Server C. You should also not update the same rows on two servers if the order in which the updates are applied matters.

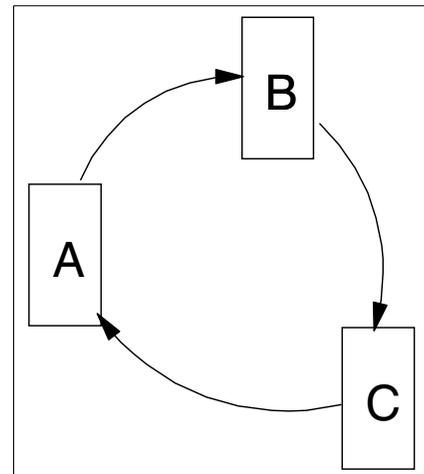


Figure 8-11 Chaining MySQL servers

To get around this problem, you have to write your client code to take care of the potential problems that can happen from updates that occur in different sequence on different servers.

The other issue with this approach is the lack of an automatic master server election system. If one of the servers in this cluster goes down, the chain is broken. You have to manually point the master for the server to another server.

Master election is a feature that is planned to be introduced in the next releases of MySQL.

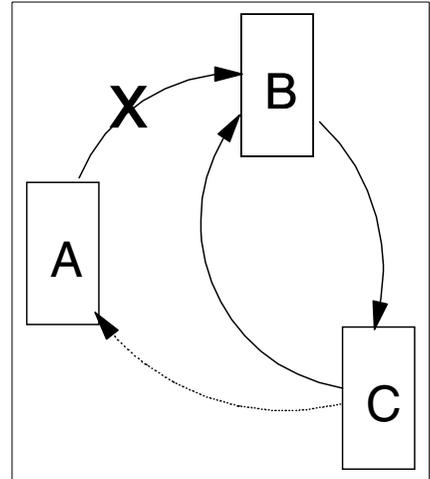


Figure 8-12 Pointing to a new master

8.5.3 How far do we go?

MySQL provides us with the elements of high availability. There are still points that require a great deal of work to make it truly bullet proof. For a system administrator or IT architect implementing an open-source database solution, there is a temptation to address these with customized scripts that automate the failover and failback process.

While some form of automatic failover availability is quite possible using scripts, we strongly believe that this is not the right approach. Customized scripts that are not part of the standard MySQL package greatly add to the complexity of the setup, adding more points of failure. The accrued benefits are minimal.

MySQL is a sufficiently robust database to handle the needs of a small- or medium-sized company. It has been used successfully for high-traffic sites such as Yahoo! Finance and Slashdot (until they moved to DB2). The high availability using replication should be treated as some form of insurance, but not as total protection against failure.



Security

Security systems are often the last things put into a home. Door locks and smoke detectors are basic and often the only security feature of modern domestic dwellings.

Any experienced IT professional can tell you that security has to be designed and built into an infrastructure from the ground up. This redbook has discussed security issues with many of the components we have presented. Linux itself is considered by most IT professionals to be a secure operating system. Using products that are inherently secure is a good first step, but when implementing an infrastructure you should use great care to understand the security needs and concerns of the people using the infrastructure. At the very least, you need to understand the impact of a security failure. Understanding the impact of someone stealing the information managed by the infrastructure, or corrupting the infrastructure itself, is going to cost money. The level of security measures put in place should be appropriate to the potential damage and liability done due to a security failure.

Computer and network security is not like a light switch that is either on or off. Security is gradient. There is a wide range of precautions, strategies and technologies to employ to provide more secure computing environments. Providing high levels of security usually makes a system more expensive to deploy and less convenient to work on. When designing security for your infrastructure you, focus on what is at stake. Systems that store confidential customer information or commercial transactions warrant spending more time and money on security. For other Web applications, it is embarrassing if the information provided is defaced or the system is brought down. The reality is that there is much less at stake.

This chapter discusses basic security practices that are built in to Linux and BladeCenters. There is incremental cost or loss of convenience to employ these techniques. The practices described are good enough for many intranet and Internet applications. If you are designing an application that will, for example collect customer's payments or medical information or interact with banks, hire a computer security expert.

9.1 Good practices

The easiest and most effective first step to securing your systems is to reduce, to a minimum, the services that you provide. Do not run application services that you don't absolutely need just because they are available and can install in a default installation. This practice is based on the principle that all software contains bugs, and bugs may compromise the systems reliability and security. The less software you run, the fewer bugs are encountered, and the more reliable and secure the system is.

Don't run the X window environment on a host that is acting as a server. The X windows system relies on other services, XFSD, the X font server and remote procedure calls (RPC).

Other services that are often installed by default by Red Hat Linux distributions that usually do not need to run on a server are:

- ▶ APMD
- ▶ GPM
- ▶ LPD
- ▶ Anacron
- ▶ XFS
- ▶ Autofs
- ▶ RHNSD
- ▶ Sendmail

To control which daemons are started on a host, first confirm the default run level of the system. You control the default run level by editing the `/etc/inittab`. We recommend that you run server hosts in run level 3. The beginning of the inittab file should look like this:

```
# Default runlevel. The runlevels used by RHS are:
# 0 - halt (Do NOT set initdefault to this)
# 1 - Single user mode
# 2 - Multiuser, without NFS (The same as 3, if you do not have networking)
# 3 - Full multiuser mode
# 4 - unused
# 5 - X11
# 6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:
```

Use the **chkconfig** command to remove symbolic links that link from the run level directories to the daemon start up scripts. For example, to prevent the line printer daemon (LPD) from starting (at all run levels, not just run level 3), issue this command as the root user:

```
chkconfig --del lpd
```

Or you can use **chkconfig** to manage the in each run level independently. For example to keep the X font server from starting at run level 3, issue this command as the root user:

```
chkconfig --level 3 xfs off
```

Another good security practice is to run application daemon processes with ordinary user privileges unless they absolutely need root privilege. Apache can run with the user ID of any user on the system, but you should never run it as root.

Named (the Internet domain name system daemon) should also run as its relatively unprivileged user ID.

In a default Red Hat Advanced Server installation, the Apache Web server and the domain name system run with the user IDs `apache` and `named`, respectively. This is a good security practice that you should emulate when you install third-party application servers.

The Linux extended Internet services daemon (`xinetd`) runs as the root user so you should disable as many services it provides as possible. Change directories to `/etc/xinetd.d`. There is a collection of configuration files in that directory with each file named for a service. Unless you know that you absolutely need that service, make sure the corresponding configuration file contains the line `disable = yes`.

User accounts, just like services, should be pared down to a minimum. There should be a good reason for each account on the system. Do not use shared or role-based accounts. Role-based accounts (common database user accounts such as `dbuser`, or financial application users all using the account named *finappuser*) are much less traceable and discourage accountability.

Administrators should have user logins rather than logging into hosts as root. Administrators should login as their own user ID and then switch to the root user ID with the `su - root` command. Using the `su` command generates a syslogd, the system logging daemon, which is a message that contains the user login name of the user that issued the command. That forces the administrators to be more accountable for their logins and passwords. It also may help in tracing the mechanism of attack if a hacker compromises a host.

9.2 OpenSSH

OpenSSH is a replacement for the `rlogin` and `rsh` for daemons for executing commands on remote systems. It's made up of a collection of tools that provide for strong authentication and encrypted communications between two hosts on an insecure network. Public-key cryptography is employed for host authentication. It can be used for user authentication. Additional security is provided by streaming cryptography, which encrypts traffic between client and host.

Do not use Telnet, `rsh`, and `rlogin` to connect to Linux hosts. The login, password, and any other sensitive configuration parameters are sent through the network connection in clear text. Use `ssh` instead for stronger authentication and security.

The OpenSSH package is made up of a collection of tools of which the most important are:

- ▶ **sshd** is the daemon process that authenticates other hosts and provides logins.
- ▶ **ssh** is the client that invokes commands on remote machines that replaces `rsh`.
- ▶ **scp** is the file copying client that replaces `rcp`.
- ▶ **ssh-keygen** is a tool for creating public-key pairs.
- ▶ **ssh-agent** is a program to hold a private key in memory.
- ▶ **ssh-add** is a tool to add a private key to the `ssh-agent`.

The first time `sshd` is started on a Red Hat Linux system (using the startup scripts provided by the distribution), a host public-key pair is automatically generated and installed in the directory `/etc/ssh`. This is the host ID key and it is used to uniquely identify each host.

The first time an `ssh` client connects to a new host, the client is offered the public that host's public key. The client has the opportunity to accept or reject the key. If the client has a high confidence that the host you connected to and the key are authentic, the client should accept the key. The client then may use the password login to the host via an encrypted session. The `ssh` client saves the host's public key in `$HOME/.ssh/known_hosts`. Then in following connections to that host, if the public key no longer works, the client complains that a man-in-the-middle attack may be happening and does not continue with the connection to the

host. If the host key changed for a legitimate reason, for example, you reloaded the host and didn't preserve the host key, remove that hosts entry from the `known_hosts` file to allow future connections.

Using RSA style authentication is a more secure alternative to password authentication. Each user that wants to use RSA style authentication needs to generate their own public-key pair.

Run the `ssh-keygen` command to create a user public-key pair:

```
mkdir .ssh
chmod 700 .ssh
ssh-keygen -t dsa -b 2048 -f ~/.ssh/id_dsa -N 'key pass phrase'
```

The commands produce two files in the `.ssh` directory:

- ▶ **id_dsa**: This file is the private key. Although it is encrypted and locked with the key pass phrase, it should be carefully guarded. The permissions on the private key are 600 and should stay that way.
- ▶ **id_dsa.pub**: This file is the public key. The public key should be distributed to the target hosts running `sshd` that the OpenSSH clients log on to. On the target hosts, copy the `id_dsa.pub` file to the users `$HOME/.ssh/authorized_keys` file. The next time the user connects from the system where they created the key pair to a system with the `authorized_keys` file, they are asked for the private key pass phrase instead of the login password. This indicates that the `ssh` daemon and client are employing RSA style authentication instead of password authentication.

The user can avoid typing the private key pass phrase for `ssh` connection using the `ssh-agent` utility.

9.3 Segregate networks

Each blade in a BladeCenter chassis has two separate 1000BT interfaces. If the BladeCenter chassis has two network modules, then two physically separate networks can be connected to each blade. In a Linux system, the two network interfaces appear in the operating systems as `/dev/eth0` and `/dev/eth1`. We recommend that you keep the administrative network connections on a separate network from the network that provides services. Management connections are logins, such as `ssh` and Simple Network management Protocol (SNMP). Services are protocols, such as HTTP, IMAP, and CIFS, which is also referred to as Samba. By only allowing certain services on each network, you can prevent users of the Web portal services from even connecting to daemons that provide logins.

In our examples, we use the `eth0` interfaces of the blades configured on the class C IP network of 10.0.0.0 for the management network. The services are bound to the `eth1` interfaces and are on the 192.168.2.0 class C IP network. We use IPChains, a packet filtering function available the Linux 2.4 kernel distributed with Red Hat Advanced Server 2.1, to enforce the separation. Setting up IPchains is explained in the following section.

Having just two networks, one for services and another for management, is probably sufficient for intranet applications. Web portals that are connected to the Internet may require a higher level of security.

9.4 IPChains

IPChains is the packet filtering firewall feature that is available with the Linux 2.4 kernel and distributed with Red Hat Linux Advanced Server. In fact, there are two mutually exclusive packet filtering technologies distributed in Red Hat Advanced Server 2.1, IPChains and *IPTables*. IPTables has a superset of functionality that IPChains. It allows for both stateless and stateful packet filtering. IPChains is easier to understand and use. It still provides a great deal of control over filtering. The default firewall setup in the Red Hat Advanced Server 2.1 installation configures the IPChains firewall. This section covers IPChains, and not IPTables.

This section shows examples of how to setup IPChains so that only certain protocols may arrive on specific network interfaces. As explained earlier, this is a way to enforce the segregation of the management and services networks.

The most common actions taken on a packet by the IPTables packet filter are to accept and allow the packet through the interface, or drop and make the packet just disappear. By installing a chain of rules that every packet must traverse before it is accepted, you can control the types of connections that are allowed to appear on each Ethernet interface.

There are three built-in chains:

- ▶ **Input chain:** For packets coming into a network interface
- ▶ **Output chain:** For locally generated packets exiting a network interface
- ▶ **Forward chain:** For packets routing through a system

You can set a default policy for each chain and then append and add to the end, or insert and add to the beginning, rules to each chain. The rules are a set of criteria to match against each packet, source address or port, destination address or port, protocol, or physical interface. The rules also specify a target if a packet matches a specific rule. The two targets we use for packet filtering are:

- ▶ **ACCEPT:** Allows the packet to continue in or out of that interface.
- ▶ **REJECT:** Drops the packet and sends an ICMP message back to the sender indicating that the packet was dropped.

9.4.1 Creating rules

Red Hat Advanced Server 2.1 comes with a utility to create a basic input filtering chain called **lokkit**. Confirm that the lokkit package is installed. If it isn't installed, mount the distribution file system and install the package:

```
rpm -i lokkit-0.50-6
```

As the root user, run the command **lokkit**.

Using the Tab key, you can toggle between the various options. Use the Space bar to select options. The return key activates the buttons at the bottom of the page. Select the option for the security level **High**. Then tab to the **Customize** button and press Enter.

In the customization display, confirm that neither device eth0 nor eth1 is selected as a trusted device.

In the Allow incoming section, select **DHCP** and **ssh**. In the other ports section, type:

```
snmp:udp
```

This produces a basic set of IPChains rules in the file `/etc/sysconfig/ipchains`. The file should look like Example 9-1.

Example 9-1 Firewall configuration written by the lokkit tool

```
# Firewall configuration written by lokkit
# Manual customization of this file is not recommended.
# Note: ifup-post will punch the current nameservers through the
#       firewall; such entries will *not* be listed here.
:input ACCEPT
:forward ACCEPT
:output ACCEPT
-A input -s 0/0 -d 0/0 161 -p udp -j ACCEPT
-A input -s 0/0 -d 0/0 22 -p tcp -y -j ACCEPT
-A input -s 0/0 67:68 -d 0/0 67:68 -p udp -i eth0 -j ACCEPT
-A input -s 0/0 67:68 -d 0/0 67:68 -p udp -i eth1 -j ACCEPT
-A input -s 0/0 -d 0/0 -i lo -j ACCEPT
-A input -s 10.0.0.19 53 -d 0/0 -p udp -j ACCEPT
-A input -s 0/0 -d 0/0 -p tcp -y -j REJECT
-A input -s 0/0 -d 0/0 -p udp -j REJECT
```

This provides a start for creating a good input filter. To limit the snmp and ssh connections to the internal network, edit the first two lines as shown here:

```
-A input -s 10.0.0.0/24 -d 0/0 161 -p udp -i eth0 -j ACCEPT
-A input -s 10.0.0.0/24 -d 0/0 22 -p tcp -y -i eth0 -j ACCEPT
```

If the host provides other services, such as a Web server, you need to open the appropriate port. To do this, add a line to `/etc/sysconfig/ipchains`. For example, to open access to the port that the HTTP daemon runs on, usually port 80, add a line to the top of the block of rules:

```
-A input -s 0/0 -d 0/0 80 -p tcp -y -j ACCEPT
```

To find out which port a daemon process runs on, consult the file `/etc/services`. For some protocols, you need to allow both TCP and UDP connections. You do this by creating two rules, one for UDP and one for TCP.

The component of Linux that implements these rules is a *kernel module*. This module that implements these rules must be notified when a change is made to `/etc/sysconfig/ipchains`. You accomplish this by running the init script:

```
/etc/init.d/ipchains start
```

You also find a link from the init run level 3 directory to this init script. This starts IPChains automatically when the system restarts. To apply the IPChains rules, issue the command as:

```
root /etc/init.d/ipchains start
```

To remove the rules from the kernel, type the command:

```
/etc/init.d/ipchains stop
```

Use care when editing and reapplying the rules in the `/etc/sysconfig/ipchains` file from a network attached machine. It's possible to apply rules that filter out your current connection. This can make it impossible for you to issue the command to remove the filter without going to the console. To avoid this, write a shell script called `test-rules.sh`, as shown in this example:

```
#!/bin/bash
/etc/init.d/ipchains stop
/etc/init.d/ipchains start
sleep 60
/etc/init.d/ipchains stop
exit 0
```

This allows you to test the new rules for 60 seconds. Then it removes the rules.



Household maintenance: System management and application development

Every home owner would like to have a butler, maid, gardener, or handyman to keep their household orderly, clean, and smoothly running. Fortunately with computer systems, the chores associated with routine maintenance are, often, fully automatic.

In the open source world, tools like `snmpd` inform us about every little thing, and tools like Multi Router Traffic Grapher (MRTG) provide a clean interface to this information. These are but two brooms in the open source closet of maintenance tools. We recommend that you also consider MON for system monitoring. If you are managing a large number of systems, consider commercial tools such as IBM Director. For the management of really big problems consider Tivoli® Enterprise™ Manager.

The need for new computers is driven in a large part by performance requirements. In environments where performance and uptime are not important, the need for systems management and monitoring are not important. A fair statement to make is: If you buy a blade-based computer, you are concerned about some kind of performance. One of the values of blade-based computing is its ease of management compared to traditional computer chassis. Open source technologies, such as those discussed in this chapter, can help you monitor and manage not only the performance of your systems, but also the health of those systems.

10.1 Simple Network Management Protocol (SNMP)

SNMP is a protocol used for network management. Red Hat Linux distributes the University of California at Davis SNMP agent and tools. The SNMP agents organize information about host in a hierarchical directory structure. The top levels of the directories are broad categories, such as “system” or “interfaces”. They have subentries that provide specific information. Each entry is called a *management information base* (MIB). For a complete tutorial on UCD SNMP, visit the project Web site at:

<http://net-snmp.sourceforge.net/>

The following section shows you how to configure the SNMP agent included in the Red Hat distribution. It provides valuable information about Linux hosts to your network and how to use the SNMP utilities to collect that information. It also explains how to use SNMP version 1 for read-only access to hosts. SNMP version 1 does not have a sophisticated authentication mechanism. Use IPChains, described in Chapter 9, “Security” on page 199, to filter the access to the port that the `snmpd` daemon binds to.

10.1.1 Configuring `snmpd`

To configure `snmpd`, follow these steps:

1. Confirm that the `ucd-snmp` package is installed. If not, mount the distribution and install the package using the `rpm` command:

```
rpm -i ucd-snmp-4.2.4-1.i386.rpm
```

The configuration file for the daemon is installed in `/etc/snmp/snmpd.conf`. The only *community*, like an account name, that is configured by default when the `rpm` package is first installed is *public*. The default view of the public community is restricted to just the system hierarchy that provides very limited information about the host.

2. Create a new community name that has unlimited read-only access to all the MIBs configured for the `snmp` agent. Add the following lines to the `/etc/snmp/snmpd.conf` file:

```
##      sec.name  source          community
com2sec mynetwork 10.0.0.0/24    bladepublic
##      group.name sec.model  sec.name
group MyROGroup  any          mynetwork
##      incl/excl subtree          mask
view all    included  .1          80
## -or just the mib2 tree-
view mib2   included  .iso.org.dod.internet.mgmt.mib-2 fc
##      context sec.model  sec.level prefix read  write notif
access MyROGroup ""      any          noauth  0    all  none  none
```

These lines create a community called `bladepublic` that can be accessed from the `10.0.0.0/24` network to read all the MIBs available from the `snmp` agent.

3. Edit the system contact information for the `snmp` agent by updating the `syslocation` and `syscontact` lines in the `snmp.conf` file:

```
syslocation "IBM, Beaverton, Oregon, USA"
syscontact "Peter Bogdanovic <pbogdanovic@us.ibm.com>"
```

4. Restart the `snmpd` agent with the command `/etc/init.d/snmpd restart`. Confirm that the `snmp` agent will restart by entering the following command:

```
chkconfig --add snmpd
```

10.1.2 Using snmp utilities

Choose a host to be a network management station that has a interface on the local private network. In our example configuration, this is the 10.0.0.0/24 network. Confirm that the `ucd-snmp-utils` package is installed. If it is not installed mount the distribution and use `rpm` command to install it:

```
rpm-i ucd-snmp-utils-4.2.4-1.i386.rpm
```

Use `snmpwalk` to list the MIBs, and their values, for a host with a configure snmp agent. To walk the MIBs of the host we configured, use the `bladepublic` community string.

```
snmpwalk -v1 blade7.bce.ibm.com bladepublic
```

You can limit the amount of the MIB directory that you browse by adding the object ID to the end of the `snmpwalk` command. If you only want the system information, add the system object ID:

```
snmpwalk -v1 blade7.bce.ibm.com bladepublic system
```

If you want to browse the disk partitions, pass the storage ID:

```
snmpwalk -v1 blade7.bce.ibm.com bladepublic host.hrStorage
```

In the following section and in 10.3, “Mon” on page 209, we cover the MRTG and Mon tools. They query the values of these object IDs to collect, present, and act on the data.

Another useful tool is `snmpnetstat`. It works like the local `netstat` command but on remote hosts. To look at the open connections on a remote host, enter this command:

```
snmpnetstat -v1 blade
```

The results look similar to the following code this if there were very few connections to the host.7.bce.ibm.com bladepublic:

```
Active Internet (tcp) Connections
Proto Local Address          Foreign Address          (state)
tcp    blade7.bce.ibm.com.ssh    blade9.bce.ibm.com.34922 ESTABLISHED
Active Internet (udp) Connections
Proto Local Address
udp    *.snmp
```

10.2 MRTG

The MRTG is a tool for collecting and displaying information on the traffic load of network links. MRTG uses SNMP to collect the data on the traffic and produces Web pages with graphics to display the data. Documentation for MRTG and sample displays are available on the MRTG Web site at:

<http://people.ee.ethz.ch/~oetiker/webtools/mrtg/>

Although BladeCenter blades running Linux are not routers, it is a good idea to measure how much traffic flows through each interface. It may help you spot routing problems, usage trends, and even security problems.

10.2.1 Installing MRTG

To install MRTG, follow these steps:

1. Confirm that the apache Web server is installed on the network monitor host. If you skipped the chapters (and other Redbooks) on installing and configuring apache, here is a very quick way to get Apache up and running. Mount a file system with the Red Hat distribution and install it with the **rpm** command:

```
rpm -i apache-1.3.27-2.i386.rpm
```

2. Start the Web server and confirm that it starts on reboot:

```
/etc/init.d/httpd start; chkconfig --add httpd
```

3. Download the RPM package version of MRTG available on the Web at:

<http://ftp.falsehope.com/home/gomez/mrtg/>

4. As of writing this redbook, the most recent version is mrtg-2.9.25-1.7.2.i386.rpm. Install the mrtg rpm file:

```
rpm -i mrtg-2.9.25-1.7.2.i386.rpm
```

5. Change directories to /etc/mrtg. Create a mrtg configuration file for each host you want to monitor using the **cfgmaker** command:

```
cfgmaker bladepublic@blade7.bce.ibm.com >blade7.cfg
```

6. Set up the snmp agent for the host prior to managing it with mrtg. Concatenate the blade7.cfg file to the mrtg.cfg file:

```
cat blade7.cfg >> mrtg.cfg
```

7. Edit the resulting mrtg.cfg file and uncomment the Options line:

```
Options[_]: growright, bits
```

8. Run the **mrtg** command:

```
/usr/bin/mrtg /etc/mrtg/mrtg.cfg
```

9. The rateup utility executed by the **mrtg** command may post a warning on the first time it is run about not being able to rotate files. Ignore the warning.

10. Run the **mrtg** command two or three more times:

```
/usr/bin/mrtg /etc/mrtg/mrtg.cfg; sleep 30; /usr/bin/mrtg /etc/mrtg/mrtg.cfg
```

11. Change directories to /var/www/html/mrtg. Display the log for one of the interfaces that is now being monitored by **mrtg** with the following command:

```
cat blade7.bce.ibm.com_2.log
```

You should see time stamps and four columns of zeros.

12. Generate and index page for the directory with the **indexmaker** command:

```
/usr/bin/indexmaker /etc/mrtg/mrtg.cfg > ./index.html
```

13. Point your Web browser at the mrtg directory of the network monitor host. In our case, this is:

```
http://blade9.bce.ibm.com/mrtg/
```

14. A page with two graphs should appear, one for the eth0 interface and the other for the eth1 interface.

15. Add mrtg to the roots crontab. Type **crontab -e** and then add the line.

```
*/* * * * * /usr/bin/mrtg /etc/mrtg/mrtg.cfg 2>/dev/null
```

That is it. You are done. MRTG is installed and ready to use.

To add more blades to the MRTG setup, go back to the where you executed the `cfgmaker` command above. Follow the steps with another blade through to the `index-maker` command.

10.3 Mon

Mon is a scheduler and alert management tool for monitoring the availability of numerous applications on a network. It is designed to be easily extensible so that you can monitor just about any application and send alerts by any mechanism that is available. Mon is really a scheduler that starts monitor scripts and triggers alert scripts. The scripts are generally written in Perl, but can be any type of application. They use the command line, standard in and standard out, to communicate with the Mon daemon.

There are a large number of monitor scripts that come as part of the Mon software distribution. These scripts check to see if a process is running and confirm where they are responding to queries. Sometimes daemon processes get into a tight loop or some other type of hung state. The other processes on the machine still works properly. If you are simply monitoring the machine to determine whether it responds to ICMP packets, it appear to be fine but the daemon that provides service is in fact broken. Using Mon, you can know when nearly any process isn't responding properly and send an alert.

The `mon.d` subdirectory of the Mon package contains 37 pre-built monitor scripts for monitoring dozens of services. There are more monitor scripts available on the Internet. Look at the "contrib" directory of the Mon Web site for additional monitor scripts, alert scripts, and other utilities:

<http://ftp.kernel.org/pub/software/admin/mon/contrib/index.html>

If you still can't find what you need, remember monitor scripts are simple to write. It's easy to produce your own.

The `alert.d` subdirectory has a collection of sample scripts for sending alerts to operators. The basic alert mechanisms of e-mail and QuickPage, a freeware paging software package are included in the package. You can find QuickPage on the Web at:

<http://www.qpage.org>

This section demonstrates how to install Mon and configure it to monitor IP stacks using the `fping` monitor and the Apache daemon using the HTTP monitor. It demonstrate a very basic Mon configuration that provides the foundation add monitored services.

10.3.1 Installing Mon

To install Mon, follow these steps:

1. On the blade that you chose for use for network monitoring, download the Mon package from the Web at:
<http://www.kernel.org/software/mon/>
2. At the time of writing this redbook, the current version is `mon-0.99.2`. Uncompress and expand the Mon tar ball in a temporary directory. We use the directory `/root/src`.
3. Mon is written entirely in Perl. This makes Mon more portable because there is nothing to compile. But it means you need a working Perl environment. If you installed the Advance Server package cluster of the Red Hat Advanced Server 2.1, as described in 3.2, "Installing operating system instances" on page 18, you have a large and sophisticated

Perl environment already. Unfortunately, Mon requires additional Perl modules that are not a part of the Red Hat Advanced Server distribution.

Use the CPAN module to download, compile and install the following Perl modules.

- MD5
- Net::DNS
- Time::Period
- Time::HiRes
- Convert::BER
- Mon::SNMP
- Authen::PAM

4. To use the CPAN Perl module, install the modules as the root user. Type the command:

```
perl -MCPAN -e shell
```

5. The first time you execute the command it ask a several questions. It is safe to select all the defaults. When it asks for ftp proxies or http proxies, enter the proxy host name if you use proxies for FTP or HTTP. The meta-data for the CPAN archive is downloaded to your system. You are asked to select CPAN mirrors to collect modules from. Try to choose mirrors that are geographically close to you. Eventually you see the CPAN shell prompt that looks like this:

```
cpan>
```

6. To install the MD5 Perl module, type the following command at the CPAN prompt:

```
install MD5
```

The module's source tar file is automatically downloaded, extracted, compiled, tested, and installed. If there are dependencies that are not already installed on your system, the CPAN application asks if you want to install the dependencies. If the CPAN tool asks you this question, always answer *yes*.

7. Repeat the previous step for each module. For example, type the command:

```
install Net::DNS
```

8. Type the following command:

```
install Time::Period
```

9. There is one other component that is used by a Mon, a monitor script that is useful but not included in the Mon package. It is called *fping*. It's an ICMP ping "work alike" that uses non-blocking I/O and supports more command line options than the usual *ping* command. Download *fping-2.2b1.tar.gz* from the Web to a directory (such as */root/src*):

<http://www.kernel.org/software/mon/>

10. Uncompress and expand *fping-2.2b1.tar.gz* with the command:

```
tar xzf fping-2.2b1.tar.gz
```

11. Change directories to *fping-2.2b1* and type the command:

```
./configure
```

12. Type the following command to copy the *fping* application to the */usr/local/sbin* directory:

```
make install
```

13. After you finish installing the additional Perl modules and build *fping*, create a directory to run Mon from. We chose to use the Mon default directory */usr/lib/mon*:

```
mkdir /usr/lib/mon
```

14. Change directories to */root/src/mon-0.99.2*. The directory you expanded the Mon package tar ball into earlier. Then copy the files to */usr/lib/mon*. You may use *tar* to copy files and

directories. This technique keeps the directory structure intact while preserving permissions. Use with the following command:

```
tar cf - . | (cd /usr/lib/mon; tar xf -)
```

Note: You can extend the old sys admin trick, the tar | tar technique, by using ssh to securely copying directories between systems. Many UNIX command shells execute subcommands (those commands listed in ()) on the command line within subshells. This means that there is a separate process created for each set of commands within parenthesis. For example to copy the current directory into the /tmp directory on fu.bar.com, you enter:

```
tar cf - . | ( ssh fu.bar.com "cd /tmp ; tar xvf -)
```

You can also take this technique to the next step and use it to copy directories between two third-party systems. That is you can use this technique to copy directories between two systems to which you are not currently logged on. For example, to copy /tmp/bob from fubu.com to /tmp/job on fu.bar.com, you enter:

```
(ssh fubu.com "cd /tmp/bob ; tar cf - ." ) | (ssh fu.bar.com "cd /tmp/job; tar xvf -")
```

The system you run this command on creates two subshells and pipes stdout from the first subshell to stdin of the second subshell. Each subshell executes ssh to their respective destination hosts. This creates studio links for you to run the command on.

The drawback of this is that all I/O goes through the system you are logged into. You can get your current system completely out of the loop by simply nesting the `ssh` commands. This is tricky and heavily depends on your shell configuration on all systems, as well as the ssh configuration. If everything is set up just right, you should be able to enter:

```
ssh fubu.com "cd /tmp/bob ; tar cf - ." | (ssh fu.bar.com "cd /tmp/job; tar xvf -") "
```

Getting this working take some maneuvering with shell quotations and ssh settings. However, it can come in handy if you get stuck at the end of a slow dial-up line.

If performance is a question, consider running each subshell in the background (with an ampersand (&) symbol after each closing parenthesis on the command line). Or play with the `-f` option of ssh.

10.3.2 Configuring Mon

Configuring Mon means changing the mon.cf file. Follow these steps:

1. To use the example.cf file, change directories to /usr/lib/mon/etc.
2. Copy the example.cf file to mon.cf.
3. Open the mon.cf in a file editor.
4. Change the authtype to pam. pam isn't listed as an option in the comments of the file but it works.

```
authtype = pam
```

5. Remove the host group definitions from the example.
6. Create a wwwservers group that corresponds to the Ethernet interfaces that provide services to users on the blades running Web servers. In our example, it's the 192.168.2.0/24 network.

```
hostgroup wwwservers 192.168.2.159 192.168.2.160 192.168.2.161
```

You can use the host names instead of the IP address. If you do, make sure the names are in the `/etc/hosts` file on the network monitoring machine. You don't want Mon to stop working if DNS fails.

7. In the watch section of the `mon.cf` file, delete all the "watch" stanzas except for the watch `wwwservers`.
8. In the service ping and the service http sections, modify the alert line to name the alert script and argument you want to use. In our case we are using e-mail alerts so the line is changed to:

```
alert mail.alert blade_admin@us.ibm.com
```

Note: It is a good idea to send alerts to mail aliases that point to all the appropriate administrators.

9. Delete the service Telnet section. Telnet was intentionally disabled on the hosts.
10. Delete the rest of the lines to the bottom of the file.
11. Save the file.
12. Open the `auth.cf` file. The default requires authentication by a valid user on the host to do anything more than list Mon data. If you wanted to further limit access to particular users on the network monitoring host you should edit this file.
13. Change directories to `/usr/lib/mon`. Start the mon daemon with the command:

```
./mon -f -c etc/mon.cf
```

14. Confirm that Mon is running and receiving connections by changing to the clients directory and issuing the command:

```
./moncmd -l localhost list pids
```

15. This should print out the Mon daemon's process ID.
16. The `moncmd` utility allows you to send a whole range of commands to the running Mon daemon. You can use `moncmd` to turn the monitors on and off, pause or restart the Mon daemon or even terminate the Mon daemon. You should read the man page in the `doc` directory for a full explanation of `moncmd`.

```
nroff -man doc/moncmd.1 |less
```

17. Enter the `monshow` command to view the current alerts.

This gets you started with Mon. From here you can go further with Mon. We encourage you to expand the number and types of monitor scripts you have scheduled and the host groups you want to watch.

In addition, IBM Director comes with every BladeCenter. IBM Director provides monitoring and management of Linux systems and Microsoft Windows clients and servers. Although it is not open source, it is a logical alternative to MON and MRTG if those tools don't meet your needs.

10.4 Eclipse

Over the course of this redbook, we've touched several times on issues of application development. Most of our examples have used a simple text editor. Of course, for large projects, this simply won't do. This begs the question: Is there an open-source tool that we can use for large projects? A graphical interactive development environment that has all the right tools? Fortunately, the answer is "Yes." That tool is Eclipse.

Eclipse is an open source software development project dedicated to providing a full-featured commercial-quality platform for the development of highly integrated tools. It is composed of three projects, the Eclipse Project, the Eclipse Tools Project, and the EclipseTechnology Project, each of which is overseen by a Project Management Committee (PMC) and governed by its Project Charter.

Isn't Eclipse just a simple integrated development environment (IDE)? While it can be used as a Java IDE in its most basic form, it can actually be extended to support other languages and other tools.

This section takes you through the basics of setting up Eclipse and using it for some simple examples.

10.4.1 Getting started with Eclipse

Installing and running Eclipse is straightforward. However, there are a couple of configuration adjustments that first-time users might have to get around. We discuss these in this section.

Hardware and operating system prerequisites

Most likely you'll be performing your development work on a desktop or workstation, not on BladeCenter itself. Eclipse works in a graphical environment and can have fairly hefty system requirements. Working on an Intel platform, at the minimum, you should have a Pentium III class machine or better with 256 MB RAM or more.

The 1.0 Release build of the Eclipse Platform was designed to run on Windows XP, Windows 2000, Windows 98, Windows ME, and Red Hat Linux Version 7.1 (x86/Motif).

The most recent 2.0 builds of the Eclipse Platform are designed to run on Windows XP, Windows 2000, Windows 98, Windows ME, Red Hat Linux Version 7.1 (x86/Motif and x86/GTK), SuSE Linux 7.1 (x86/Motif and x86/GTK), and Solaris 8 (SPARC/Motif).

Software prerequisites

You will need to install the Java Developer Kit (JDK). Eclipse needs JDK 1.3 or higher, although JDK 1.4.1 is recommended. Eclipse does not come with a JDK, so you will either use the one that came with your Linux distribution or download a copy from the following Web sites:

- ▶ For Windows: <http://www7b.boulder.ibm.com/wsd/wspvtdevkit-info.html>
- ▶ For Linux: <http://www-106.ibm.com/developerworks/java/jdk/linux130/?dwzone=java>

You also need to get Eclipse itself. You should download either the latest release build or the most recent stable development build. These builds are available on the Web at:

<http://eclipse.org/downloads/index.html>

Installing Eclipse for Windows

Installing Eclipse is straightforward in Windows:

1. Install the JDK.
2. Extract the Eclipse archive on your workstation. This creates a directory called *eclipse*. For Windows, a possible location is `c:\eclipse`.
3. Start Eclipse. In Windows, run `eclipse.exe` from the directory you installed it in. The very first time you run Eclipse, it performs some final installation steps. Then the Eclipse splash display appears. Thereafter, only the splash display appears, before going to the Eclipse workbench.

Installing Eclipse for Linux

Eclipse is available for Linux in two ways: one uses Motif and the other uses GTK. The GTK version is more aesthetically pleasing than Motif, so that is what we have chosen to work with.

The setup process is similar to Windows, with a couple of caveats:

1. Install the JDK.
2. Extract the Eclipse archive. This creates a directory called *eclipse*.
3. To start eclipse, from within the Eclipse subdirectory, type:

```
./eclipse
```

If you're using Red Hat 7.3 or later and the most recent version of Eclipse for Linux, there should not be any problems. However, if you're using Red Hat 7.2, you may run into a problem concerning libgtk. This is because the latest version of Eclipse as of this writing requires GTK2. To work around this, install the latest GTK2 files for Red Hat on your system.

Anatomy of the Eclipse workbench

Figure 10-1 shows the main view of the Eclipse workbench.

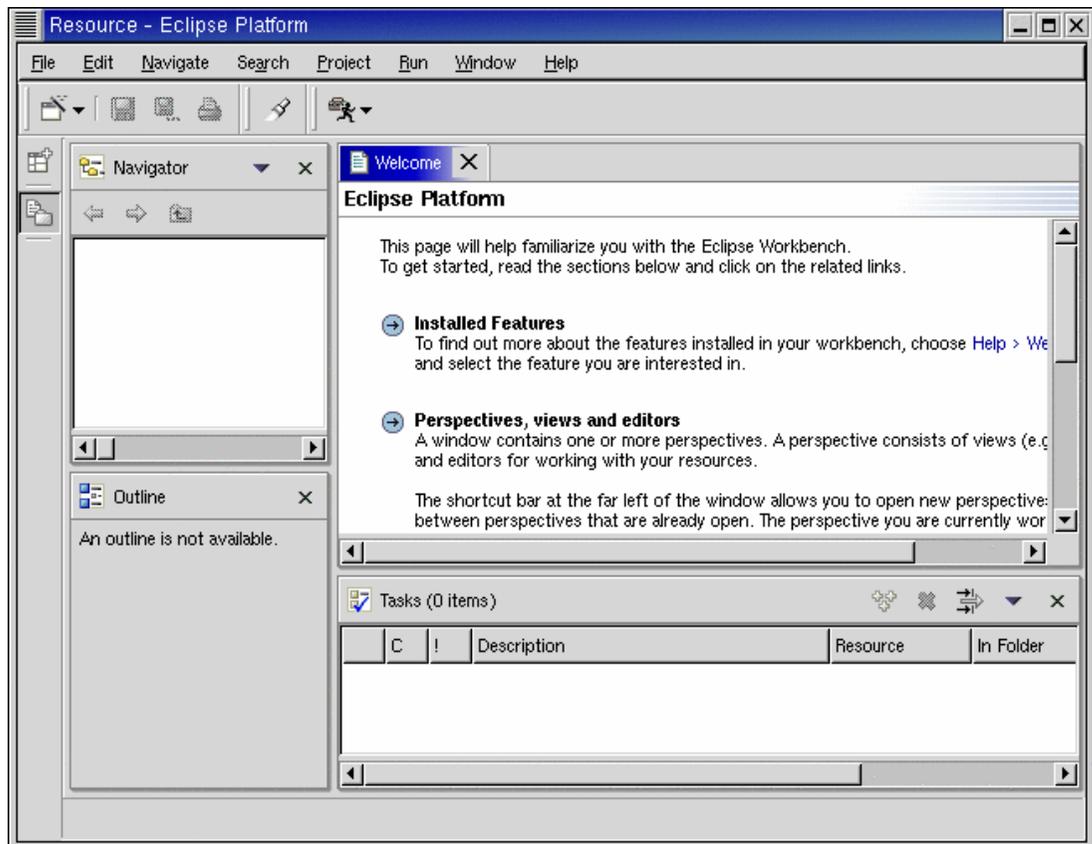


Figure 10-1 The main view of the Eclipse workbench

Eclipse's main window is broken down into smaller windows called *Views*. Views display different information about your work. The Views shown in our workbench are:

- ▶ Navigator
- ▶ Outline
- ▶ Tasks
- ▶ Welcome Note

When you're doing your actual work, *Editors* also come into play. As the name implies, the Editors are small windows where you actually edit your code.

A *Perspective* in Eclipse is a configuration of Views and Editors. It contributes to the actual appearance of the menus and the toolbar. The chosen perspective determines the actual appearance of the workbench.

The vertical toolbar located at the left-hand side of the window allows you to switch between perspectives. The current perspective is displayed on the title bar. Resource is the current perspective in our example.

If you click the icon with the small plus on the vertical toolbar, it shows you the currently configured perspectives (Figure 10-2).

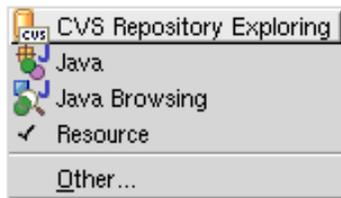


Figure 10-2 Currently configured perspectives

Selecting any other perspective changes the views of the main workbench, according to what is appropriate. For example, the CVS Repository perspective looks like the example in Figure 10-3.

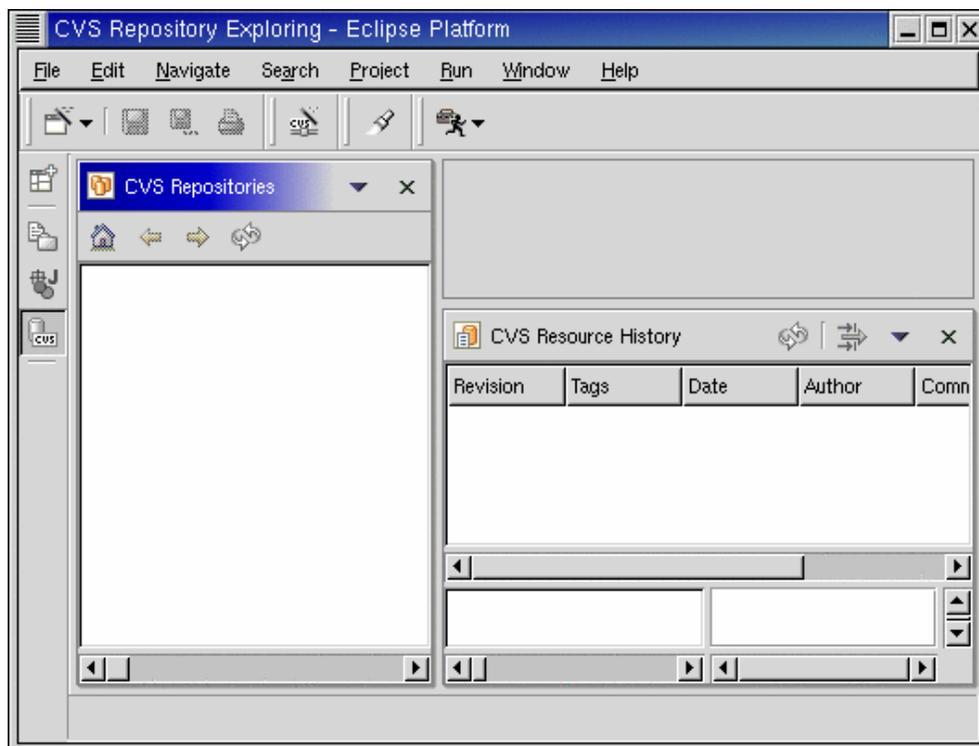


Figure 10-3 CVS perspective

Note that the CVS perspective shows views that pertain to the CVS repositories and CVS resource histories.

The Java Perspective, on the other hand, looks like the example in Figure 10-4.

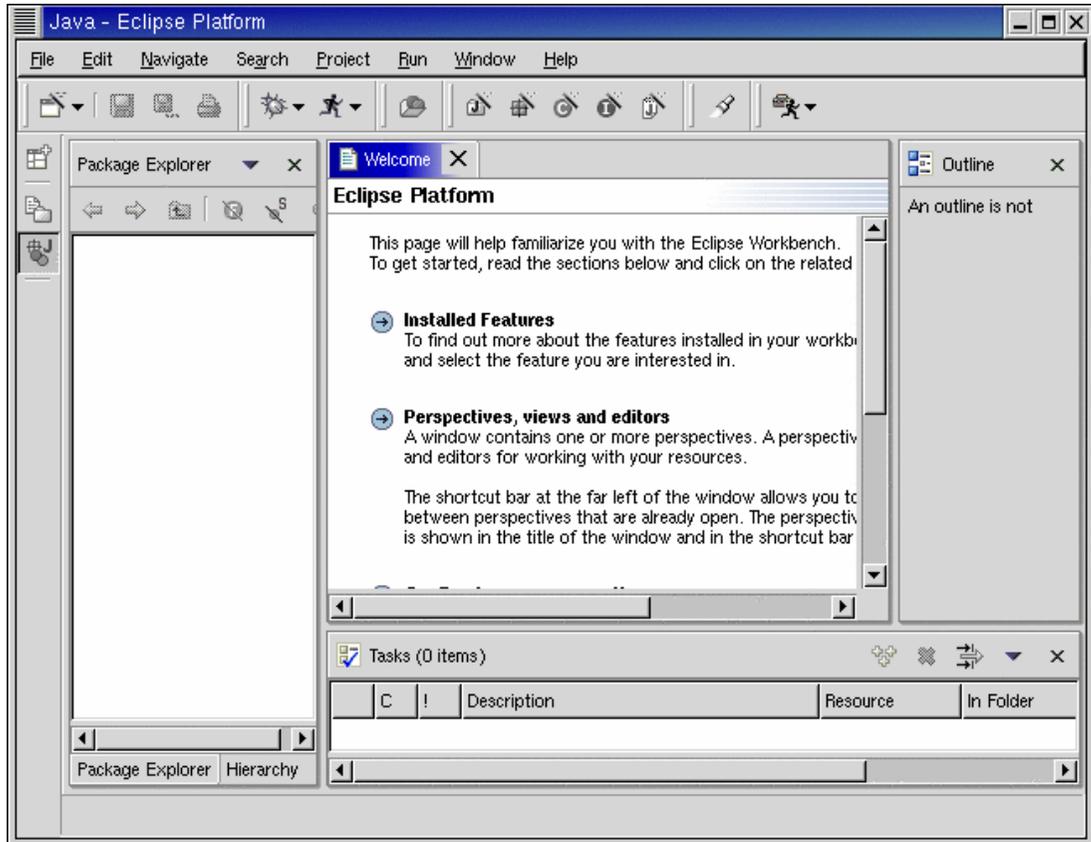


Figure 10-4 Java perspective

The Java perspective shows the Package Explorer view.

To add a view, click **Window-> Show View**. You see a list of available views as shown in Figure 10-5.

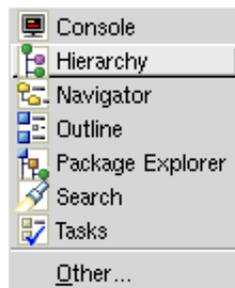


Figure 10-5 Views

Select the view you want, and it appears in the workbench. Within the Eclipse workbench, you can rearrange your views to better suit your working preferences. Simply drag and drop the views to where you want them placed. Figure 10-6 shows an example of rearranged views.

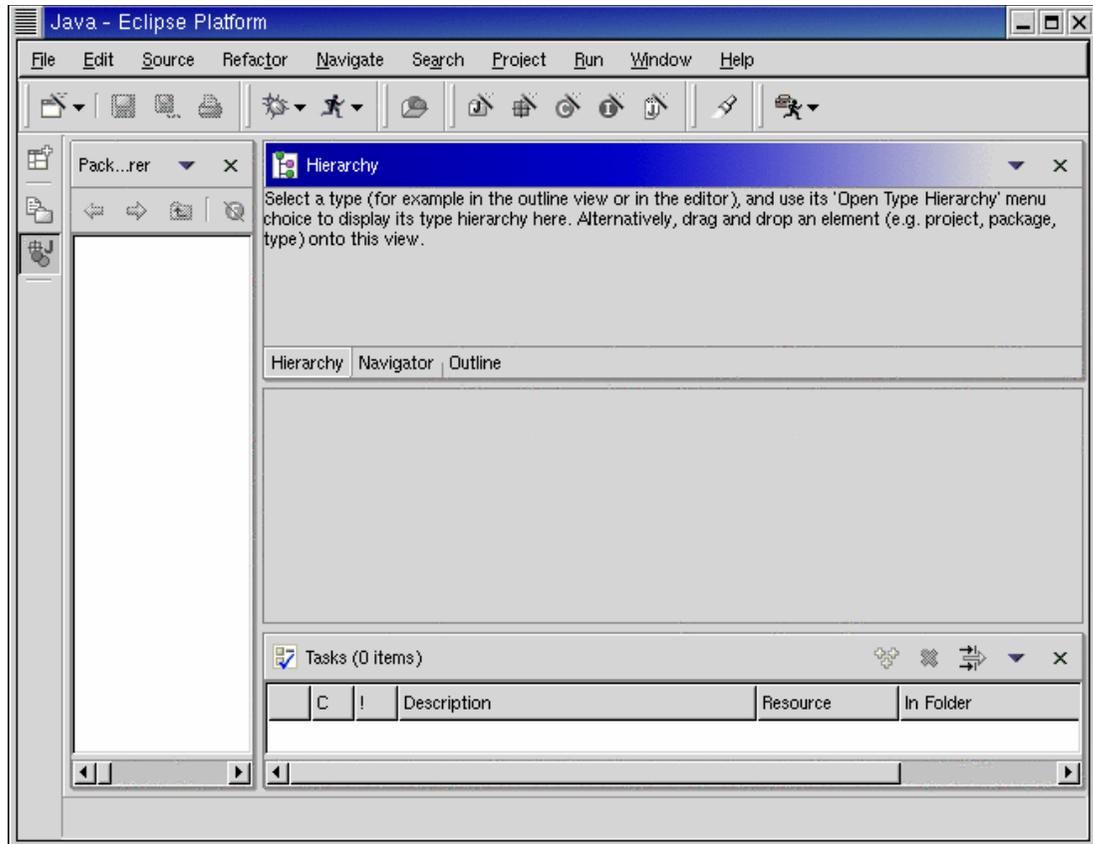


Figure 10-6 Rearranged views

10.4.2 Working with Eclipse

Eclipse is a feature-rich tool, and very flexible, too. As such, it's easy to get lost in its myriad capabilities. The best way to learn to use Eclipse is through some simple examples.

Hello, world! in Java

Although Eclipse is an multipurpose development environment, it has a certain affinity for Java. We take you through the “Hello, world!” program in the Eclipse environment.

Creating a new project

Eclipse allows you to organize your work into projects. To start with a new Java project for our “Hello, world” program, follow these steps:

1. Click **File-> New-> Project**. A New Project wizard opens (Figure 10-7).

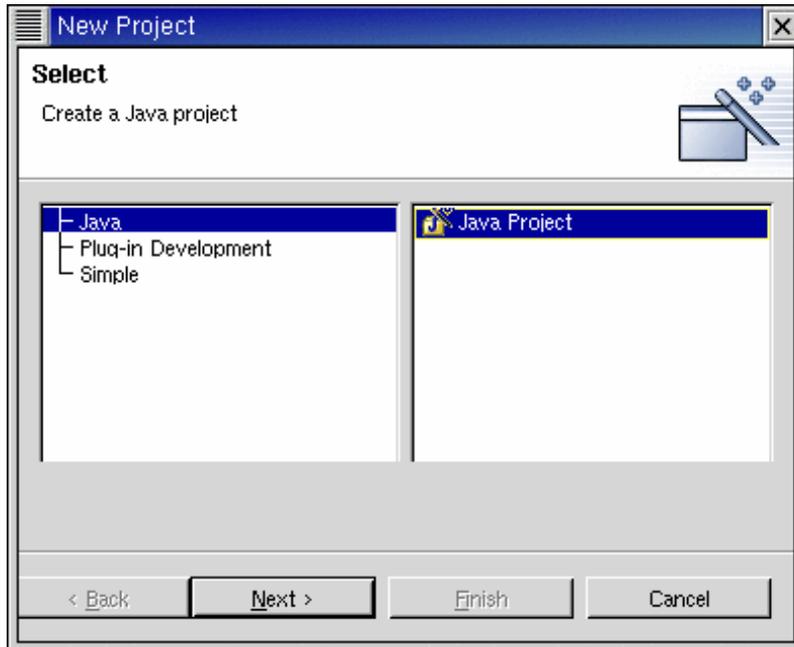


Figure 10-7 New Project wizard

2. Select **Java** and **Java Project**, and click **Next**.
3. On the next window (Figure 10-8), type Hello World Project for the Project name field. Fill in this information. It also prompts you for the location of the project. You can use the default or select a new location.

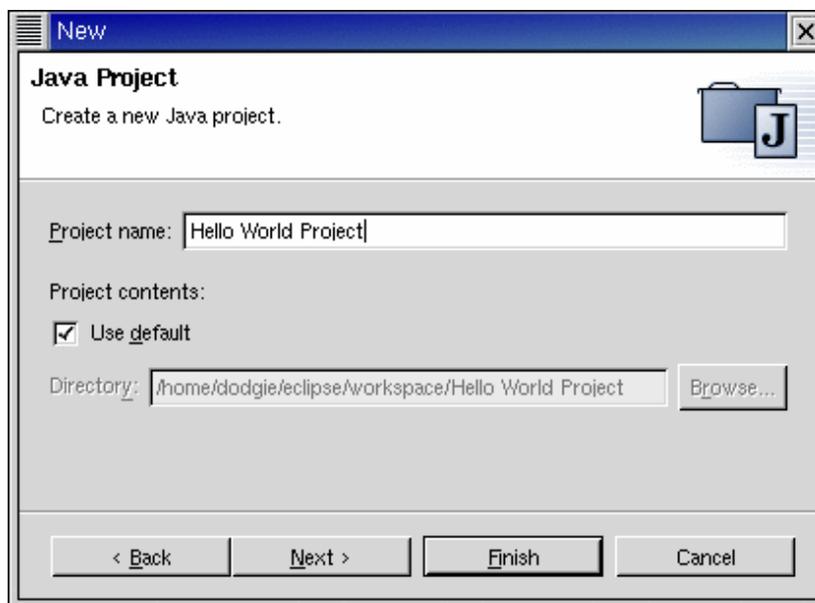


Figure 10-8 New Java Project

4. Click **Finish**. Eclipse automatically creates the working directory for the project. It contains the files `.project` and `.classpath`, which contain bookkeeping information about the project.

If you haven't yet opened the Java Perspective, Eclipse automatically opens it for you.

Some of the views now have content. For example, the Navigator view (Figure 10-9) displays the physical file information of the project directory. For now, `.project` and `.classpath` are the only files in the directory. The Package Explorer view displays the logical component information.

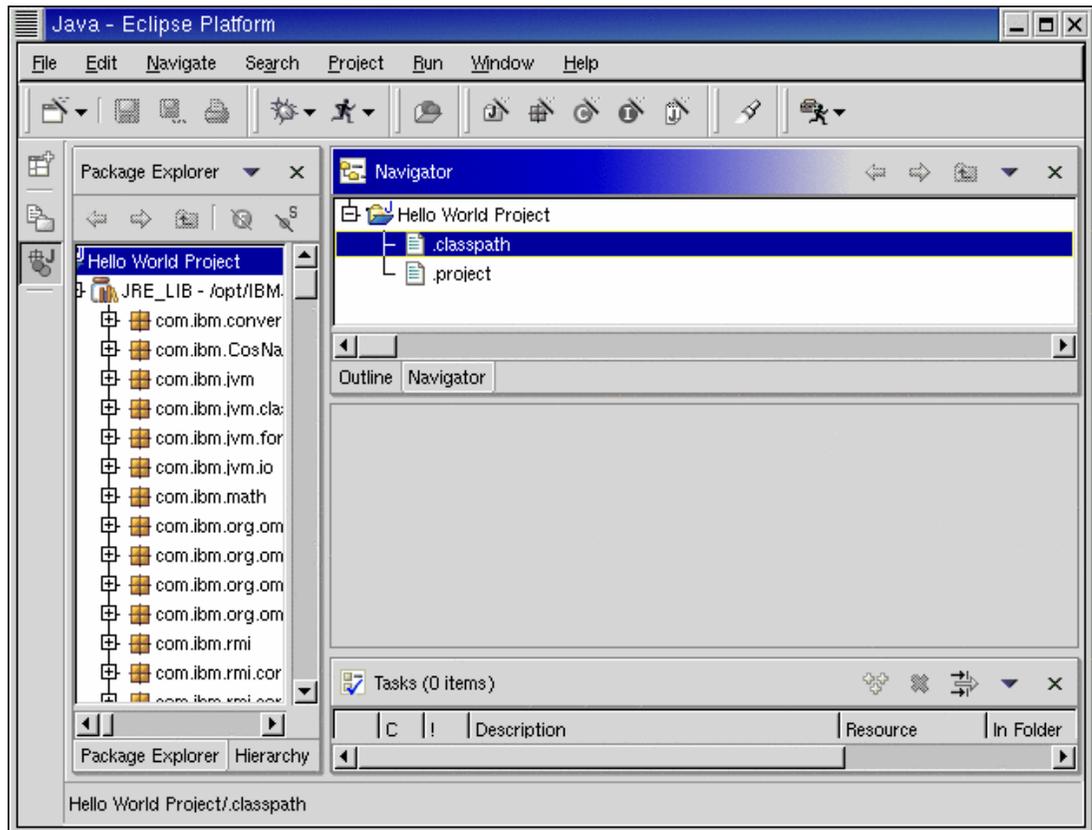


Figure 10-9 Skeletal framework for our Java application

Creating a package

In Java, a package is a group of classes that are related in some way. We create our “Hello World” application within a package. To create a new package, follow these steps:

1. Click **File-> New-> Package**. This opens another wizard (Figure 10-10).
2. Type helloWorldPackage in the field for the name of the package.



Figure 10-10 New Java Package

3. Click **Finish** to create the package.

Eclipse creates a new package called helloWorldPackage. The Navigator and Package Explorer views are updated.

Creating the class

Now let's create the main class for our program.

1. Click **File-> New-> Class**. This opens the class wizard.
2. Type the name of the class `HelloWorldClass` in the Name field.
3. For Modifiers, select **public**.
4. For Superclass, keep it at **java.lang.Object**.
5. Keep Interfaces empty.
6. Select the option to create the stub **public static void main(String[] args)**.

These steps are illustrated in Figure 10-11.

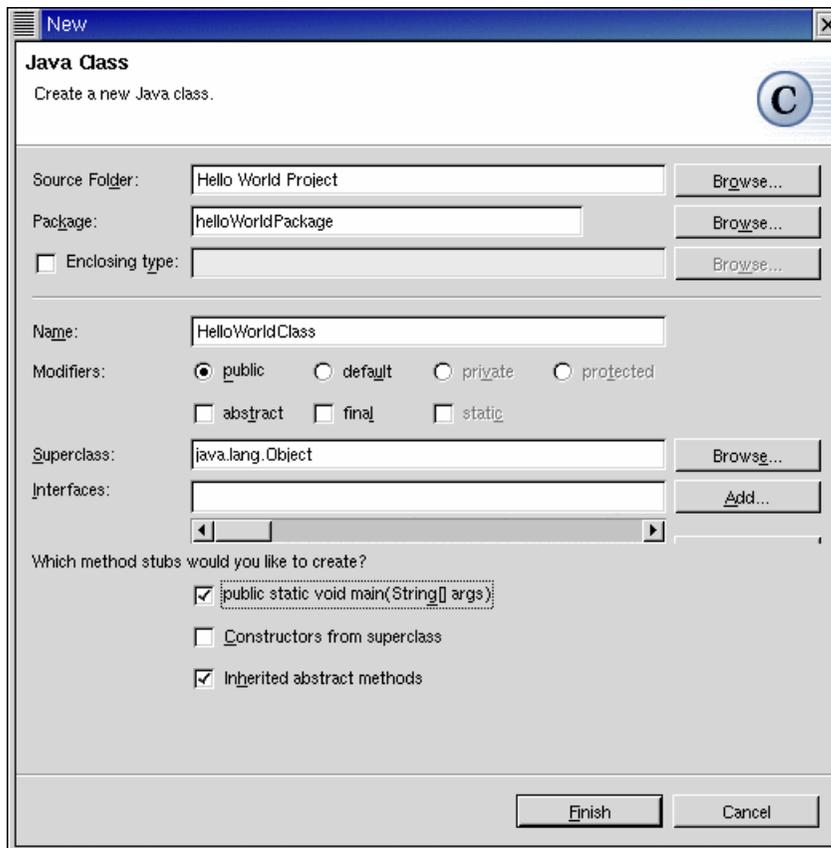


Figure 10-11 New Java Class

7. Click **Finish**. Eclipse creates the class `HelloWorldClass.java` and automatically compiles `HelloWorldClass.class`.

Your views should reflect the changes in both the Package Explorer and the Navigator.

Eclipse also automatically opens the source code of `HelloWorldClass.java` in an editor, showing the skeleton stub for the program. See Figure 10-12.

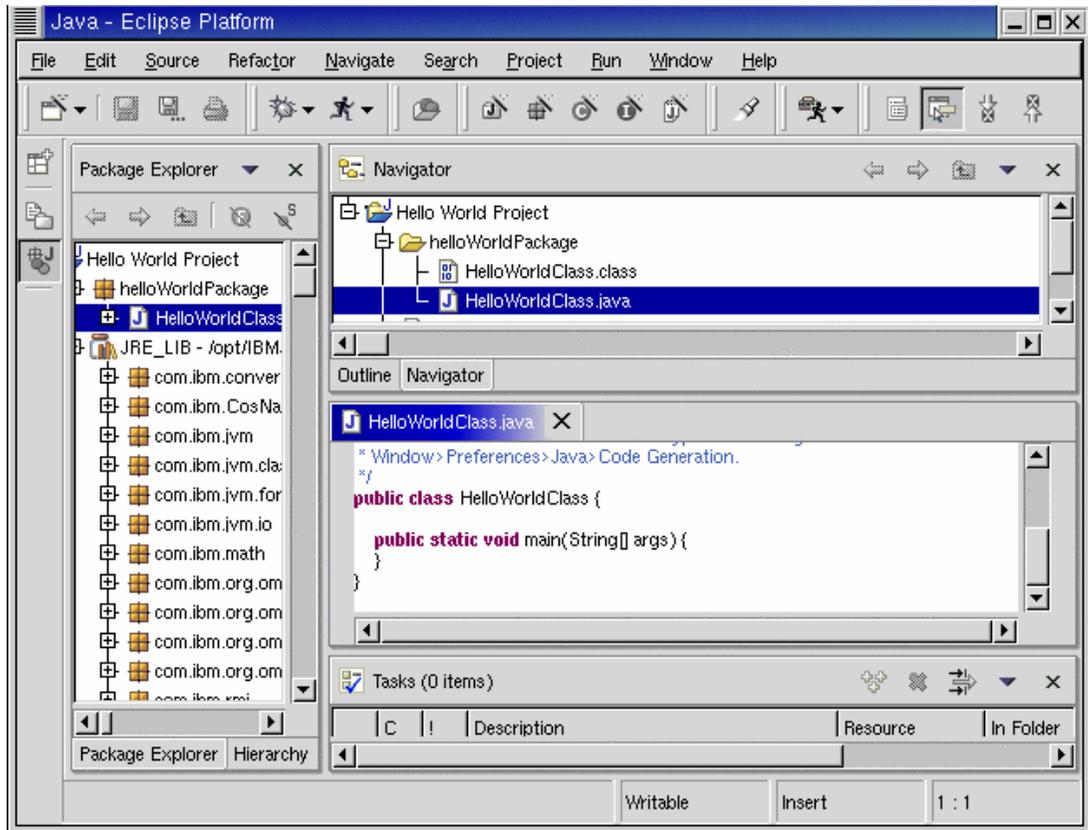


Figure 10-12 Our first class

8. Modify the code stub so that the main code looks like this:

```
public class HelloWorldClass {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

9. Save the file by clicking **File-> Save** or **File-> Save All**. This saves the contents of the file and recompiles the source.

Running the application

To run the application from within Eclipse, click **Run-> Run As-> Java Application**. Eclipse executes the program and shows you the output in a console within the Workbench (Figure 10-13).

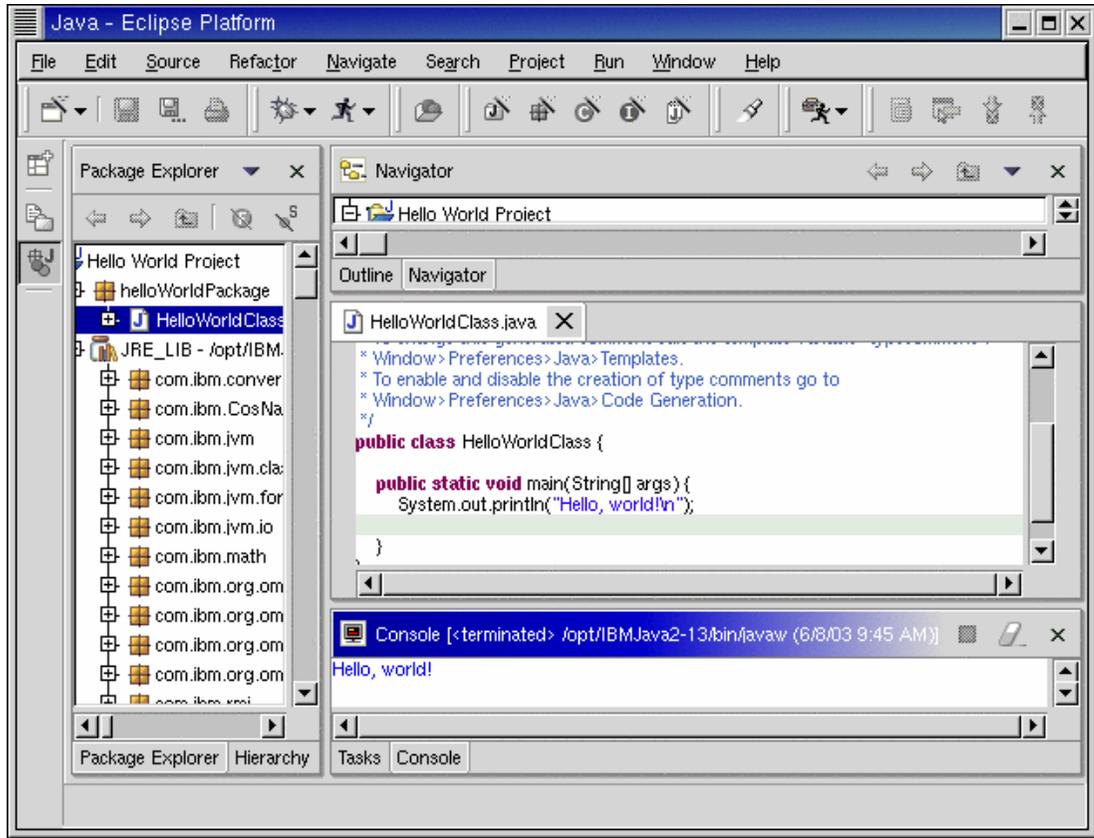


Figure 10-13 Output of our execution

Congratulations! You've run your very first (and very traditional) Java program written within Eclipse.

10.4.3 Tomcat plug-in for Eclipse

As discussed earlier, Eclipse is an extensible development environment. This makes it possible for developers to write plug-ins to give the core Eclipse product new capabilities.

For example, the base Eclipse package does not include capabilities to develop for the Tomcat environment. Sysdeo, a Java application development company, has written a Tomcat plug-in that addresses this need.

The plug-in gives Eclipse the following capabilities:

- ▶ Starting and stopping Tomcat
- ▶ Registering a Tomcat process to the Eclipse debugger
- ▶ Creating or importing a WAR project
- ▶ Adding Java projects to the Tomcat classpath
- ▶ Exporting a Tomcat project to a WAR file

Let's see how to install this plug-in for Tomcat.

Software prerequisites

To set up an Eclipse-based Tomcat development environment, you need the following packages (in addition to what you've already set up):

- ▶ Tomcat
- ▶ Eclipse Tomcat plug-in from the Web at:
<http://www.sysdeo.com/eclipse/tomcatPlugin.html>

Install Tomcat as explained in 7.2, "Tomcat" on page 141.

Installing the Tomcat plug-in

Unzip the Tomcat plug-in archive (tomcatPluginV21.zip) into the plugins subdirectory of Eclipse.

Open Eclipse and click **Window-> Customize Perspective**. This opens the Customize Perspective window (Figure 10-14). In the Available Items tree, click **Other** and select **Tomcat**.

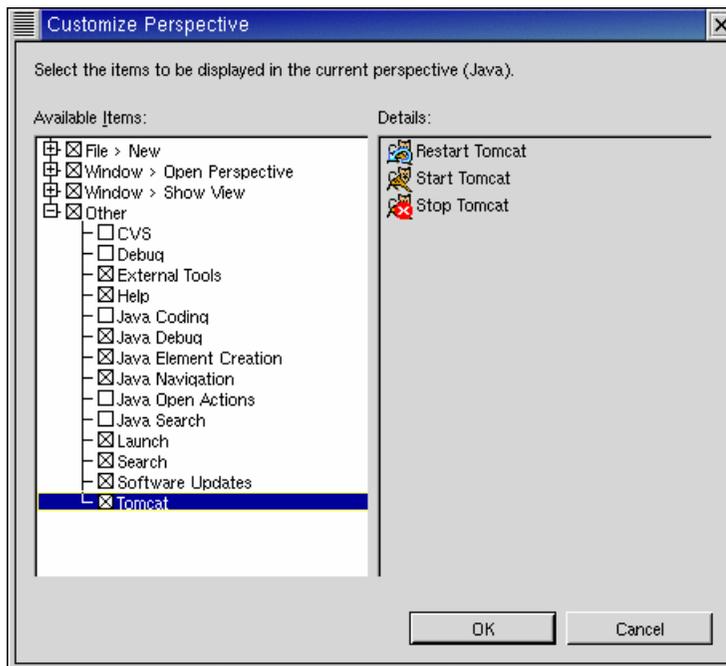


Figure 10-14 Customize Perspective

Configuring Tomcat

Click **Windows-> Preferences**. Then select **Tomcat**. This opens the Tomcat Preferences window (Figure 10-15). Set it up to match the settings of your Tomcat server.

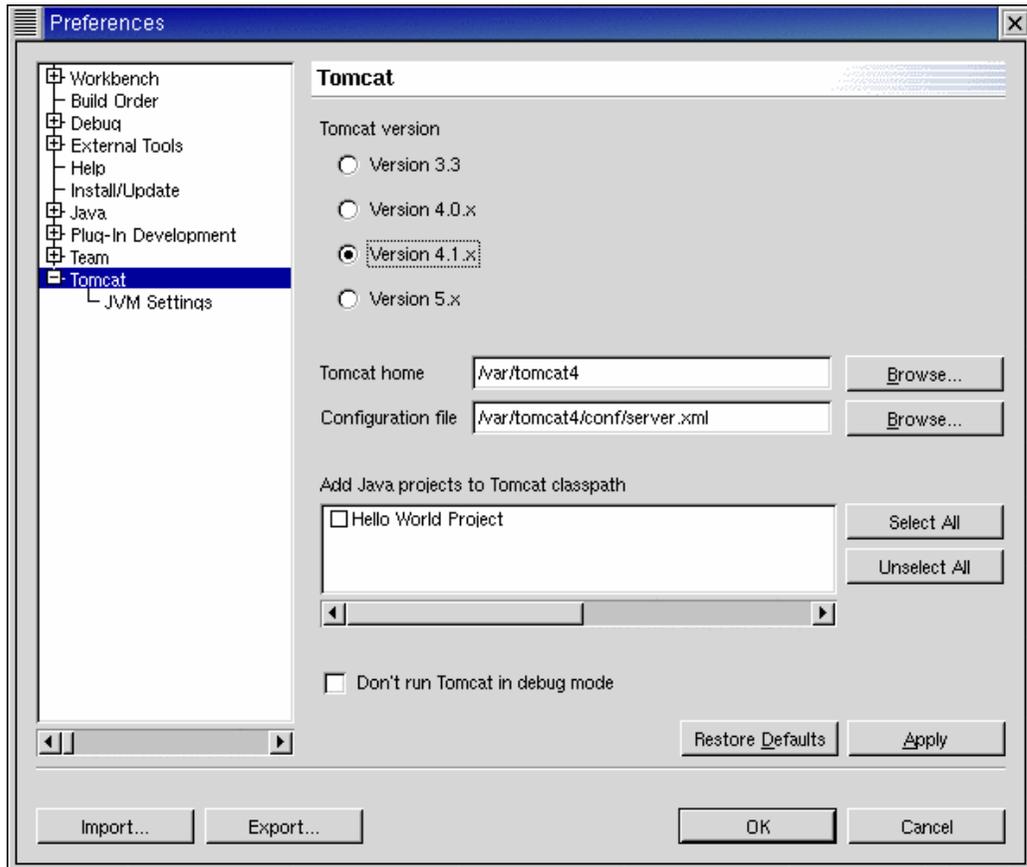


Figure 10-15 Tomcat preferences

Tomcat projects

From here on, you can create Tomcat projects using a wizard:

1. Click **File-> New-> Project** to open the New Project window (Figure 10-16).

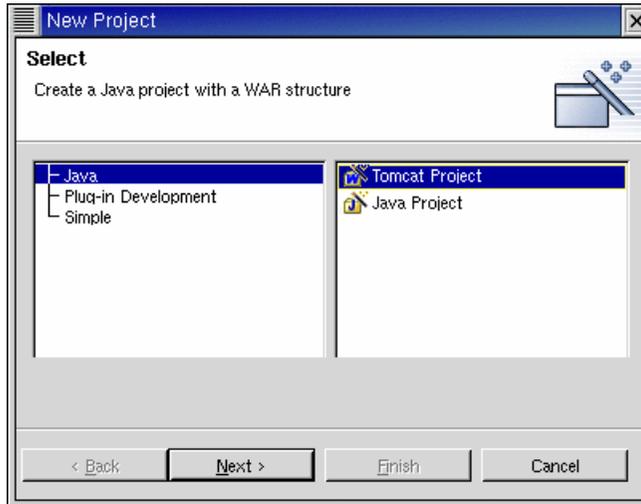


Figure 10-16 New Project with Tomcat

2. Select **Tomcat Project** and click **Next**.

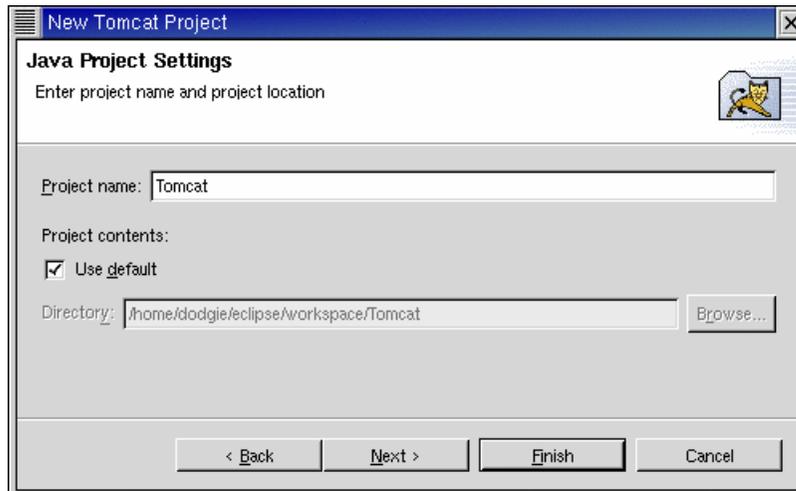


Figure 10-17 Setting the context

- Fill in the fields for the Context name and click **Finish**. In the Java workbench, you should see the Tomcat elements in your Package Explorer and Navigator views (Figure 10-18).

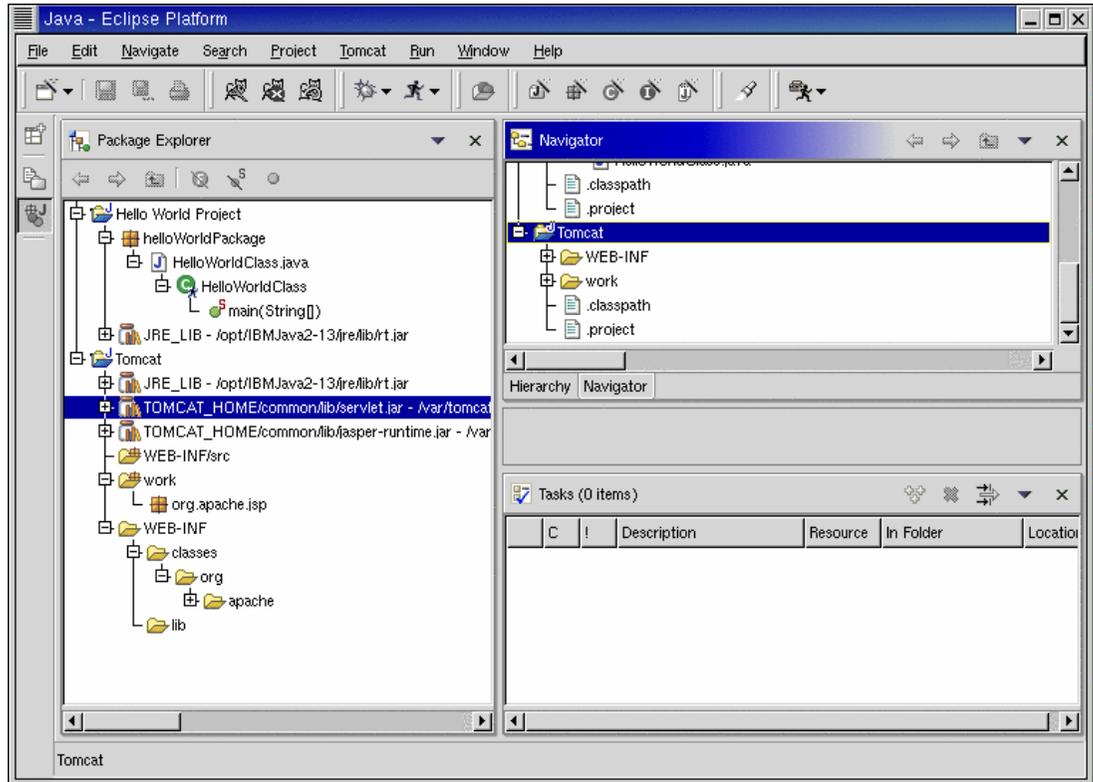


Figure 10-18 Tomcat projects

To create a new class for Tomcat, click **File-> New-> Class**. The parameters for Tomcat class files are similar to the example in Figure 10-19.

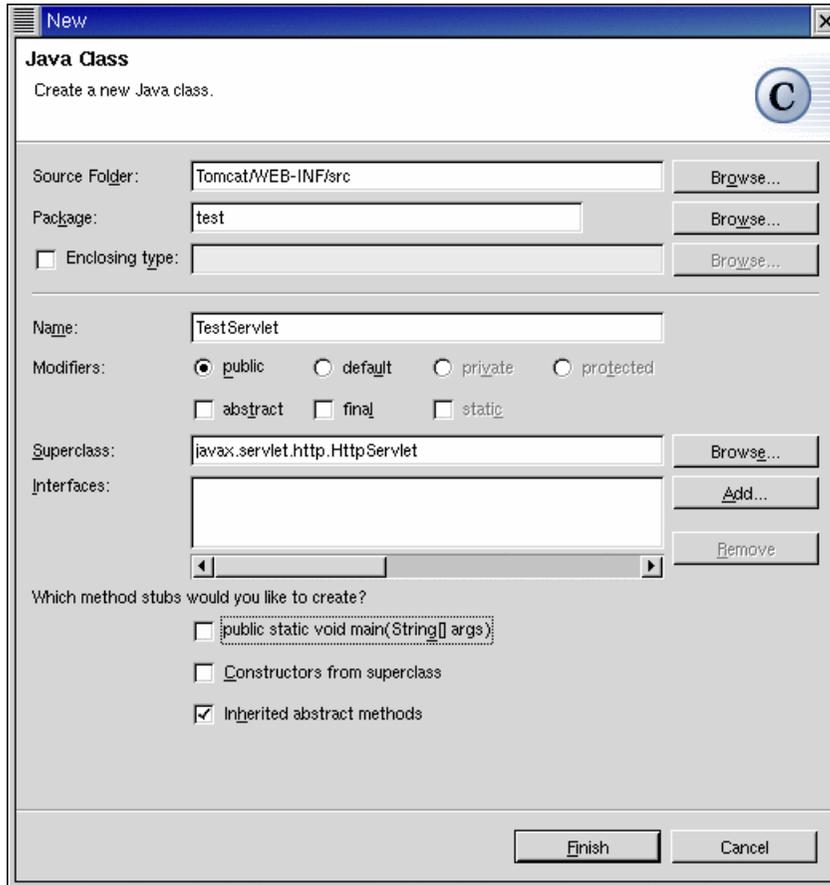


Figure 10-19 Tomcat class file

Click **Finish**. Eclipse creates the stub class for your new servlet class. Edit the file to match your requirements. See Figure 10-20.

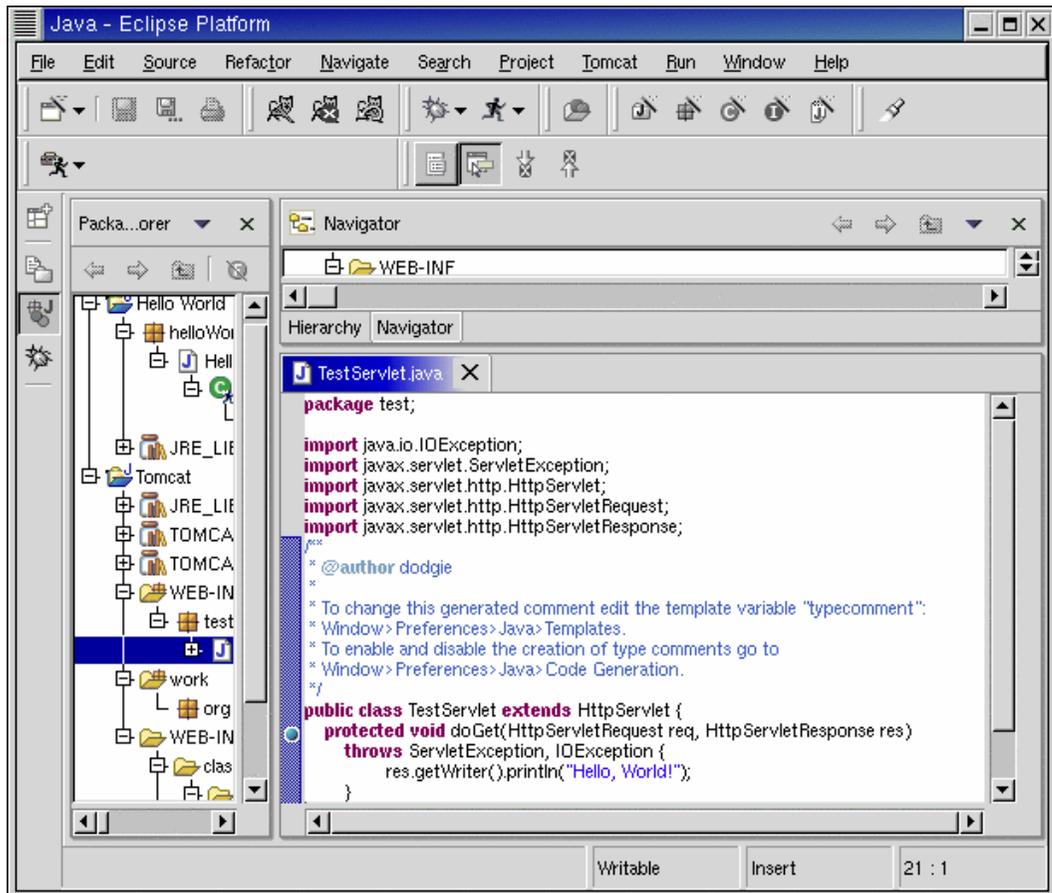


Figure 10-20 Servlet stub

Updating the context in server.xml

Right-click the Tomcat project to update the Context information in Tomcat's server.xml. Select Tomcat Project. This opens up the submenu shown in Figure 10-21.

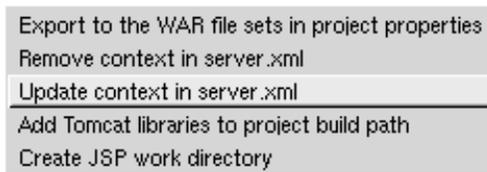


Figure 10-21 Updating the context in server.xml

For this to work, make sure you have write permissions to the server.xml file.

Starting and stopping Tomcat from Eclipse

Installing the plug-in modifies the main menu of Eclipse. Click the **Tomcat** menu option to open the submenu shown in Figure 10-22.

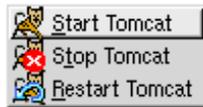


Figure 10-22 Tomcat controls

10.4.4 For more information

Your main source for Eclipse information is on the Web at:

<http://www.eclipse.org>

The site contains the latest downloads and bug fixes for Eclipse. It also has good articles on Eclipse development and support newsgroups that can help you work around any problems you might encounter.

A good starting tutorial on Eclipse, on which part of this chapter was based, is located on the Web at:

<http://www.ugrad.cs.ubc.ca/~cs410/lectures/resources/EclipseIntroductionLab/>

You can find another good tutorial on Eclipse on the Web at:

http://www.3plus4software.de/eclipse/tomcat1_en.html

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 234. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *The Cutting Edge: IBM @server BladeCenter*, REDP3581
- ▶ *Deploying Apache on IBM @server BladeCenter*, REDP3588
- ▶ *Deploying Samba on IBM @server BladeCenter*, REDP3595
- ▶ *Implementing Linux with IBM Disk Storage*, SG24-6261
- ▶ *Linux Application Development Using WebSphere Studio 5*, SG24-6431
- ▶ *Linux Handbook: A Guide to IBM Linux Solutions and Resources*, SG24-7000

Other publications

These publications are also relevant as further information sources:

- ▶ Adams, D.J. *Programming Jabber: Extending XML Messaging*. O'Reilly & Associates, January 1, 2002. ISBN 0-596-00202-5.
- ▶ Alexander, Christopher; Ishikawa, Sara; Silverstein, Murray. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, 1977. ISBN 0-195-01919-9.
- ▶ Collier-Brown, David; Eckstein, Robert; Kelly, Peter. *Using Samba*. O'Reilly & Associates, November 1999. ISBN 1-56592-449-5.
- ▶ Gamma, Eric; Helm, Richard; Johnson, Ralph; Vlissides, John. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Pub Co, January 15, 1995. ISBN0-201-63361-2.
- ▶ Hall, Marty. *Core Servlets and JavaServer Pages*. Prentice Hall PTR, May 26, 2000. ISBN 0-130-89340-4.
- ▶ Hall, Marty. *More Servlets and JavaServer Pages*. Pearson Higher Education, December 26, 2001. ISBN 0-130-67614-4.

Online resources

These Web sites and URLs are also relevant as further information sources:

- ▶ Internet Software Consortium
<http://www.isc.org/>
- ▶ Apache, Tomcat, and all the other Apache Software Foundation projects
<http://www.apache.org>

- ▶ Information on the Red Hat Linux distribution
<http://www.redhat.com>
- ▶ Linux kernel source
<http://www.kernel.org>
- ▶ Sourceforge
<http://sourceforge.net>
- ▶ The document “Filtering Malware and Spam with Postfix”
<http://advosys.ca/papers/printable/postfix-filtering.html>
- ▶ Postfix
<http://www.postfix.org>
- ▶ SpamAssassin
<http://spamassassin.org>
- ▶ Anomy Sanitizer
<http://mailtools.anomy.net/>
- ▶ Amavis
<http://www.amavis.org>
- ▶ Inflex
<http://pldaniels.com/inflex/>
- ▶ qmail scanner
<http://qmail-scanner.sourceforge.net/>
- ▶ MIMEDefang
<http://www.roaringpenguin.com/mimedefang>
- ▶ “Jabber Server Farming HOW-TO” by Ryan Eatmon
<http://www.tldp.org/HOWTO/Jabber-Server-Farming-HOWTO/index.html>
- ▶ AIM-Transport
<http://aim-transport.jabberstudio.org/>
- ▶ JIT, the Jabber ICQ Transport
<http://jit.jabberstudio.org/>
- ▶ MSN-Transport
<http://msn-transport.jabberstudio.org/>
- ▶ Yahoo-Transport
<http://yahoo-transport.jabberstudio.org/>
- ▶ “Java Web Applications”
<http://www.onjava.com/pub/a/onjava/2001/03/15/tomcat.html>
- ▶ “Deploying Web Applications to Tomcat”
<http://www.onjava.com/pub/a/onjava/2001/04/19/tomcat.html>
- ▶ “Creating a Web Application with Ant and Tomcat4”
<http://www.onjava.com/pub/a/onjava/2003/01/08/tomcat4.html>
- ▶ Tomcat Server Configuration Reference
<http://jakarta.apache.org/tomcat/tomcat-4.1-doc/index.html>

- ▶ “Demystifying Tomcat's server.xml file”
<http://www.onjava.com/pub/a/onjava/2002/07/31/tomcat.html>
- ▶ “Configuring Apache and Tomcat 4 with JK 1.2” by James Goodwill
<http://www.onjava.com/pub/a/onjava/2002/11/20/tomcat.html>
- ▶ “Integrating Apache and Tomcat4 for UNIX” from the Galatea Flashguides
<http://www.galatea.com/flashguides/apache-tomcat-4-unix.xml>
- ▶ “Enabling SSL in Tomcat/Apache in UNIX” from Galatea Flashguides
<http://www.galatea.com/flashguides/apache-tomcat-ssl-unix.xml>
- ▶ “Tomcat 4 Clustering HOWTO” by Filip Hanik
<http://cvs.apache.org/~fhanik/index.html>
- ▶ “In-session Memory Replication” by Filip Hanik
<http://www.filip.net/tomcat/tomcat-javagroups.html>
- ▶ “Apache 1.3.23 + Tomcat 4.0.2 + Load Balancing” by Pascal Forget
<http://www.ubeans.com/tomcat/>
- ▶ A Featurewise Comparison of PostgreSQL and MySQL
http://www.mysql.com/doc/M/y/MySQL-PostgreSQL_features.html
- ▶ Why not MySQL?
<http://openacs.org/philosophy/why-not-mysql.html>
- ▶ Firebird
<http://firebird.sourceforge.net/>
- ▶ Interbase
<http://freshmeat.net/projects/interbase>
- ▶ SQLite
<http://freshmeat.net/projects/sqlite/>
- ▶ GNU SQL Server
<http://freshmeat.net/projects/gnusqlserver>
- ▶ “SQL Databases for Linux”
<http://freshmeat.net/articles/view/305/>
- ▶ “Non-SQL Databases for Linux”
<http://freshmeat.net/articles/view/307/>
- ▶ MRTG
<http://people.ee.ethz.ch/~oetiker/webtools/mrtg/>
- ▶ Eclipse
<http://www.eclipse.org>

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Symbols

/etc/exports file 65
<Connector></Connector> element 154
<Context></Context> element 157
<Engine></Engine> element 155
<Factory></Factory> element 165
<Host></Host> element 156
<Logger></Logger> element 158
<Server></Server> element 153
<Service></Service> element 154
<Valve></Valve> element 159

Numerics

1U rack mount server 9

A

ACCEPT rule 203
additional directories 64
aliases 53
AMaViS 113
Anaconda 20
Anomy 108
Anomy Sanitizer 106
Ant 152
Apache 14
 authentication 58
 configuring and testing 71
 integrating with Tomcat 167
Apache 2.0 root 71
Apache HTTP Server 14, 68
Apache HTTP Server Version 2.0 68
Apache JServ 141
Apache Software Foundation (ASF) 68, 141
Apache Web server 68
appBase 163
application development 205
application foundation 17
application logic 192
application services 12
architectural pattern 8
architecture 7
 non-functional requirements 13
 tiers 10
ASF (Apache Software Foundation) 141
attributes 37
auth_idap-1.4.8-3.i386.rpm 39
authdaemonrc 104
authentication 12
authoritative servers 14
authorized client network 91
automatic failover 197

B

basic authentication 58
Bayesian filtering 113
Berkeley Internet Name Domain (BIND) 15
BIND (Berkeley Internet Name Domain) 15
blade 9
blade server 8
blade-based computing 9
blade-based server 9
BladeCenter 3, 9
 business value 4
 Kickstart requirements 21
 sample Kickstart configuration 22
 when it is not the right platform 10
BOOTP server 19
business value
 BladeCenter 4
 Linux 4
 open source 4

C

caching servers 14
CD-ROM 18
central authentication 52
Certificate Authority 165
Certificate Signing Request (CSR) 166
Chain Certificate 166
chaining servers 196
chassis 9
CIFS (Common Internet File System) 62
cluster configuration 33
cluster logic 192
clustered server 114
cn 36
Coccinella 129
Common Internet File System (CIFS) 62
communication protocol 67
Comprehensive Perl Archive Network (CPAN) 107
connector 154
container 141
context configuration XML file 163
Courier 85
 IMAP and POP3 93
 replacing UW-IMAP 92
 Squirrelmail 94
 virtual mail server 99
 virtual users and domains 94
CPAN (Comprehensive Perl Archive Network) 107
CSR (Certificate Signing Request) 166
customized scripts 197
Cyrus IMAP 85

D

- daemons 14
- data management tier 11
- dead node 26
- demilitarized zone (DMZ) tier 11
- deployment descriptor 150
- DHCP
 - failover protocol 26
 - fault tolerance 26
 - server 21
- DHCP (Dynamic Host Configuration Protocol) 14, 26
- direct routing 74
- Directors 73–74
- DNS 29
 - highly available 32
 - server 19
- DNS (Domain Name System) 14
- domain name services 84
- Domain Name System (DNS) 14
- Dynamic Host Configuration Protocol (DHCP) 14, 26

E

- e-business infrastructure 2
- e-business pattern 8
- Eclipse 212
 - for Linux 214
 - for Windows 213
 - hardware and operating system prerequisites 213
 - Hello, world! program 217
 - Project 213
 - software prerequisites 213
 - Tomcat plug-in 223
 - workbench 214
- Eclipse Tools Project 213
- EclipseTechnology Project 213
- Editors 215
- e-mail 83
 - building a server with Sendmail and UW-IMAP 85
 - domain name services 84
 - message access 84
 - virus 105
- e-mail address whitelist 110
- Enterprise Information Portal 171
- entry 36
- ethers 53
- events 140
- Exodus 133

F

- FAST SAN storage 3
- fault tolerance 26
- file services 61
- floppy drive 18
- forward chain 203
- foundation 17
- fping 210
- FQDN (fully qualified domain name) 125
- fully qualified domain name (FQDN) 125

G

- gateway 136
- genhash command 77
- gethostip 20
- givenname 36
- gq 45
- group 53

H

- hardware 18
- heartbeat daemon 30
- heartbeat patch 33
- Hello, world! in Java 217
- high availability 12, 194
- high-level architecture 10
- home directories 56
- horizontal scaling 193
- host appBase 163
- hosts 53
- httpd.conf file 168
- human-machine communication 137

I

- IMAP (Internet Message Access Protocol) 85, 87
- inetd daemon 86
- inetOrgPerson object class 37
- Inflex 113
- infrastructure
 - functional aspects 12
 - network 25
 - non-functional aspects 13
 - objectives 3
- initrd 18
- initrd-everything.img file 20
- input chain 203
- instant messaging 123
- Intel i386 BIOS technology 19
- Internet e-mail systems 83
- Internet Message Access Protocol (IMAP) 85
- Introspection 140
- invoker servlet 144
- IP network 66
- IP tunneling 74
- IPChains 202–203
- IPTables 203
- ipvsadm code 74
- ISPMAN 105

J

- Jabber 123
 - clients 124, 128
 - Coccinella 129
 - configuration file 127
 - Exodus 133
 - farming 136
 - server 125
 - servers 124
- jabber.xml 127

- jabberd 127
 - for an intranet 135
 - installing 127
 - spool directory 127
- jabberd 1.0 124
- jabberd server 124–125
- Jakarta 141
- JAMM (Java Mail Manager) 99
- Java Mail Manager (JAMM) 99
- Java Servlet Development Kit (JSDK) 141
- Java Servlet Specification 140
- Java Web application 147
- JavaBeans 140
- JavaServer Page (JSP) 140
- Jetspeed 15, 149, 171
- jetspeed.war file 149
- JK connector 167
- JK2 connector 167
- JSDK (Java Servlet Development Kit) 141
- Jserv 141
- JSP (JavaServer Page) 140

K

- Keepalived 75
- kernel module 204
- keyboard 18
- keyboard, video, and mouse (KVM) 18
- Kickstart 18, 20
- KVM (keyboard, video, and mouse) 18

L

- LAN Manager 62
- LDAP
 - Apache authentication 58
 - concepts 35
 - Data Interchange Format 38
 - entry 36
 - gq 45
 - hierarchy 37
 - home directories 56
 - modifying entries 43
 - object 36
 - objects 36
 - populating the directory 42
 - Samba authentication 58
 - server authentication 52
 - servers 35
- LDAP (Lightweight Directory Access Protocol) 34
- ldapadd command 42
- LDIF (LDAP Data Interchange Format) 38
- LDIF record 38
- Lightweight Directory Access Protocol (LDAP) 34
- Linux
 - business value 4
 - Sendmail clusters 113
- Linux Virtual Server (LVS) 14, 73
- load balancer 116
- load balancing 73
 - MySQL queries with workload manager 191

- logger 154
- LogLevel 114
- lokkit 203
- LVS (Linux Virtual Server) 14

M

- MAC address 19
- machine-human communication 137
- machine-machine communication 137
- mail 36
 - mail delivery agent (MDA) 83
 - mail exchange (MX) 117
 - Mail Retrieval Agent (MRA) 84
 - mail transfer agent (MTA) 83
 - maildir 84, 93–94
 - MailStone 122
 - makeuserdb command 95
 - management 12
 - management information base (MIB) 206
 - master-slave relationship 195
 - mbox 94
 - MDA (mail delivery agent) 83
 - message access 84
 - messaging 12
 - MIB (management information base) 206
 - migrate_aliases.pl 57
 - migrate_base.pl 57
 - migrate_group.pl 57
 - migrate_hosts.pl 57
 - migrate_netgroup.pl 57
 - migrate_netgroup_byhost.pl 57
 - migrate_netgroup_byuser.pl 57
 - migrate_networks.pl 57
 - migrate_passwd.pl 57
 - migrate_protocols.pl 57
 - migrate_rpc.pl 57
 - migrate_services.pl 57
 - migration script 57
 - MIMEDefang 113
 - mod_jk connector 167
 - mod_jk.conf file 168
 - mod_perl 69
 - MOD_SSL 14
 - mod_webapp connector 167
 - Model-View-Controller (MVC) architectural pattern 10
 - modular approach 8
 - MON 15, 209
 - mouse 18
 - MRA (Mail Retrieval Agent) 84
 - MRTG (MultiRouter Traffic Grapher) 15, 207
 - MTA (mail transfer agent) 83
 - MultiRouter Traffic Grapher (MRTG) 207
 - MX (mail exchange) 117
 - MX-based failover 117
 - MySQL 16, 184–185
 - load balancing with workload manager 191
 - replication 188, 191, 193
 - RPM packages 186
 - securing 187

N

- name server switch (NSS) 53
- NAT (Network Address Translation) 73
- NetBIOS 62
- netgroup 53
- network 12, 53
- Network Address Translation (NAT) 73
- network edge tier 11
- Network File System (NFS) 65
- network infrastructure 25
- NFS 61
 - configuring 65
 - packages 65
 - server 19
- NFS (Network File System) 65
- nfs-util 21
- nfs-utils package 65
- non-functional aspects 13
- non-functional requirements 13
- NSS (name server switch) 53
- nss_ldap-172-3.i386.rpm 39

O

- object class 37
- objectives of infrastructure 3
- Obtuse 113
- omshell 27
- open source
 - databases 183, 185
 - e-business infrastructure 8
 - e-business software components 11
- open source business value 4
- open source infrastructure 7
- Open SSL 14
- OpenLDAP 13, 39
 - client 42
 - server 35, 39
 - virtual mail server 99
- openldap-2.0.21-1 39
- openldap-clients-2.0.21-1 39
- openldap-devel-2.0.21-1 39
- openldap-servers-2.0.21-1 39
- OpenSSH 201
- operating system instance 18
- operational components 13
- organizationalPerson object class 37
- output chain 203

P

- PAM (pluggable authentication module) 53
- passwd 54
- pattern 8
- Patterns for e-business 8
- Perl module 69
- persistence 140
- person object class 37
- Perspective 215
- PHP module 70
- php-ldap-4.0.6-16.i386.rpm 39

- pluggable authentication module (PAM) 53
- POP3 (Post Office Protocol) 85, 87
- port attribute 165
- portal 171
 - Jetspeed 171
- portlet 176
- Post Office Protocol (POP3) 85
- Postfix 14, 84
 - replacing Sendmail 90
 - virtual mail server 99
 - virtual users and domains 94
- PostgreSQL 184
- pre-execution environment (PXE) 18–19
- private switched network 119
- protecting a site 106
- protocols 54
- public key cryptography 201
- PXE (pre-execution environment) 18–19
- PXE boot 19
- pxelinux.0 19
- pxelinux.0 file 20

Q

- qmail 84
- qmail scanner 113
- QueueDirectory 114–115

R

- RAFC (Resource Access Control Facility) 34
- Realserver 73
- Red Hat distribution 21
- Red Hat Kickstart 20
- Redbooks Web site 234
 - Contact us xi
- RedHat Package Management System (RPMS) 18
- REJECT rule 203
- request 141
- Resource Access Control Facility (RACF) 34
- resource records 14
- Root Certificate 166
- root context 144
- rootpw 41
- round-robin DNS 115
- rpc 54
- RPMS (RedHat Package Management System) 18
- rsync utility 30

S

- Samba 15, 62
 - adding users 63
 - basic file server 62
 - file services 61
 - packages 62
 - passwords 63
- Samba authentication 58
- Samba Web Administration Tool (SWAT) 62
- SAN storage 10
- scale-out strategy 9

- scale-up strategy 9
- schema 37
- scp 201
- scripts 57, 197
- Secure Sockets Layer (SSL) 68
- security 12, 199
 - good practices 200
 - IPChains 203
 - OpenSSH 201
 - segregate networks 202
- segregate networks 202
- Sendmail 83, 86
 - building an e-mail server with UW-IMAP 85
 - clusters on Linux 113
 - installing and activating 86
 - replacing with Postfix 90
- server authentication with LDAP 52
- server configuration 32
- Server Message Block (SMB) protocol 62
- server.xml file 152, 169
- server-to-server communication 128
- service command 64
- services 54
- servlet 140
- servlet development environment 145
- servlet engines 141
- servlet reloading 144
- ServletContext 147
- shadow 54
- shared storage 11
- shared-nothing-server-farm 193
- simple network management protocol (SNMP) 206
- simple servlet 146
- single host 66
- slapadd command 42
- SMB (Server Message Block) 62
- smbadduser command 63
- smbmount command 64
- smbpasswd command 63
- sn 36
- SNMP (simple network management protocol) 206
- snmp utilities 207
- snmpwalk command 207
- socket factory 165
- software stack 10, 13
- spam 105
- SpamAssassin 106
- Spambouncer 113
- Squirrelmail 94
- ssh 201
 - ssh-add 201
 - ssh-agent 201
 - sshd 201
 - ssh-keygen 201
- SSL
 - module 68
 - Tomcat 164
- stateful packet filtering 203
- stateless packet filtering 203
- structural pattern 8

- su command 201
- SWAT (Samba Web Administration Tool) 62
- switch 11
- system management 205
- System V command 143
- systems management tools 15

T

- Tcl/Tk 129
- TFTP server 19–20
- three-tier e-business pattern 8
- Tomcat 15, 141
 - configuration file 152
 - integrating with Apache 167
 - plug-in for Eclipse 223
 - required files 142
 - SSL 164
 - Web Application Manager 159
- top object class 37
- translation 141
- transport 136
- transport map 102

U

- uid 36
- University of Washington IMAP server 85
- UW-IMAP
 - building an e-mail server with Sendmail 85
 - installing and activating 86
 - replacing with Courier 92
 - server 85

V

- video 18
- Views 214
- virtual mail servers with Postfix, OpenLDAP, Courier 99
- Virtual Routing Redundancy Protocol (VRRP) 74
- virus, e-mail 105
- vmlinuz file 20
- VRRP (Virtual Routing Redundancy Protocol) 74

W

- WAR file 148
- Web application 140
- Web Application Archive (WAR) file 148
- Web Application Manager 159
- Web application server 140
- Web cluster 74–75
- web.xml 150
- WEB-INF directory 148
- Webmail 94
- whitelist of e-mail addresses 110
- wildcard 66
- workers.properties file 169
- workload manager 191

X

xinetd daemon 86

Z

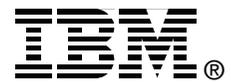
zones 14



Redbooks

IBM @server BladeCenter, Linux, and Open Source: Blueprint for e-business on demand

(0.5" spine)
0.475" x 0.873"
250 <-> 459 pages



IBM *e*server BladeCenter, Linux, and Open Source Blueprint for e-business on demand



Discover open source projects to reduce cost and improve reliability

Install and configure Linux and critical open source network services

Learn best practices to implement reliable services

Every construction project relies on a few critical components. When building a house, the critical components include the foundation, plumbing, and electrical wiring. When building a computing environment, the critical components include a robust operating system, file, and network services.

This IBM Redbook takes a modular approach to building an e-business on demand infrastructure. It covers many topics including Linux installation on IBM *e*server BladeCenter™ and IBM Fibre Array Storage Technology (FAST) storage area network (SAN) storage. This redbook explains:

- ▶ How to implement failover for core Internet services
- ▶ How to use a single LDAP directory for Linux system accounts, Apache, Samba, Postfix, Sendmail, and Jetspeed
- ▶ An implementation of load balanced services using Linux Virtual Server (LVS), and failover with Linux Heartbeat
- ▶ How to install and configure critical file services using Linux, Network File System (NFS), Samba, and IBM FAST storage
- ▶ Practices for security, systems management, configuration, and performance

If you are looking to reduce the cost of your computing infrastructure, provide critical IT services, install Linux on BladeCenter blades, and install and configure SAN storage with Linux and BladeCenter, this redbook is for you.

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:
ibm.com/redbooks**