- Table of Contents
- Index

Sun™ ONE Messaging Server: Practices and Techniques for Enterprise Customers

By Dave Pickens

Start Reading

Publisher: Prentice Hall PTR
Pub Date: September 18, 2003
ISBN: 0-13-145496-X
Pages: 288

As messaging or email increases in volume and size, the need for a scalable enterprise messaging system becomes more apparent to many organizations. The Sun(TM) ONE Messaging Server product fills this requirement and more. However, as with any system, the planning, installation, and routine maintenance tasks have a significant impact on throughput and availability.This book details best practices for architecting, deploying, and integrating the Sun ONE Messaging Server 5.2 product. It covers topics ranging from the basics of planning the system, to a sample installation, and on to monitoring and tuning the system to ensure that it is operational.

- Table of Contents
- Index

Sun™ ONE Messaging Server: Practices and Techniques for Enterprise Customers

By Dave Pickens

Start Reading

Publisher: Prentice Hall PTR
Pub Date: September 18, 2003
ISBN: 0-13-145496-X
Pages: 288

Copyright
Figures
Tables
Code Samples
Acknowledgments
Preface
- Sun BluePrints Program
- Who Should Use This Book
- Before You Read This Book
- How This Book Is Organized
- Related Documentation
- Shell Prompts
- Typographic Conventions
- Ordering Sun Documents
- Accessing Sun Documentation
- Using UNIX Commands
- Contacting Sun Technical Support
- Sun Welcomes Your Comments

Chapter 1. Messaging Overview
- Connectivity
- Number of Devices
- Number of Messages
- Average Message Size
- Protocols
- Security and Privacy
- Regulatory Issues

Chapter 2. Messaging Services
- Sun's Messaging Strategy

Messaging Services Beyond the Basics
Integrated Yet Open—Project Orion
SDN Concept
Conclusion
Chapter 3. Messaging Architectures
Directory
MTA
Mailstore
Proxy Servers
Simple Single-Layer Architecture
Simple—Alternative Architecture
Typical Architecture
Secure—Basic Architecture
High Availability—Failover Architecture
Chapter 4. Installation Preparation
Preparation Process
Network Connectivity
Chapter 5. System Startup
Basic System Status
Provisioning
Sample Data File
Sample Provisioning Script
Test User Generation Script
Chapter 6. Software Installation and Configuration
Simple Installation
Automated Installation Script
Chapter 7. Message Transfer Agent Configuration
Changing the Mappings
Direct LDAP Lookup
Adding New Domains to the MTA
SMTP Authentication
Chapter 8. Advanced Messaging Client Configuration
What Is a Shared Folder?
Supported Standards
Limitations
Setup Procedures
Chapter 9. Customization
Changing and Adding a Logo
Removing and Adding Options on the Options Tab
Single Sign On
Setting the Initial Welcome Email
Over-Quota Limits and Warning Email
Customizing Return Errors
Chapter 10. Security
Network
System
Messaging Software Protocols
Conclusion
Chapter 11. Migration
Basic Steps (Generic)
Sendmail (UNIX Mail)
Exchange, Novell Groupwise, and Lotus Notes

Chapter 12. Performance Tuning
Netscape Directory Server
Solaris OE
MMP
MTA Tuning
Notices
Postmaster Mail
Chapter 13. Advanced MTA Configuration
Conversion Channel
Other Possibilities
Chapter 14. Highly Available Messaging Deployment
High Availability Architecting Differences
Conclusions
Chapter 15. Managing Messaging Services and Preventive Maintenance
Periodic Maintenance Checklists
Chapter 16. Monitoring a Sun ONE Messaging Server
SNMP
Alternative Tools
Appendix A. Case Studies
Acme University
Baker Tech
Community City College
Appendix B. Majordomo Integration
Preparing for Integration
Glossary
Bibliography
Index

PREVIOUS NEXT

Copyright

Figures

FIGURE 3-1 Messaging Server, Storage, and Firewall Messaging System

FIGURE 3-2 Alternate Configuration With SMTP Firewall

FIGURE 3-3 Alternate Configuration With SMTP Relays and Firewall

FIGURE 3-4 Proxy Configuration With SMTP Relays and Firewall

FIGURE 3-5 Simple Failover Configuration

FIGURE 3-6 Failover With Relays and Firewall

FIGURE 5-1 top Command Output

FIGURE 5-2 Administration Interfaces Architecture Overview

FIGURE 5-3 Delegated Administrator for Messaging

FIGURE 6-1 Simple Architecture With Administration Ports

FIGURE 6-2 DC Tree and UG Organization Tree

FIGURE 8-1 Web Mail Shared Folder Permissions

FIGURE 8-2 Getting to the Permissions Screen

FIGURE 8-3 Sharing a Folder Other Than the Inbox

FIGURE 10-1 Security Layers

FIGURE 10-2 Secure Network Architecture for Messaging Environment

FIGURE 13-1 MTA Conversion Channel Diagram

FIGURE 14-1 High Availability Configuration Failover

FIGURE 14-2 Failover Using Both Nodes in a High Availability Configuration

FIGURE A-1 Acme University Architecture Diagram

FIGURE A-2 Baker Tech Architecture Diagram

FIGURE A-3 Community City College Architecture Diagram

Tables

TABLE 6-1 Values Required for Installation

TABLE 8-1 Web Mail Permission and RFC2086 Rights

TABLE 10-1 Enterprise Messaging Access in a Typical Enterprise

TABLE 10-2 Enterprise Messaging Access in a University

Code Samples

CODE EXAMPLE 5-1 ps -ef Command Output

CODE EXAMPLE 5-2 configutil Output—Current Configuration Settings

CODE EXAMPLE 5-3 Sample CLI Showing Creation of "testuser" Account

CODE EXAMPLE 5-4 Sample Template

CODE EXAMPLE 5-5 Test User Script Usage Example

CODE EXAMPLE 5-6 Add Test User Script Error Message

CODE EXAMPLE 5-7 Add Test User Completion Message

Acknowledgments

This book was certainly not a one-person effort. There are many people to thank and I am sure I will miss a few.

First and foremost are the other contributors to this effort: Portia Shao, Chad Stewart, and Dan Liston. They all added significantly to this book in terms of content, technical review, and overall comments. This book would not be as good nor as complete without their contributions. Portia Shao contributed the Advanced Messaging Client Configuration chapter, Chad Stewart contributed the Performance Tuning chapter, and Dan Liston contributed the Majordomo appendix.

As a technical product manager, Portia frequently provides answers and research regarding the messaging server to the engineers in the field. Chad is a Senior Consultant at Sun Microsystems working in the Professional Services Organization. Dan contributes to the free software environment by supporting majordomo.

Next, I would like to thank Kelly Caudhill for her time and effort during the final months of this project to review rough drafts and provide feedback.

I cannot fail to mention the best help that a writer at Sun could have—George Wood, the writer/editor who kept me on my toes and pitched in to write some portions when words just would not come to mind; Billie Markim and Sue Blumenberg for additional editing assistance; and Dany Galgani, the graphics designer who turned my scribbles into art.

I would also like to thank my manager, Casey Palowitch, for his support this past year and for encouraging me to tackle a project of this magnitude.

Last but not least, I would like to thank my wonderful wife and kids, who put up with me working many long and late hours.

Preface

The *Sun™ ONE Messaging Server Practices and Techniques for Enterprise Customers* book is published under the auspices of the Sun BluePrints™ program. This book is a collection of practices and techniques for deploying a messaging system. These practices and techniques have been gathered from many customers' messaging system deployments and internal testing labs. The book covers some things that advanced users might believe is common knowledge but is not. The goal of this book is to make the administration of Sun™ Open Net Environment (Sun ONE) Messaging Server (formerly known as iPlanet™ Messaging Server) easier by collecting this knowledge and organizing it as you might encounter it during the deployment of a messaging project, that is, from planning to day-to-day operation.

Sun BluePrints Program

The mission of the Sun BluePrints program is to empower Sun's customers with the technical knowledge required to implement reliable, extensible, and secure information systems within the data center using Sun products. This program provides a framework to identify, develop, and distribute preferred practices information that applies across the Sun product lines. Experts in technical subjects in various areas contribute to the program and focus on the scope and advantages of the information.

The Sun BluePrints program includes books, guides, and online articles. Through these vehicles, Sun can provide guidance, installation and implementation experiences, real-life scenarios, and late-breaking technical information.

The monthly electronic magazine, *Sun BluePrints OnLine*, is located on the Web at:

http://www.sun.com/blueprints.

To be notified about updates to the Sun BluePrints program, please register on this site.

Who Should Use This Book

This book is intended for readers with varying degrees of experience with and knowledge of computer system and server technology, who are designing, deploying, and managing a Sun ONE Messaging Server within their organizations. Typically these individuals already have UNIX® knowledge, but have been given the added responsibility for messaging too.

The book is targeted at enterprise customers deploying the Sun ONE Messaging Server software version 5.2 and later. An enterprise customer is an organization that is running messaging for its own internal use and is not providing messaging services to other organizations; that is, it is not an applications service provider (ASP) or Internet Service Provider (ISP). The organization could be small (thousands of users), large (100,000 users), or anywhere in between. This book offers practical advice on design, architecture, deployment, and operation, with these customers in mind.

Before You Read This Book

This book covers some of the basics of messaging and the services such as Domain Name Service (DNS) or Lightweight Directory Access Protocol (LDAP) that messaging relies upon, but cannot address these services thoroughly. You should have some basic knowledge of messaging systems and architecture, and be comfortable with using GUI-based tools and the UNIX command line (shell). See one or more of the following documents for this information.

- *DNS and BIND*, 4th Edition, October 2002, O'Reilly http://www.oreilly.com/catalog/dns4/
- *DNS & BIND Cookbook*, October 2002, O'Reilly http://www.oreilly.com/catalog/dnsbindckbk
- *LDAP System Administration*, March 2003, O'Reillyhttp://www.oreilly.com/catalog/ldapsa/
- *Essential Systems Administration*, 3rd Edition, August 2002, O'Reilly http://www.oreilly.com/catalog/esa3/
- Sun BluePrints on Naming and Directory Services
  http://www.sun.com/solutions/blueprints/browsesubject.html#nds

How This Book Is Organized

This book is modeled after the typical process an enterprise uses to deploy its messaging infrastructure, from the initial planning steps to day-to-day operations.

It follows a basic systems development life cycle (SDLC) for an enterprise messaging system—planning, testing, deployment, and maintenance. Each of these phases addresses practices and techniques to enhance availability, performance, and ease of use.

The book has 16 chapters and two appendixes.

Chapter 1, "Messaging Overview," on page 1—This chapter provides an overview of the factors facing messaging implementations, how messaging systems are being used, what the messaging trends within enterprises are, future uses of messaging currently being developed, and so forth. This chapter is designed to provide the basis for establishing messaging as a mission-critical system within the enterprise and expose readers to issues that they may not currently be considering.

Chapter 2, "Messaging Services," on page 7—This chapter provides an overview of the Sun ONE Messaging Server product as it fits into the software delivery network (SDN) concept, along with brief descriptions of the individual components that go into making an enterprise messaging system work. It highlights specific strengths of the Messaging Server compared with other offerings in the market. The main emphasis of this chapter is on covering the interoperability of products that support open standards and the advantages they bring.

Chapter 3, "Messaging Architectures," on page 15—This chapter describes the architectures of some of the more common configurations and explains that there are almost infinite combinations. It outlines the pros and cons of each architecture to provide you with information to determine which architectures meet your enterprise messaging requirements.

Chapter 4, "Installation Preparation," on page 31—This chapter outlines some issues and practices that are important during the pre-installation. These issues can have significant impact on installation, operations, and recovery capability. It provides insight into situations that normally cause consternation. References are made to specific sections of manuals or additional supplemental materials. Think of this chapter as a reminder regarding operating system best practices that can be found in other BluePrints and elsewhere.

Chapter 5, "System Startup," on page 41—This chapter covers the basics of getting the system started and provisioning users once the system is operational. It is designed to provide an understanding of the various mechanisms for provisioning as well as the pros and cons of each method. You can easily automate provisioning, but there are times when manual entry is required too.

Chapter 6, "Software Installation and Configuration," on page 69—This chapter provides information and caveats that you may need during the installation phase of the overall messaging environment. It also discusses scalability issues. For additional details, refer to the *iPlanet Messaging Server Installation Guide for UNIX*.

The chapter discusses the pros and cons of various answers to configuration questions and installation options so that you can avoid post-installation pitfalls, whether they are related to flexibility (that is, top domain name selection in directory), scalability, availability, performance, or ease of use. Thus, this chapter covers items not found in the current documentation and conveys information that can only be learned through experience

Chapter 7, "Message Transfer Agent Configuration," on page 91—This chapter provides best practices and techniques regarding the setup and configuration of the Message Transfer Agent (MTA) component within the Sun ONE Messaging Server. Due to its complexity, this is an area that can cause significant issues related to security as well as basic functionality. This section dissects the default "out-of-the-box" MTA configuration file to provide a starting point for the reader. Many users of the previous versions, Sun Internet Mail Server (SIMS) or Netscape Messaging Server (NMS) had never seen an Innosoft PMDF product MTA configuration file. Therefore, this area is very intimidating and confusing. This chapter addresses some typical changes in plain language.

Chapter 8, "Advanced Messaging Client Configuration," on page 103—This chapter covers the following key concepts and topics for using shared folders: what a shared folder is, supported standards, limitations, how to let your administrator read your mailbox, and how to share a folder in an Internet Message Access Protocol (IMAP) client, Netscape Messenger, and Outlook Express.

Chapter 9, "Customization," on page 123—This chapter describes how to customize the Messaging Server. Customers typically make several customizations right after installing the basic Messaging Server (Sun ONE Directory Server, Sun ONE Web Server, Sun ONE Delegated Administration, email, and perhaps even Sun ONE Calendar Server). The most common of these include changing the look and feel of the web mail interface (Sun One Messenger Express and providing a single sign on (SSO) between the web mail, web-based calendar, and Delegated Administration interfaces. Some of the other common customizations that are done almost immediately include defining the welcome message for new accounts, along with the over-quota message for people about to go over quota or already over quota. Some customers would also like to customize some of the return errors that the message system sends back to users.

Chapter 10, "Security," on page 153—This chapter discusses in detail the specific issues surrounding the security of the Messaging Server, including the server platform, the various protocols and their impact, and securing the contents of the messages. This chapter divides the topic of security as it relates to the Messaging Server into three different layers or topics—network, system, and messaging system protocols.

Chapter 11, "Migration," on page 167—This chapter describes the best practices for migration and identifies potential problems that may occur during the migration phase. After the basic Messaging Server is installed, one of the more difficult tasks is to migrate the existing user base and mailbox contents. Different techniques can be used, but only specific techniques are valid for specific migrations, Exchange for example. Additionally, other parts of the migration have specific issues, such as using the migration as an opportunity to standardize mail address formats while maintaining legacy addresses that can be addressed.

Chapter 12, "Performance Tuning," on page 179—As with any system, performance is a key element to getting the most return on investment, as well as maintaining happy users. This chapter contains practices and principles specifically related to performance tuning of the Messaging Server, which may differ or contradict conventional tuning wisdom. This chapter points out the areas on which a Messaging Server administrator should concentrate.

Chapter 13, "Advanced MTA Configuration," on page 189—This chapter contains examples of the conversion channel feature of the MTA, including some sample scripts. It also discusses some of the other possibilities for advanced MTA configuration.

Chapter 14, "Highly Available Messaging Deployment," on page 201—Some organizations do not see messaging as a mission-critical service or, for whatever reason, they decide not to implement highly available messaging. This chapter re-enforces why messaging is mission critical and needs high availability. It addresses specific issues (pros and cons) with various high-availability architectures that customers have implemented as well as some of the caveats to keep in mind when planning and installing messaging in a high-availability environment. These lessons have been learned the hard way at various customer sites and are found nowhere else in the documentation or technical notes.

Chapter 15, "Managing Messaging Services and Preventive Maintenance," on page 209—As with any system, your messaging server requires routine maintenance. This chapter outlines the best practices and issues surrounding day-to-day and routine maintenance involved in managing a messaging server, specifically the Sun ONE Messaging Server. While the current documentation explains the basic commands, it does not address automation or scripting of these functions, nor does it adequately cover techniques that can improve backup and recovery time.

Chapter 16, "Monitoring a Sun ONE Messaging Server," on page 215—This chapter explains how to monitor your systems and the Messaging Server software that comprises your email infrastructure. System monitoring is an important part of the overall management effort. Tools can range from simple monitoring of the basic hardware and network infrastructure to more complex monitoring such as response time and error logging. They can be homegrown, open source, or commercial products. You can implement one or many.

Appendix A,"Case Studies," on page 221—This appendix contains a series of case studies to illustrate several points made throughout this book as well as to highlight some specific lessons learned. Architecture diagrams and time lines are provided for reference. These cases occurred over the past few years and are actually a composite of the case studies of several different customers.

Appendix B, "Majordomo Integration," on page 231—This appendix contains procedures for integrating all of the functionality of majordomo with sendmail into the Messaging Server.

This book is based on the following software:

- Solaris™ 8 or Solaris 9 Operating Environment (Solaris OE)
- Sun ONE Messaging Server 5.2
- Sun ONE Directory Server 5.1
- Sun ONE Web Server 6.0
- Sun ONE Calendar Server 5.1.1

It does not cover in detail basic UNIX administration, DNS or LDAP services, command reference information, or other information that is normally found in the product manuals. Moreover, the book does not address older versions of messaging software such as Sun™ Internet Mail Server (SIMS v3.x or SIMS v4.x) software or Netscape Messaging Server (NMS v3.x or NMS v4.x) software.

Related Documentation

The following table lists manuals that provide additional useful information. The Sun ONE products were formerly known as iPlanet products so the titles of many of the manuals listed contain iPlanet instead of Sun ONE.

Title	Author and Publisher	Part Number
iPlanet Messaging Server 5.2 Administration Guide	Sun Microsystems	816-6009
iPlanet Messaging Server Installation Guide for UNIX	Sun Microsystems	816-6014
iPlanet Directory Server Installation Guide	Sun Microsystems	816-5610
Sun ONE Calender Server 5.1.1 Installation Guide	Sun Microsystems	816-6414
iPlanet Messaging Server Reference Manual	Sun Microsystems	816-6020
iPlanet Messenger Express 5.2 Customization Guide	Sun Microsystems	816-6010
Solaris 8 (SPARC Platform Edition) Installation Guide	Sun Microsystems	806-0955
Solaris 9 Installation Guide	Sun Microsystems	816-7171
Solaris System Administrators Guide on Security Services	Sun Microsystems	806-4078

These manuals are located at:

http://docs.sun.com/db/prod/sunone.

Shell Prompts

Shell	Prompt
C shell	*machine-name*%
C shell superuser	*machine-name*#
Bourne shell and Korn shell	$
Bourne shell and Korn shell superuser	#

Typographic Conventions

Typeface	Meaning	Examples
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your.login file. Use ls -a to list all files. % You have mail.
AaBbCc123	What you type, when contrasted with on-screen computer output	% **su** Password:
AaBbCc123	Book titles, new words or terms, words to be emphasized. Command-line variables; replace with real names or values.	Read Chapter 6 in the *User's Guide*. These are called *class* options. You *must* be superuser to do this. To delete a file, type rm *filename*.

Ordering Sun Documents

The SunDocs^{SM} program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation

You can view, print, or purchase a broad selection of Sun documentation, including localized versions, at:

http://docs.sun.com/.

Using UNIX Commands

This document does not contain information on basic UNIX commands and procedures such as shutting down the system, booting the system, and configuring devices. See one or more of the following for this information:

- *Solaris Handbook for Sun Peripherals*
- AnswerBook2™ online documentation for the Solaris OE
- Other software documentation that you received with your system

Contacting Sun Technical Support

If you have technical questions about this product that are not answered in this document, go to:

http://www.sun.com/service/contacting.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. You can submit your comments by going to:

http://www.sun.com/hwdocs/feedback.

Sun ONE Messaging Server Practices and Techniques for Enterprise Customers, ISBN number 0-13-145496-X, part number 817-0763-10.

Please include the title, ISBN number, and part number of your document with your feedback.

Chapter 1. Messaging Overview

This chapter provides an overview of the factors facing messaging implementations today, how messaging systems are being used, what messaging trends within enterprises are, future uses of messaging currently being developed, and so forth. This chapter provides the basis for establishing messaging as a mission-critical system within the enterprise and exposes you to issues that you may not currently be considering. This chapter contains the following topics:

- Connectivity
- Number of Devices
- Number of Messages
- Average Message Size
- Protocols
- Security and Privacy
- Regulatory Issues

Electronic messaging, or email as it is more commonly referred to, is becoming more of a mission-critical network service every year. It is doubtful if any person in an organization can identify everyone or everything that relies upon the messaging system. Typically, the only time it becomes clear who and what actually relies upon the messaging system is when there is a major outage or problem. Many factors are behind this trend, driving messaging to becoming more and more mission critical. Some of these factors are:

- Connectivity is getting better.
- Number of devices is increasing.
- Number of messages (traffic) is increasing.
- Size of the messages (attachments) is getting larger.
- Protocols to access email are changing.
- Security is more of a concern.
- Regulatory issues

Connectivity

Today more bandwidth is available than at anytime before. It is no longer uncommon in an environment to find 10BASE-T switched network access, and in many cases now 100BASE-T, throughout an organization. In universities, it is common to find network ports in the dorm rooms (sometimes more than one port), faculty offices, study rooms, the library, and other places across campus. Some campuses are deploying Gigabit Ethernet in labs and select faculty offices. Corporate organizations are also deploying bandwidth like never before, with 100BASE-T to offices and Gigabit Ethernet in the data center and select facilities. Wires are no longer a constraint either. Bandwidth is even available from thin air as many organizations are deploying wireless networks (802.11a/b/g) or have plans in place to do so in the near future.

This access to bandwidth anytime and anywhere results in more messaging usage that now comes from a diverse population of clients (devices). No longer do users have to return to their *base of operations*, also known as a desk or cubicle, to send and receive email.

Older methods of modeling and understanding of messaging systems were based upon dial-up connections, low bandwidth, and limited access assumptions. In today's environment, these assumptions no longer apply.

Number of Devices

Cheaper electronics, personal digital assistants (PDAs), cell phones, and computers have resulted in a plethora of devices on the network, many of which are email enabled by default or can be quickly messaging enabled. It is no longer safe to assume a ratio of one person per device (access point). It is, given today's penchant for connectivity and always-on models, possible to have two or three access points per person. This can, in fact, lead to situations where users are generating two or three connections simultaneously. It is not that humans (or the software for that matter) have learned to multitask so well, but rather that humans are not logical.The scenario of a student running to class while leaving a desktop computer running (and checking email in the background), accessing email from class or across campus with a PDA or laptop, is not far fetched. In the corporate world, an equivalent scenario would be J. Q. Manager leaving an office desktop running (and checking email) while leaving for a meeting and checking email on a PDA during the meeting. This means that you can no longer simply say one user equals one connection (device), and must plan for more connections in the future.

Number of Messages

In many ways, email has taken over what the telephone used to do. Today it is not uncommon for someone who is an active email user to receive over 100 messages per day or more. Try to think of the last time you had 100 voice mail messages waiting in your voice mail box. Some say that instant messaging (IM) is going to overtake email and email will be obsolete. There is no doubt that instant messaging will affect email in some manner, but IM is a real-time communication method akin to actually talking on the phone. Email is like calling someone who is not there or is busy, and leaving a message on their answering machine or voice mail. Email is asynchronous and does not require the user's immediate attention like instant messaging does, although many people leave email running all the time and use it like IM in some ways.

Another issue with IM is interoperability. IM is an immature technology when compared with email. It is hard to bridge across Yahoo! and AOL or MSN using IM, for example. The situation is getting better with the advent of new protocols such as Simple Internet Protocol (SIP) and SIMPLE, but IM is not there yet—and it is not quite as universal as email.

Another issue driving up the quantity of messages being sent and received is that other systems are becoming more integrated with email. Today many organizations are looking for *unified messaging*, providing a single point for email, faxes, and voice mail. Unified messaging allows integration between an organization's voice mail system (or fax system) and an email (messaging) system in such a way that the voice mail system actually stores the voice mail messages in a person's email inbox (or other folder). That way you can read your email and listen to your voice mail (or see your faxes) without having to check two separate systems. This capability adds yet another factor in terms of volume as well as size, since audio attachments can be large depending upon the sampling rate.

At some point in the future, IM might actually participate in this unified messaging environment. Imagine that email becomes the answering machine or recording device for IM sessions—for example, you are not able to participate in the 11:00 a.m. IM session to discuss the new marketing campaign, but the conference (including all the attachments and collaboration) gets saved in your inbox. How exactly has IM reduced your messaging requirements? IM might, in fact, add more traffic to your messaging system.

Average Message Size

Partly because of the increase in bandwidth but also as a result of the desire for fuller, richer multimedia experiences (for example, singing and dancing PowerPoint presentations), the average email is getting bigger. Where three or four years ago it was normal to have 10-kilobyte messages with occasional 100-kilobyte messages traversing the messaging system, today those figures are noticeably larger—somewhere around 25 to 30 kilobytes average message size, with occasional multimegabyte (one megabyte plus) messages appearing more frequently. Older models for messaging systems that were fine during the days of dial-up Internet where a 10-kilobyte email would take a minute to send just do not apply today.

Protocols

Older models for architecture and sizing generally based everything on Post Office Protocol (POP) and Simple Mail Transfer Protocol (SMTP) only. POP was for retrieving mail, and SMTP was for transferring mail between systems. Prior to POP, there was no protocol even to read email; rather, email clients like Pine and electronic mail (elm) really just browsed the inbox directly via the Network File Server (NFS) or the file system. These were typically sized as generic or lightweight interactive logins.

Today, Internet Message Access Protocol (IMAP) and web mail have taken over POP in enterprise accounts, and while SMTP is still the transfer protocol, other transfer protocols such as Short Messaging Service (SMS) for pagers and PDAs have been added too. SMS does not carry the same overhead in terms of headers, signatures, and attachments that SMTP does, but it does not do attachments either. One advantage that SMS offers beyond being lightweight is the ability to embed short responses such as *YES* and *NO* within the message for quick reply by the recipient. Environments such as hospitals that rely upon pagers typically use SMS to allow for a message with response.

Security and Privacy

This is a huge topic on which an entire book could be written. As we have increased reliance upon email, increased bandwidth, increased access to bandwidth, and increased the number of devices on the network, we have also increased the need for security and privacy. More and more customers are using Secure Sockets Layer (SSL) methods to secure the communication protocol whether it is POP, IMAP, SMTP, or HTTP (web mail).

Many customers are adding virus scanning to their messaging layer—what used to be uncommon (virus-scanning messages in the messaging system) is now common. In reality, this was not a complete surprise or a giant step. Many organizations began with scanning just messages coming into their system from the Internet. Two or three years ago, when customers asked about virus scanning, that was it. Then, it became necessary or desirable to scan outgoing email (being a good Internet citizen and all that) and to scan everything between users too. So, nowadays it more likely to scan everything due to issues of viruses within the enterprise.

In addition to virus scanning, many organizations also want to eliminate spam, also called unsolicited bulk email (UBE) or unsolicited commercial email (UCE). This adds yet an additional workload to the messaging system that was not there five years ago.

Regulatory Issues

New regulatory issues beyond those on privacy are facing institutions these days. One of the more recent interpretations of existing laws (the Freedom of Information Act or their state-level equivalents) classifies email as official written correspondence for schools and government entities. In other cases, email is becoming a legal issue due to the Enron-type accounting scandals. And so email regarding official matters must be archived or retained for a set number of years. Therein lies the problem. How exactly can you pinpoint which emails are related to official matters and archive only those emails?. Many times the answer is that you cannot. Therefore, archiving everything is required. Archiving increases the requirement for storage as well as the need for solid backup and recovery procedures. At Sun, the term "infinite mailbox" is being used to describe just such a message system.

Chapter 2. Messaging Services

This chapter provides an overview of the Sun ONE Messaging Server product as it fits into the software delivery network (SDN) concept, along with brief descriptions of the individual components that go into making an enterprise messaging system work. It highlights specific strengths of the Messaging Server product compared with other offerings on the market. The main emphasis of this chapter is on covering the interoperability of products that support open standards, and the advantages they offer.

Sun's Messaging Strategy

Sun Microsystems, Inc. was founded on the philosophy of open systems, open standards. The mantra at Sun is "agree on standards and compete on implementation." This philosophy is no different whether it is the Solaris OE or the Sun ONE Messaging Server product. In fact, the "ONE" in Sun ONE stands for Open Network Environment, in respect of open standards.

Open Standards

One of the nice things about messaging is that it is a mature area in the Internet space and has been around for more than 25 years. Thus, there are many mature, open protocols for messaging, unlike some of the other Internet protocols such as instant messaging (IM) or calendaring which still do not offer truly ubiquitous protocols although some are emerging like SIP/SIMPLE and iCAL. The current messaging protocols are:

- Internet Message Access Protocol (IMAP)
- Post Office Protocol (POP)
- Simple Mail Transfer Protocol (SMTP) and Extended Simple Mail Transfer Protocol (ESMTP)
- Lightweight Directory Access Protocol (LDAP)
- HyperText Transfer Protocol (HTTP)
- Secure Sockets Layer (SSL)

Popular Clients

By supporting standards, the Sun ONE Messaging Server is client agnostic, so Sun does not offer a thick (native) client for the various operating systems such as Windows, Mac OS, or Linux. Some of the more popular clients are:

- Netscape™ 7.0
- Mozilla
- Outlook
- Eudora
- Ximian

Any client that supports IMAP or POP along with SMTP should work just fine. Most modern clients go beyond this basic support, adding LDAP for address book lookup and SSL for security.

For a good technical overview of the Sun ONE Messaging Server product, including a list of supported open standards, obtain "Sun ONE Messaging Server version 5.2—A Technical Whitepaper" from your local Sun Sales Representative or System Engineer.

Messaging Services Beyond the Basics

Beyond the basics of providing messaging services, the issue is how these services are provided. Can the product scale? Is the product secure? How hard is the product to install and manage? How easily can users be provisioned? How flexible is the product? There are many messaging products out there, and each of them is architected and designed slightly differently. One product may store user names and passwords in a flat file, while others leverage LDAP. One product may provide integrated antivirus measures but not allow you to integrate a slightly better third-party product for antivirus protection.

There are several key items:

- Directory Services
- Web Messaging
- Address Book
- Calendar
- Portal
- Web Services
- Anyone, Anytime, Anywhere, Any Device

Directory Services

The directory is the brain or memory of the Sun ONE Messaging Server. It is used across the various products within the Sun ONE product line to provide user information, authentication, storage of policies and rules, configuration information, and registration of web services—for example, universal description, discovery, and integration (UDDI). It plays a central role in being able to easily provision accounts and services without managing separate user data for each application in an environment. By leveraging a directory as the central repository for user information, provisioning is a matter of granting privileges to the user or group of users to specific resources (services) by configuring attributes appropriately—by changing an attribute and access to a service. This eliminates the need to provision users in many separate systems.

Web Messaging

When the Web first started becoming a popular way to provide some abstraction regarding where you were located, the computer you were using, and the resource (for example, email) you were trying to access, adding an additional software package to provide this web mail interface was the norm. However, as time went by, this became a feature demanded by customers as part of the base messaging software, to eliminate the need to select, deploy, and manage something separate. By offering web mail as part of the messaging server, yet providing the ability to customize the "look and feel" of it for your users plus control which users have access to web mail, the Sun ONE Messaging Server offers savings over having to integrate a separate web mail software utility too. The nice part though is that should you decide, either for legacy or other reasons, to select and integrate another web mail interface—for example, IMP—you still have that option.

Address Book

A core requirement of messaging is being able to store and retrieve contact information. The Sun ONE Messaging Server leverages the underlying directory to provide personal address books. A new feature coming to the address book functionality is shared address books, which allow you to share your address book entries with other people and applications in a secure manner (for example, only those people and applications you wish to have access).

Calendar

As part of the overall Sun ONE product line, Sun offers a web-based calendar product called the Sun ONE Calendar Server. By providing calendar management for people, resources, and events, calendaring can be offered as a service to an entire organization and beyond.

The main issue regarding calendar technology adoption is lack of widely adopted calendar standards. iCal, SyncML, and vCal have been available for some time now; however, there is no single calendar standard that *all* vendors use.

Portal

Portals are very hot these days, but people rarely think beyond the basics to what lies behind the portal or makes a good portal. Simply put, a portal is technology that aggregates services and content together in a secure manner for a particular community of users. The services behind the scenes are things such as messaging and calendar services, while the content can be a variety of things, from static HTML content to true web applications and services.

A portal really brings to life the concept that the sum of the parts is greater than the whole. Without quality services and applications provided to the right people at the right time, a portal is just another pretty interface.

Sun's philosophy is to leverage network identity management and scalable services like the Sun ONE Messaging Server, along with world-class partners such as Altio and FatWire, to provide a best-of-breed approach to meet customer portal needs with the Sun ONE Portal Server product.

By combining these things and leveraging web services for rolling out new services, the Sun ONE Portal Server provides a solid portal platform, today and tomorrow.

Web Services

By making messaging a web service, or at least a service that it is always on and always there much like dial tone, the possibilities for use become significantly greater—now it can truly become the asynchronous messaging backbone for more than just person-to-person communication. Messaging can become integrated into workflow and business processing, becoming the transport of choice.

Anyone, Anytime, Anywhere, Any Device

Since early 1996 and before Sun released Java™ to the world, Sun's motto has been "Anyone, Anytime, Anywhere, and Any Device." This is definitely true with the Sun ONE Messaging Server.

By thinking "service" and providing device- and locale-neutral messaging, the number of nodes that can take advantage of such a messaging service (system) is enormous. Metcalf's law (formulated by Robert Metcalf, founder of 3COM and regarded as the inventor of Ethernet) states that the "value" or "power" of a network increases in proportion to the square of the number of nodes on the network.

Marc Andreesen, one of the founders of the Web, said:

> "A network in general behaves in such a way that the more nodes that are added to it, the whole thing gets more valuable for everyone on it because all of a sudden there is all this new stuff that was not there before. You saw it with the phone system. The more phones that are on the network, the more valuable it is to everyone because then you can call these people. Federal Express, in order to grow their business, would add a node in Topeka and business in New York would spike. You see it on the Internet all the time. Every new node, every new server, every new user expands the possibilities for everyone else who is already there."
>
> Reference: http://www.si.edu/resource/tours/comphist/ma1.html.

Integrated Yet Open—Project Orion

Project Orion is a new and innovative initiative with the goal of making enterprise infrastructure software predictable in its delivery, more freely accessible for evaluation, and even more affordable to purchase.

Project Orion is designed to take a view of the entire enterprise infrastructure software life cycle process, from development through production and ongoing operation, identifying and reducing the complexity and cost associated with each step.

Project Orion leverages Sun's proven competency in developing and releasing large-scale systems software, best demonstrated by its multi-platform Solaris OE. The effort will align the integration, testing, and release of all of the company's software products and pricing models. One of the biggest changes in Sun's software release strategy has been to create a specific release model where major Solaris OE releases are only done every two years, providing stability for customers, and predictable minor releases are scheduled like clockwork on a quarterly basis. This is sometimes referred to as the *Solaris train*. All new software or features that are ready are allowed on board and released as part of the Solaris OE. Any software or features that miss the train catch the next one the following quarter, assuming the boarding criteria have been met. This allows for both quality and rapid release of features.

Project Orion brings this release model to the Sun ONE software packages, just as the Sun Solaris train model does. As each individual Sun ONE software component product satisfies the Project Orion criteria, it boards the software train. Each software train leaves on a regular quarterly schedule. New component product features or versions that are not ready for boarding catch the next software train if they are ready. Each software train goes through extensive end-to-end testing based on customer use scenarios prior to shipping. Component products must successfully complete testing prior to shipping on a quarterly-release software train.

Project Orion also allows customers to select best-of-breed components from Sun's partners if they so choose. If you already have a specific Java Application Server, continue to use it—Sun ONE is integrated, yet open.

SDN Concept

Today's migration towards "always-on" services requires a new type of network architecture, one that is built from the top down with the goal of delivering software as "services" regardless of the final delivery technology (such as wireless or broadband) or what specific "service" is being delivered (for example, web services versus messaging). These solutions require optimal architectures that can support ubiquitous service access.

The emphasis on service delivery is the heart of the SDN, a service-based network architecture for data center deployments. The SDN architecture services provide a foundation for scalable e-services, such as those offered by Sun ONE, while helping customers meet demands for reliability and performance.

Several growth areas affect future computing platforms and the services delivered by organizations today:

- Growth and availability of network bandwidth
- Growth of data-intense wireless services
- The need for disaster recovery and mission-critical service delivery
- Growth of computer processing, taking advantage of rich content

These factors significantly enhance the need for scalable, highly secure, and high-performance network topologies that can support high-velocity change. The Sun Professional Services Software Delivery Network architecture service offerings have been developed to help customers meet these needs while supporting future technology requirements.

The SDN architecture is a highly scalable, maintainable, supportable network architecture that can be deployed in Internet data centers (IDCs), service provider networks (SPNs), and other areas and projects that are designed, integrated, and supported by Sun Professional Services and Enterprise Services as SunTone Certified, where possible.

The majority of SDN architecture sales are made in conjunction with a fairly large infrastructure solution project such as Messaging or Directory design and implementation. Many of these are for large service provider (SP) organizations, but the concepts, availability, and security issues apply to most organizations.

SDN architecture is project based, usually coupled with a data center implementation similar to the business model already seen in EMEA, often including Web services and Sun ONE or Wireless. It will be an essential component of these implementations to enable achievement of our customers' Quality of Service (QoS) requirements.

For more information see:

http://www.sun.com/service/sunps/architect/delivery/.

Conclusion

By sticking with open standards, thinking of messaging as a "service," and looking at future possibilities for use (for example, portals) when evaluating or architecting a messaging infrastructure, the result will be a solid, scalable, open architecture with flexibility to meet future needs not yet defined.

Chapter 3. Messaging Architectures

This chapter describes the architectures of some of the more common configurations and explains that there are almost infinite combinations. It outlines the pros and cons of each architecture to provide you with information to determine which architectures meet your enterprise messaging requirements. Chapter 10, "Security," on page 153," addresses security in detail, but a secure architecture is discussed in this chapter to indicate the use of firewalls in multiple layers, that is, a demilitarized zone (DMZ) as not all messaging systems actually are behind firewalls.

This chapter covers the following topics:

- Directory
- MTA
- Mailstore
- Proxy Servers
- Simple Single-Layer Architecture
- Simple—Alternative Architecture
- Typical Architecture
- Secure—Basic Architecture
- High Availability—Failover Architecture

Often there is more than one method of doing things. Designing and installing a messaging system is no different. Depending upon your organization's specific goals, skills, and networking environment, one architecture may be more relevant than another.

Generally, the architectures can be organized into several categories or combinations of categories:

- Simple Single Layer
- Multitiered
- Secure
- Highly available

To help you understand more about messaging architecture, this chapter reviews some of the basic parts of the messaging system first.

Four basic parts of a messaging system are important or can be sized.

- Directory
- Gateway, also called message transfer agent (MTA)
- Mail server, also called mailstore
- Proxy server

Directory

The directory or user store in the messaging architecture stores user information such as ID, password, and email address. The software of many messaging servers utilizes the user store mechanism of the host, such as /etc/hosts on the Solaris OE. Others, such as the Sun ONE Messaging Server, utilize a directory or LDAP service to store and access user information.

The Sun ONE Messaging Server ships with and requires a fully compliant LDAP directory that contains directory objects specific to the Sun ONE Messaging Server software. These directory objects extend the default Internet Engineering Task Force (IETF) schema with additional attributes. A complete guide to the Sun ONE Messaging Schema is part of the existing documentation.

These additional attributes contain information such as an alternate address, or alias as it is sometimes called. Other attributes are used to store user preferences for web mail and configuration information about email services, as well as group information (mailing lists) and personal address books. In the Messaging Server software, information regarding processing a user's inbound email such as vacation messages, server side filters, and forwarding is also stored in the directory.

The directory is a lot like a database—a very small, fast database. One thing to note is that when directories or LDAP were originally developed, they were primarily designed to be mainly read oriented, say a 90 percent read and 10 percent write ratio. Today's usage of the directory has changed significantly. Things like messaging, calendar, and portal all store preferences and information in a directory server. The read/write ratio is now closer to 80 percent read and 20 percent write. So it is critical that the directory is available and performance of the directory is good or better than good.

MTA

The MTA, which is sometimes referred to as the mail gateway or SMTP server, routes the mail to its destination or rejects it if not properly formatted or addressed, or simply not for this messaging system. This is typically done via SMTP or related protocols such as the Extended Simple Mail Transfer Protocol (ESMTP) and Local Mail Transfer Protocol (LMTP). Basically, the MTA is to email what a Cisco router is to Ethernet packets.

The MTA is also what determines how messages are handled for specific users based on their preferences stored in the directory (for example, a vacation message). And, it is the place where expansion of mail lists, groups, and aliases occurs. The MTA is typically where advanced processing gets done and how integration with third-party software packages such as virus scanning and antispam functionality occurs.

Mailstore

The basic function of the mailstore, sometimes incorrectly referred to as the mail server, is to store and send email to users via IMAP, POP, and web mail. The Sun ONE Messaging Server software is somewhat unique among the messaging systems on the market. Most messaging systems store email in a file system only or within a database only. The Messaging Server system offers a hybrid approach, storing messages in a file system, but also storing a copy of the header information (date, time, subject, sender, and so forth) in a database. This improves performance so that when users log in or sort email in their client, very little if any file system interaction is necessary. The information all comes from the database, which should be mostly in memory.

Mailstores can often be hundreds of gigabytes due to requirements for archiving and the volume and size of email messages these days, as outlined in the first chapter. It is not uncommon to find terabyte-sized mail stores, for example:

50,000 mailboxes * 20-megabyte quota = 1,000,000 megabytes = 1000 gigabytes = 1 terabyte

Proxy Servers

To allow for flexibility, redundancy, and abstraction and to provide a layer of security, organizations often utilize a concept known as proxy servers. They are fairly standard within the web server side of the infrastructure, but less so for messaging.

The Sun ONE Messaging Server software environment offers two proxy servers:

- messaging multiplexer proxy (MMP)
- messenger express multiplexer (MEM)

MMP proxies POP and IMAP connections, whereas the MEM off-loads the web mail client from the mailstore, so in a true sense MEM is a front-end, not a proxy.

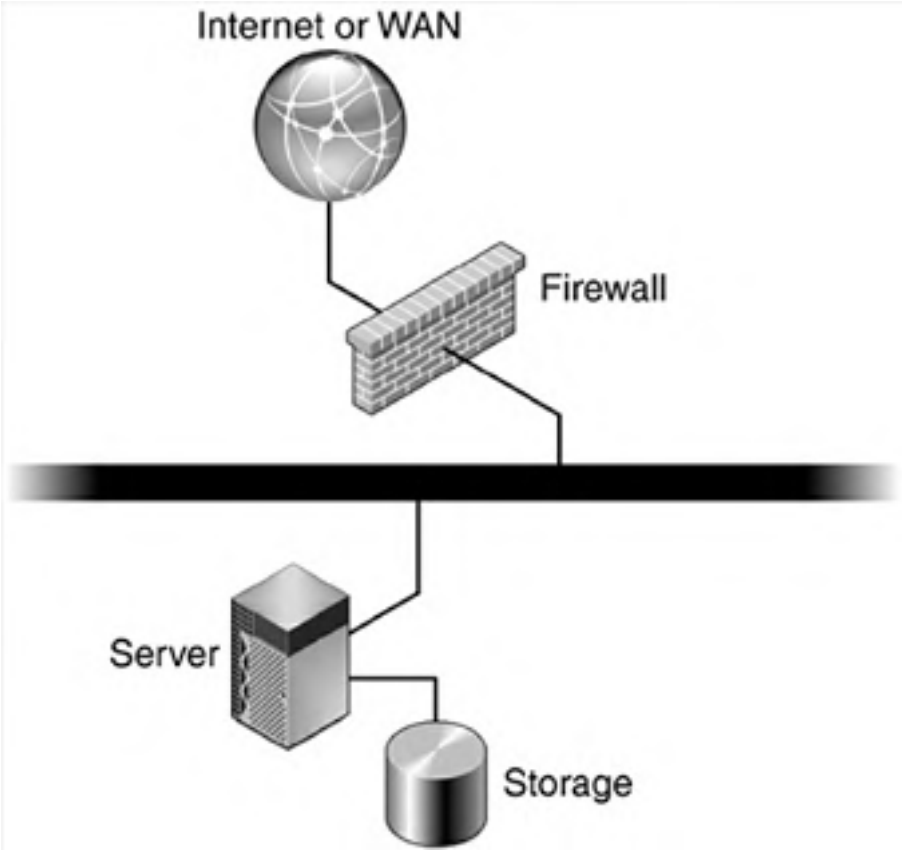
Why a proxy function? Well, in a large Internet Service Provider or in environments that have grown beyond a single mailstore, the proxy hides the fact you have multiple back-end servers, allowing a single client configuration (for example, smtp.company.com and imap.company.com) regardless of which mail server the user's physical inbox is on.

Sometimes network security requirements dictate the use of a proxy mechanism as well so the service, such as POP or IMAP, can be exposed while the content server (mailstore) is not.

Simple Single-Layer Architecture

In the simplest form, sometimes referred to as messaging in a box, all the components of the messaging server run on a single system (FIGURE 3-1) and no proxy is involved.

Figure 3-1. Messaging Server, Storage, and Firewall Messaging System



The components are:

- Directory
- MTA
- Mailstore

The benefits and drawbacks to this type of architecture are:

- Simple
- Easy to manage
- Easy to troubleshoot
- Low total cost of ownership (TCO)
- Limited scalability— This architecture is obviously limited in scalability to the computer system's size (CPU and memory) and operating system's scalability. Do not let this fool you into thinking that it cannot scale at all. In some messaging architectures, servers utilizing Sun hardware and the Solaris OE have scaled as high as 16 CPUs and are supporting thousands of concurrent users.

- No high availability— With everything in a single server, you have no redundancy beyond what is provided with that single system. You can get some availability through redundant components such as:
  - Power supplies
  - Network interfaces
  - CPU/Memory boards
  - RAID protected storage
- Security— Just because it is a simple configuration does not mean that it is entirely without security or that you cannot secure the system. Standard practices of turning off unused services, for example telnet, and replacing them with alternatives like ssh still apply, as does the use of firewall technology. However, without some form of relay or proxy for SMTP traffic from the Internet, this system will be accessible through SMTP directly from the Internet.

So, while a simple configuration is less secure than other configurations, it is not completely insecure. Overall this simple configuration tends to work for labs, training facilities, and very small systems where simplicity is the foremost requirement.

Simple—Alternative Architecture

The preceding configuration or architecture is very simple. One of the most common additions to the simple configuration is virus scanning in some manner. There are various methods of adding virus scanning to the messaging architecture including:

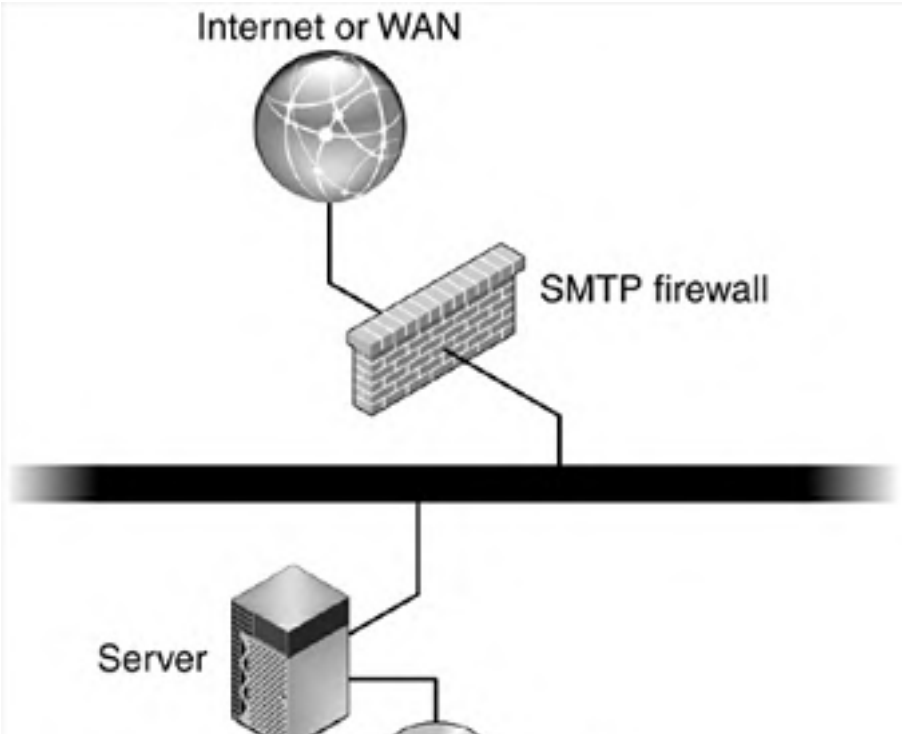
- Adding a virus appliance such as Borderware or Symantec
- Adding a virus firewall such as Trend Micro's VirusWall
- Adding virus scanning software on the messaging server itself. Each of these approaches has pros and cons.

As many organizations are well aware, relying only on desktop virus-scanning software does not eliminate all viruses for many reasons. Since viruses spread through email in addition to other methods, adding virus scanning to the messaging environment is a natural choice.

By combining a simple messaging install with an SMTP firewall product (FIGURE 3-2) offering antivirus (and potentially antispam) protection, the system accomplishes several things:

- Off-loads antivirus scanning from the messaging system— Often scanning takes significant processing power due to the requirement to examine all attachments as well as uncompressed attachments that are stored in compressed formats, such as zip files. It is not uncommon to have compressed files within compressed files. The level to which you scan is configurable, but each level takes more power.
- Isolates the Sun ONE Messaging Server from direct Internet access— Many hackers are well aware of exploits via SMTP and use the SMTP protocol to hack into people's networks or systems. By placing a firewall between the Internet and the mail server, a level of security is added. However, firewalls that offer SMTP relaying function are often not nearly as secure as the Sun ONE Messaging Server relay—careful consideration is required.
- Reduces the messaging workload— In addition to off-loading the antivirus and antispam workload, it also off-loads the rejection of email not destined for your messaging server.
- Maintains overall simplicity— Still maintains most of the benefits of simplicity while adding additional security.

Figure 3-2. Alternate Configuration With SMTP Firewall



Typically the main drawbacks of this configuration are:

- Added server requirement— The need to now manage two physical servers adds slightly more workload for the system administrator.
- Messaging headers— To scan all messages, sometimes messaging headers must be rewritten and forwarded to the scanning virus wall from the messaging server.
- Lack of flexibility— There are not a whole lot of optional configurations with a firewall and virus scanner in place; sometimes, this is the only such option available.
- Little, if any, redundancy— Since there is only one messaging system, there is no redundancy, or little beyond that which the single system provides (that is, RAID storage or redundant power supplies). Messages may or may not queue up on the virus wall server, depending upon its capabilities.

Although many sites use a virus firewall in front of the messaging server, there are disadvantages when putting another SMTP server in front of the Sun ONE Messaging Server's MTA as the outer most SMTP server in your organization. Here is a partial list of the major reasons:

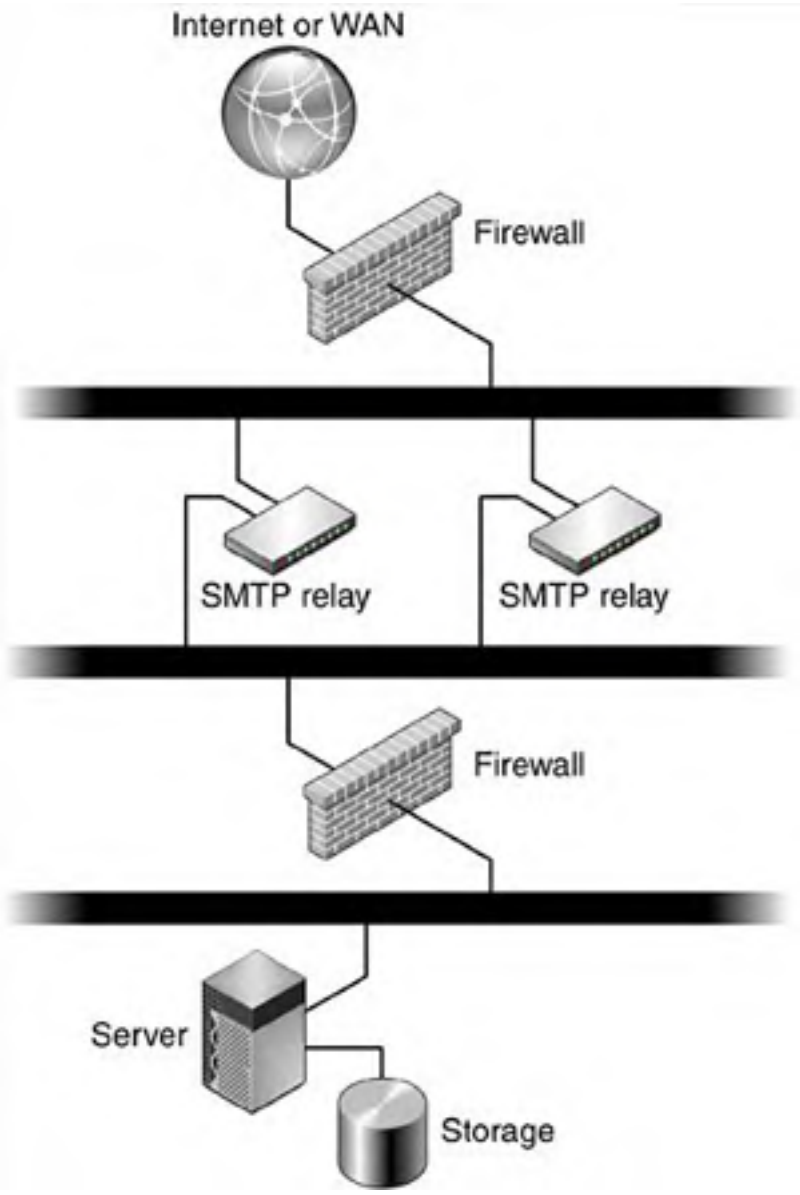
- First and foremost, the vendors specialize in virus filtering. They are not experts in MTA technology, so their SMTP server is basic and not as full featured as the Sun ONE Messaging Server
- Limited if any SMTP extensions support. Which means:
  - No SMTP AUTH
  - No NOTARY (for example, delivery receipt requests)
  - Deliver By (certain date)
  - Size-based extensions
  - Pipelining
  - SSL/TLS
- MIME support is minimal, no support for other messaging formats (for example, RFC1154, which is what Microsoft used before Exchange, NeXT Mail, BINHEX or UUENCODE)
- Limited if any realtime blackhole list (RBL) support
- Handling of very long header lines (a common technique to exploit buffer overflow errors in various mail clients)
- Tools for blocking mail based on various pieces of originator information
- No MMP
- Limited mail routing capabilities

However, for simplicity sake, many organizations still elect to use this alternative architecture.

Typical Architecture

A slightly more typical architecture (FIGURE 3-3) more commonly found adds a couple of SMTP relays to the simple configuration.

Figure 3-3. Alternate Configuration With SMTP Relays and Firewall



Often one relay is configured as inbound and the other outbound. These relays off-load the routing and rejecting of messaging. These relays can also run antivirus and antispam software. This configuration assumes that the only protocol coming from the Internet or going out to the Internet is SMTP. Users access the messaging system internally only, through a virtual private network (VPN) or through the firewall.

Combining a simple messaging installation with a pair of MTAs (SMTP routers) and a firewall accomplishes several things:

- Reduces routing workload for messaging— Some of the routing workload is being off-loaded, so messages destined for other mail servers internally or externally do not use the main messaging server.
- Isolates the messaging server from direct Internet access— Many hackers are well aware of exploits via SMTP

and use the SMTP protocol to hack into people's networks or systems. By placing a firewall between the Internet and the mail server, a level of security is added. By no means is this 100 percent secure, but it does add some security.

- Off-loads antivirus scanning from the messaging system— Antivirus scanners such as Sophos Sweep or Symantec for UNIX can be loaded and integrated with the MTA of the Sun ONE Messaging Server.

- Duplicate MTAs— While they are typically configured as one MTA, with one MTA handling inbound messages and the other handling outbound messages, they can be configured identically and used as redundant systems with one MTA handling (but not exclusively) inbound messages and the other primarily handling outbound messages. This is accomplished via round-robin DNS and mail eXchanger record (MX) configuration.

The main drawbacks of this configuration are:

- Added server requirements— The need to manage more physical servers adds more workload for the system administrator.

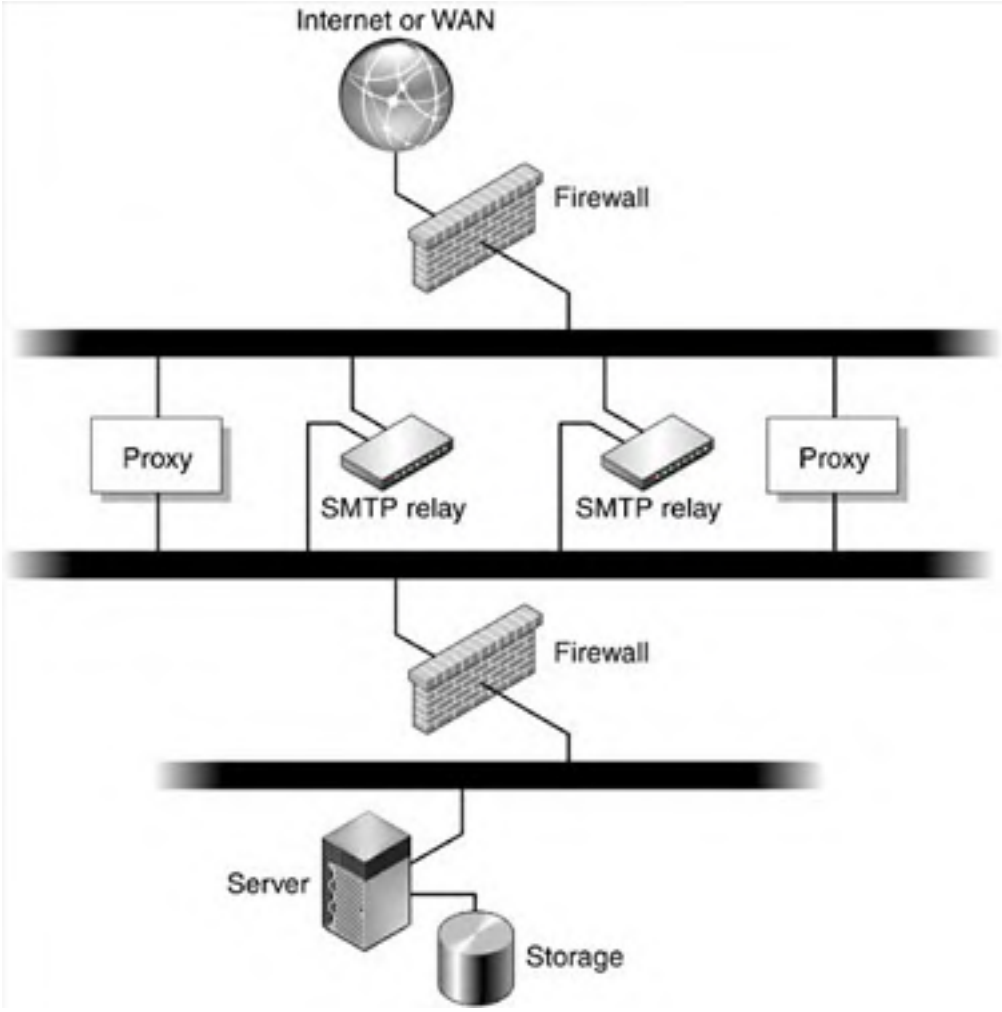
- The need to maintain two MTAs— The need to edit and maintain both MTAs and keep them configured and synchronized with one another adds some complexity.

- Little, if any, redundancy— Since there is only one messaging system, there is no redundancy, or little beyond that which the single system provides (that is, RAID storage or redundant power supplies). If one of the MTAs fails, messages will still queue up for delivery on the MTAs (for users) and outgoing messages will still get sent to the Internet, but no users will be able to read them.

Secure—Basic Architecture

This architecture (FIGURE 3-4) continues to build upon the typical architecture, adding the proxy servers for user access (that is, IMAP or POP).

Figure 3-4. Proxy Configuration With SMTP Relays and Firewall



The addition of these proxy servers extends the protocols through the firewall securely. Users must authenticate to these servers first, then they are proxied to the messaging server and only the messaging server.

Note

This configuration does not address all aspects of messaging security such as SSL, Secure Multipurpose Internet Mail Extensions (SMIME), or encrypted file system. Some of these methods are discussed in more detail later in this book. This architecture only addresses the physical and basic network layout.

Adding the proxy servers for IMAP, POP, and web mail:

- Extends the messaging server externally without requiring a virtual private network (VPN).
- Reduces routing workload for messaging— Some of the routing workload is being off-loaded, so messages destined for other mail servers internally or externally do not use the main messaging server.

- Provides duplicate MMP and MEM servers which adds redundancy— Using round-robin DNS or a network-based load balancer, redundancy for this type of server can be accomplished.
- Isolates messaging server from direct Internet access— Many hackers are well aware of exploits via SMTP and use the SMTP protocol to hack into people's networks or systems. By placing a firewall between the Internet and the mail server, a level of security is added. By no means is this 100 percent secure, but it does add some security.
- Off-loads antivirus scanning from the messaging system— Antivirus scanners such as Sophos Sweep or Symantec for UNIX can be loaded and integrated with the MTA of the Messaging Server.

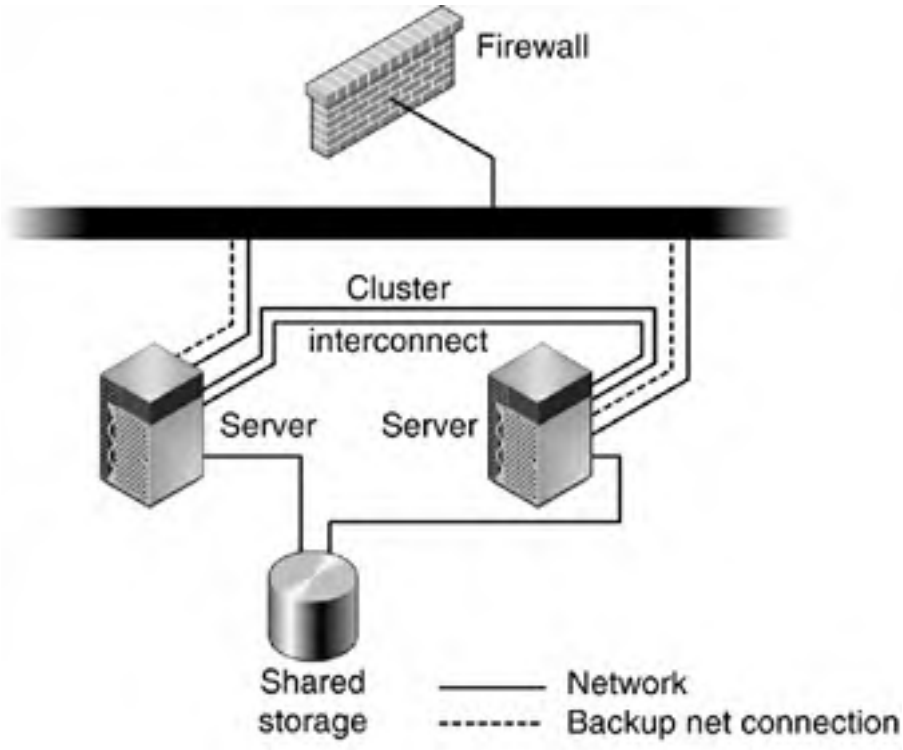
The main drawbacks of this configuration are:

- Added server requirements— The need to manage more physical servers adds more workload for the system administrator.
- Need to maintain two MTAs— The need to edit and maintain both MTAs and keep the configurations synchronized with one another adds some complexity.
- Additional firewall configuration required— Due to all the ports and servers, the firewall must be configured appropriately.
- Little, if any redundancy— since there is only one messaging system, there is no redundancy or little beyond that which the single system provides (that is, RAID storage or redundant power supplies). If one of the MTAs fails, messages will still queue up for delivery on the MTAs (for users) and outgoing messages will still get sent to the Internet, but no users will be able to read email. Web mail users will not have anything.

High Availability—Failover Architecture

One of the easiest and simplest ways of architecting a high-availability configuration is to cluster, using the Sun™ Cluster 3.0 software for example (FIGURE 3-5).

Figure 3-5. Simple Failover Configuration



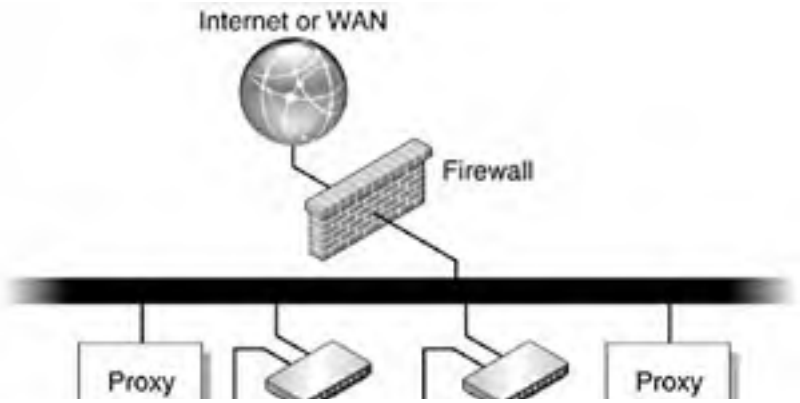
This architecture provides a highly available yet simple configuration. As you would expect, the benefits and drawbacks of this architecture closely mirror those of the simple configuration, with the exception being it is now highly available.

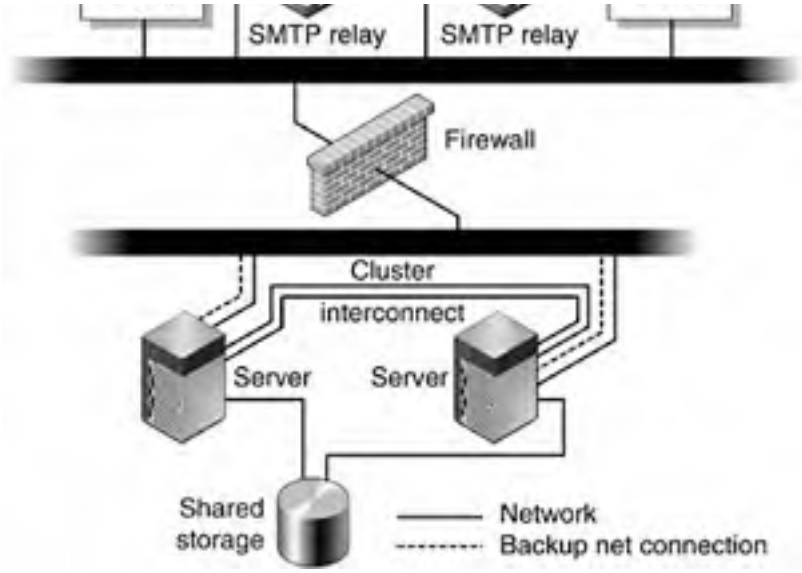
The clustering software not only handles hardware failures but in may cases, software issues as well, trying to restart failed daemons or processes first, then triggering a complete restart or failover of the entire system.

Failover itself can vary between minutes to as much as an hour depending upon the specific configuration, settings, and storage status. Should a catastrophic storage failure occur, parity and sanity checks on the storage subsystem can add significant to the normally fast (that is, less than 10 minutes) failover time.

A more complex high availability configuration (FIGURE 3-6) combines the portions of the secure architecture with that of the simple failover.

Figure 3-6. Failover With Relays and Firewall





By combining the failover of the *secure configuration* with parts of the *simple failover,* you can obtain availability for the routing (MTA), access (proxy), and mailstore, yet have the ability to extend services such as web mail, IMAP, and POP access to users external to the internal network.

One aspect of availability that will be addressed further in Chapter 14, "Highly Available Messaging Deployment," on page 201," is Directory Server availability. Currently there are two models for providing availability of LDAP (directory) services—the traditional failover using Sun Cluster or VERITAS products and the Multiple Master Replication feature of the Sun ONE Directory Server. Each has its own benefits and drawbacks, with many people looking at Multiple Master Replication as the new defacto method for addressing availability of the directory server.

Sun has a Reference Architecture program that defines the hardware and software components needed to build end-to-end solutions that meet specific business needs. Each Reference Architecture has been designed, tested, and documented, so users can reduce the complexity, costs, and risks of deploying new technology in their enterprises. Sun's Reference Architectures combine:

- A documented multitiered architecture
- Recommended technology products from Sun and other vendors
- Architecture, sizing, and implementation guides

For more details on the Messaging Reference Architecture, see:

http://www.sun.com/products/architectures-platforms/refarch/specs.html#g1_5.1.

◄ PREVIOUS NEXT ►

Chapter 4. Installation Preparation

To continue in further chapters, it is recommended that a functional messaging system be available to the reader for hands-on work. This chapter outlines some issues and practices that are important during the pre-installation. These issues can have significant impact on installation, operations, and recovery capability. It provides insight into situations that normally cause consternation. References are made to specific sections of manuals or additional supplemental materials. Think of this chapter as a reminder regarding operating system best practices that can be found in other BluePrints and elsewhere.

This chapter contains the following topics:

- Preparation Process
- Network Connectivity

Preparation Process

This section provides an overview of the preparation process. It covers the following topics:

- Good Computing Practices
- Differences Between Production and Non-production
- Basic Solaris OE Installation

Good Computing Practices

A very wise system engineer at Sun once gave me perhaps the best piece of advice ever: "Prior planning prevents poor performance."

The general idea is to start with a solid foundation (the Solaris OE) and build a solid structure (the Sun ONE Messaging Server) on it. If an incomplete or poor Solaris OE installation is done, do not expect applications such as the Messaging Server to operate correctly or efficiently. It all starts with good computing practices and preparation.

In some organizations, standards and practices regarding system installation and configuration exist. While most organizations agree that standards are a good thing, many simply do not go to this level of detail or effort. If your organization has such standards and practices, that is the place to start. Then incorporate or adjust (if absolutely necessary) anything specific to the messaging environment.

For those organizations that have no standards or perhaps only basic system administration knowledge, there are definitely things that will make the overall process smoother. This chapter outlines some of the basic issues that might interfere with getting the messaging system operational.

Differences Between Production and Non-production

There is a distinct difference between a production and a non-production environment or system. In many situations, it goes way beyond the issue of simply being able to reboot the system at will. Other issues such as change control procedures, security requirements, patching, disk layout, upgrades, and documentation are all different for a production environment than for a non-production environment. Standards and practices typically address these issues within an organization.

One issue that this chapter addresses specifically is the disk layout or partitioning. To simplify the process of creating a prototype or test system for messaging, the system hard drive configuration has only three partitions:

- swap has at least 256 megabytes minimum, up to 1x physical memory.
- / has four gigabytes or more.
- /export/home uses the remainder of the drive.

In a production environment, you would definitely create additional partitions to segregate specific data and applications (some of this is also discussed in Chapter 12, "Performance Tuning," on page 179. In addition to the preceding partitions, you would have partitions for the following functions:

- queues— Used to temporarily store the messages when routing
- store— Also referred to within the messaging server as a partition. There might be multiple stores, depending upon the number of mailboxes or specific policies.
- var— /var directory in the Solaris OE where logs and some temp files are stored
- usr— /usr directory in the Solaris OE
- logs— Sometimes depending upon specific needs or volume of logs, it is a good idea to have a separate volume or partition to write to.
- opt— Where optional program binaries are stored. Messaging software can be installed here if desired.

Your organization might have specific practices regarding disk layout. Start with this and modify or incorporate the messaging requirements.

Note

Do not take the AUTO LAYOUT defaults of the Solaris installation program. This will typically undersize the root and other partitions.

For some basic system (Solaris OE) install practices, the following resources are recommended:

- *Solaris System Administrator's Guide*, 3rd Edition, Janice Winsor
- Sun Blueprint book, *Operating Environment: Solaris 8 Installation and Boot Disk Layout*, March 2000, Richard Elling
- Sun Blueprint book *Configuring Boot Disks*, December 2001, John S. Howard and David Deeths
- Solaris Documentation Set

Basic Solaris OE Installation

Some advice regarding the initial Solaris OE installation that can prevent issues in the future:

- Install the latest update.
- Install the entire distribution.

The Solaris 8 OE will be installed on our prototype system. As a general practice, it is good to start with the most recent update. Currently, this is Solaris 8 OE 02/02. You could also install the latest Solaris 9 OE 04/03, however, some installation instructions may be slightly different. Using the latest release will reduce the amount of patching required.

One of the areas that can sometimes create problems is the specific installation of the Solaris OE. Those familiar with the overall install process know that the Solaris OE installation program provides five options when performing an interactive install:

- Entire distribution with OEM Support
- Entire distribution
- Developer
- End-user
- Custom

The mistake that is often made is to install something less than the entire distribution without specifically knowing that your applications will work correctly in this configuration. The entire distribution loads libraries and other files that are often required by applications for various reasons such as dynamic linking. Not having access to these libraries or files will create some interesting issues that may appear only after the program has been installed and operational for some time. So unless your organization has a specific standard regarding what is installed, use the entire distribution. If you begin to have problems or the software runs strangely (sometimes things like character sets missing can cause issues), investigate which load of the Solaris OE was installed. For the demo or prototype system, the Entire Distribution of Solaris 8 OE 02/02 is used.

For details regarding installing the Solaris OE, refer to the *Solaris Installation Guide*.

Now that Solaris is installed, the next step is to patch the system. Begin by downloading the latest Solaris Recommended Patch Cluster for the specific version of the Solaris OE from the SunSolve^{SM} web site at:

http://sunsolve.sun.com.

In our case, the Solaris 8 Patch cluster contains the latest security and recommended patches for Solaris 8 OE. Follow the installation instructions provided with the Solaris Recommended Patch Cluster.

Now check the Release Notes for the Sun ONE Messaging Server, or any other application that you plan on installing, for any additional required patches that may not be included as part of the Solaris Recommended Patch Cluster. You can retrieve individual patches from the SunSolve web site as well. These patches are generally available, however, in some rare situations they may only be available to customers with support contracts.

Now is the time to consider how best to maintain your system patch level. Several tools or utilities can help you do this:

- Patch Manager— Automates patch management and patch analysis accuracy. Provides configuration-specific patch analysis, automated patch download, patch dependency resolution, and install. Available for Solaris OE 2.6 through Solaris OE 9, Sun Cluster, Network Storage, Sun Enterprise™ 10000, and Sun Fire™ systems.

- PatchPro/PatchPro Expert— PatchPro Interactive generates a custom patch list that can be downloaded in a single tar file. This file is based on selections of various Sun hardware and software products. PatchPro Expert is a signed applet that analyzes your system and generates a custom patch list. The applet will attempt to detect software in all categories listed for PatchPro Interactive. PatchPro Expert requires a Java-enabled Netscape™ browser. Available for Solaris OE 2.6 through Solaris OE 9.

- PatchCheck— Replaces PatchDiag. Determines the patch levels on your system against Sun's Recommended and Security patch list. Additionally, it operates from input files and lists all patches that pertain to packages installed on the system. This tool is similar to the PatchDiag Tool that you may have used in the past, with the added advantage of producing reports in HTML format that allow you to select and receive your desired patches.

Why go to this degree of effort to patch the system?

In many cases, this will be the only chance to patch to this degree of thoroughness. Once a system is in production, it becomes more difficult to obtain maintenance windows.

There are several Sun technologies to assist an organization in installation and management of the Solaris OE:

- JumpStart— The JumpStart™ system is useful for much more than installing the Solaris OE. Solaris JumpStart is an automatic installation (auto-install) process available in the Solaris OE and comes free with Solaris. It allows system administrators to categorize machines on their network, and automatically installs systems based on the category (Class) to which a machine belongs. In many ways, JumpStart is similar to the RedHat Linux KickStart functionality.

  The JumpStart system is like a scripting language; the JumpStart framework provides a toolkit of operators that can be used individually or combined. These operators function well individually, but their true power is realized when they are combined.

  You can even perform JumpStart over a wide area network (WAN) in the newer versions of the Solaris OE. With the boot command, you can specify the location of the JumpStart profile and sysidcfg information to use to perform the installation. You can specify a path to an HTTP server, an NFS server, or a file that is available on local media.

  For a complete list of Sun BluePrints on the Jumpstart Flash Archive, see:

  http://www.sun.com/solutions/blueprints/browsesubject.html#jumpstart.

- Solaris Flash— The Solaris Flash feature provides new installation and provisioning functionality. System administrators can capture a snapshot image of a complete server—including the Solaris OE, the applications stack, and the system configuration—into a new Flash Archive format. Using this system image, administrators can then replicate reference server configurations onto multiple servers or cloned. Solaris Flash images can be deployed using standard media or over the network via HTTP and NFS. Solaris Flash images can be installed using custom Solaris JumpStart scripts, the Solaris Web Start graphical interface, or Solaris OE interactive installation.

  Solaris Flash technology provides the ability to layer Flash Archives. You can create partial Flash Archives to install in a variety of ways. This feature increases the flexibility for rapid modular deployment.

  For example, you can create one archive that contains the Solaris OE files, a second archive that contains the files necessary to run a Web server, and a third archive that contains the files for an NFS server. You can then install the first and second archives to one machine to create a web server, and the first and third archives to create an NFS server.

  For more details, see:

  http://wwws.sun.com/software/solaris/webstartflash/ and
  http://wwws.sun.com/software/whitepapers/wpsolarisinst/solaris_installation_deployment.pdf.

- Change Manager— Change Manager is part of the Sun™ Management Center product family. Change Manager

is a provisioning and change management software product that delivers a fast and easy way to install, configure, update, provision, and audit the software stacks running on Sun systems. It can significantly improve IT staff efficiency and productivity in a computing environment that relies on replicated servers to provide software services. Change Manager software utilizes Solaris Flash, Solaris Live Upgrade, and Solaris JumpStart technologies to provision servers. It can leverage existing Jumpstart scripts and Flash Archive files to a degree.

Change Manager works on Solaris 8 OE platform 2/02 or later as well as the Solaris 9 OE. It is not bundled with the Solaris OE, but rather is a separate package available for purchase.

For more details, see:

http://wwws.sun.com/software/solaris/sunmanagementcenter/ds/ds-smccm/index.html.

- N1™ for Blades— Due to their specific nature, Blade servers are somewhat different than other types of servers when it comes to provisioning the operating environment and configuring the underlying hardware. Sun recently introduced the N1 Provisioning Server 3.0 Blades Edition software to address the provisioning issue for their Blade servers. This software provides a powerful management environment for Sun Fire Blades and Shelves. Running on one or more dedicated servers, this software performs many of its management functions through an out-of-band management network. The N1 Provisioning Server 3.0 Blades Edition software enables system administrators to rapidly design, configure, provision, and scale blade-based logical server farms automatically. The software manages this pool, along with other networking resources to quickly reconfigure, deploy, and decommission large collections of blades whether they are in the same data center or are geographically dispersed.

  As the required testing and certification of the Sun ONE Messaging Server and related software take place on these Sun Fire Blade servers, the N1 Provisioning Server 3.0 Blades Edition software will undoubtedly be very useful.

  For more details, see:

  http://wwws.sun.com/software/products/provisioning_server/.

Network Connectivity

Some quick caveats regarding network connectivity. It goes without saying that network connectivity is required. There are, however, a couple of items that can cause issues or confusion in this area, either during the installation of the operating system or during the installation of the Messaging Server.

The issues tend to fall into one of five areas within networking:

- Host Name Resolution With /etc/hosts and DNS
- Naming Services Setup and Best Practices
- Network Load Balancing
- DHCP
- Domain Name

Host Name Resolution With /etc/hosts and DNS

You can do two things to avoid problems.

First, put all critical hosts that are absolutely, positively required for operation into the /etc/hosts file. Second, put the fully qualified host name (FQN) in the /etc/hosts file in addition to the short name.

During some parts of the installation process, the Messaging Server installation program needs fully qualified host names. If it cannot resolve the FQN, manual entry is required. Unfortunately humans are not as accurate as computers, so errors can occur. Following this advice allows the installation program to automatically obtain the FQN directly.

Example:

```
root@sparc5-1# cat /etc/hosts
#
# Internet host table
#
127.0.0.1 localhost loghost
10.0.0.171 demo   demo.test.sun.com
# Directory Server node
#
10.0.0.172 ldap   ldap.test.sun.com
# SunCluster nodes
#
10.0.0.173 node0  node0.test.sun.com
10.0.0.174 node1  node1.test.sun.com
```

This example provides for resolution of both the fully qualified and non-qualified names if the normal naming service, such as NIS, XX NIS+, or DNS becomes unavailable or slow. This can prevent some outages or problems. Your organization's standards may dictate otherwise—after all, this does add some maintenance over relying upon the naming service for everything.

Naming Services Setup and Best Practices

A number of options for naming services are available, including /etc/hosts, DNS, NIS, NIS+, and LDAP. The key here is to make sure that any naming service being used is accurate and available. If you use /etc/hosts, you must maintain it and keep it up to date. If you are using DNS, you must make sure that it is properly configured and is functional for each one of the name servers listed in the /etc/resolv.conf file. Why? If you simply test that the DNS is working properly, you will only be testing that the server listed first resolves correctly, not the second or other servers listed:

```
root@sparc5-1# cat /etc/resolv.conf
domainname test.sun.com
nameserver 10.0.62.1
nameserver 10.0.62.14
root@sparc5-1#
```

So in the event of an issue with 10.0.62.1, 10.0.62.14 will be used. However, if we never tested 10.0.62.14, it may not work either.

Network Load Balancing

Network-based load balancing allows for failover of IP addresses and services. This can be done several ways:

- Round-robin DNS feature of newer revisions of DNS (bind).
- Layer 3/4 switches— Alteon is an example.
- Software— Resonate is an example.

There are pros and cons to each of these ways, however. While round-robin DNS is inexpensive, it also takes several minutes or longer to fail over. Layer 3/4 switches and software operate quickly, within seconds or even faster, but they are expensive. Depending on your availability requirements and budget, you must begin planning your networking availability strategy up front. Otherwise, this will impact the installation and configuration of the messaging systems. Renaming and re-addressing the systems is not a simple task.

DHCP

The Dynamic Host Configuration Protocol (DHCP) for a server running the Messaging Software is not supported. There are places within the software that the IP address of the server, for security reasons, is coded.

Domain Name

It is generally a good idea to set or configure the default domain name, either through the command line or the /etc/defaultdomain. Not having this set can sometimes cause problems. However, the default does not have to match up to any specific domain name of the Messaging Server, DNS, or Network Information Service (NIS).

Chapter 5. System Startup

This chapter covers the basics of getting the system started and provisioning users once the system is operational. It is designed to provide you an understanding of the various mechanisms for provisioning as well as the pros and cons of each method. You can easily automate provisioning, but there are times when manual entry is required too.

First, this chapter reviews the daemons running on a test or demo system so you can see what a correctly installed system should look like from top or ps -ef commands and utilities. Then, the chapter describes the various options for administration of the system, including provisioning. Some specific administrative actions are only available using one method or another. There are others you really should do one way and *not* another. A simple example of provisioning accounts and users by using a Perl script is provided, plus a script for generating test users.

This chapter covers the following topics:

- To check on the status of the test system installed for this book, use the ps -ef command: (should be operational)
- Provisioning
- Sample Data File
- Sample Provisioning Script
- Test User Generation Script

Basic System Status

First, the system should be installed and operational. If you had difficulties or are unsure that the system is operational, there are several ways to check the system status using either the start and stop scripts from the previous chapter or more familiar UNIX commands and utilities such as top and ps -ef.

To check on the status of the test system installed for this book, use the ps -ef command:

Code Example 5-1 ps -ef Command Output

[View full width]

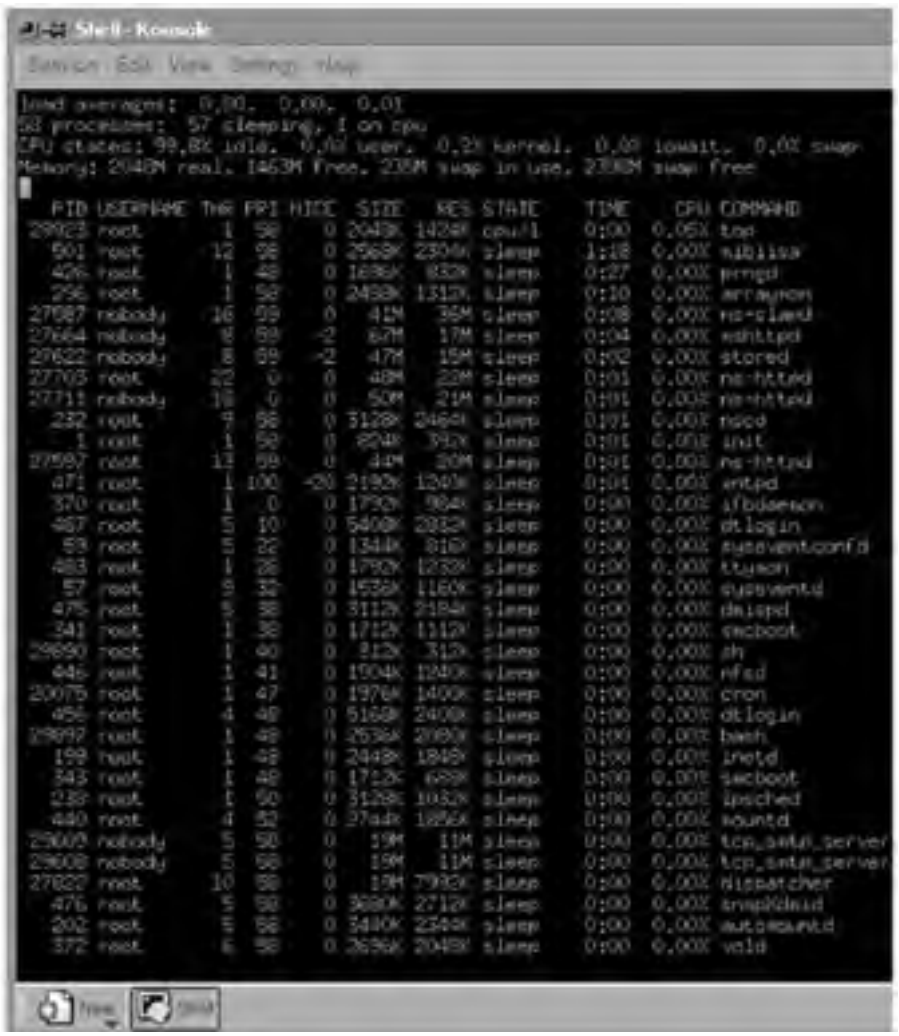
```
UID   PID  PPID  C    STIME TTY      TIME CMD
  root     0     0  0   Apr 11 ?       0:01 sched
  root     1     0  0   Apr 11 ?       0:01 /etc/init -
  root     2     0  0   Apr 11 ?       0:00 pageout
  root     3     0  0   Apr 11 ?      35:08 fsflush
  root   482     1  0   Apr 11 ?       0:00 /usr/lib/saf/sac -t 300
  root   487   456  0   Apr 11 ?       0:00 /usr/dt/bin/dtlogin -daemon
  root   176     1  0   Apr 11 ?       0:00 /usr/sbin/rpcbind
  root    57     1  0   Apr 11 ?       0:00 /usr/lib/sysevent/syseventd
  root    59     1  0   Apr 11 ?       0:00 /usr/lib/sysevent/syseventconfd
  root   202     1  0   Apr 11 ?       0:01 /usr/lib/autofs/automountd
  root   343   341  0   Apr 11 ?       0:00 /usr/sadm/lib/smc/bin/smcboot
  root   217     1  0   Apr 11 ?       0:00 /usr/sbin/syslogd
  root   232     1  0   Apr 11 ?       0:02 /usr/sbin/nscd
  root   238     1  0   Apr 11 ?       0:00 /usr/lib/lpsched
  root   486   456  0   Apr 11 ?       0:01 /usr/openwin/bin/Xsun :0 -nobanner -auth /var/dt
➥ /A:0-lcaa5a
  root 27697     1  0   Apr 14 ?       0:00 /A1000/demo6789/ims52/bin/msg/admin/bin/enpd
  root   476     1  0   Apr 11 ?       0:00 /usr/lib/dmi/snmpXdmid -s sparc5-1
nobody 27822     1  0   Apr 14 ?       0:00 /A1000/demo6789/ims52/bin/msg/imta/bin/dispatcher
  root   370     1  0   Apr 11 ?       0:00 /usr/sbin/ifbdaemon /dev/fbs/ifb0
  root   251     1  0   Apr 11 ?       0:00 /usr/lib/utmpd
  root   199     1  0   Apr 11 ?       0:00 /usr/sbin/inetd -s
  root   296     1  0   Apr 11 ?       0:10 /usr/lib/osa/bin/arraymon
  root   311     1  0   Apr 11 ?       0:00 /usr/lib/osa/bin/sparcv9/rdaemon 29 203 5
  root   372     1  0   Apr 11 ?       0:00 /usr/sbin/vold
  root   446     1  0   Apr 11 ?       0:00 /usr/lib/nfs/nfsd -a 16
  root   485   482  0   Apr 11 ?       0:00 /usr/lib/saf/ttymon
  root   456     1  0   Apr 11 ?       0:00 /usr/dt/bin/dtlogin -daemon
  root   340   311  0   Apr 11 ?       0:00 /usr/lib/osa/bin/sparcv9/rdaemon 29 203 5
  root   341     1  0   Apr 11 ?       0:00 /usr/sadm/lib/smc/bin/smcboot
  root   466     1  0   Apr 11 ?       0:00 /usr/lib/snmp/snmpdx -y -c /etc/snmp/conf
  root   483     1  0   Apr 11 console  0:00 /usr/lib/saf/ttymon -g -h -p sparc5-1 console
➥  login:  -T sun -d /dev/console -l
  root   440     1  0   Apr 11 ?       0:00 /usr/lib/nfs/mountd
  root   471     1  0   Apr 11 ?       0:02 /usr/lib/inet/xntpd
  root   475     1  0   Apr 11 ?       0:00 /usr/lib/dmi/dmispd
  root   426     1  0   Apr 11 ?       0:28 /usr/local/sbin/prngd /var/spool/prngd/pool
  root   489     1  0   Apr 11 ?       0:00 /usr/openwin/bin/fbconsole -d :0
  root   502   487  0   Apr 11 ?       0:00 dtgreet -display :0
  root   501   466  0   Apr 11 ?       1:18 mibiisa -r -p 32787
  root 27601     1  0   Apr 14 ?       0:00 ./ns-admin -d /A1000/demo6789/ims52
➥ /admin-serv/config
nobody 27664     1  0   Apr 14 ?       0:04 /A1000/demo6789/ims52/bin/msg/store/bin
➥ /mshttpd -d 5 -D 6
nobody 27821     1  0   Apr 14 ?       0:00 /A1000/demo6789/ims52/bin/msg/imta/bin
➥ /job_controller
nobody 27635     1  0   Apr 14 ?       0:01 /A1000/demo6789/ims52/bin/msg/store/bin/popd -d 5
  root 29899 29897  0 13:27:35 pts/3    0:00 ps -ef
  root 27709     1  0   Apr 14 ?       0:00 ./uxwdog -d /A1000/demo6789/iws60
➥ /https-sparc5-1.central.sun.com/config
  root 27700     1  0   Apr 14 ?       0:00 ./uxwdog -d /A1000/demo6789/iws60
➥ /https-admserv/config
nobody 27622     1  0   Apr 14 ?       0:02 /A1000/demo6789/ims52/bin/msg/admin/bin/stored -d
nobody 29608     1  0 09:32:28 ?       0:00 /A1000/demo6789/ims52/bin/msg/imta/bin
➥ /tcp_smtp_server
  root 27710 27709  0   Apr 14 ?       0:00 ns-httpd -d /A1000/demo6789/iws60
➥ /https-sparc5-1.central.sun.com/config
```

```
 root 29890 29888  0 13:27:21 pts/3    0:00 -sh
UID   PID  PPID  C    STIME TTY      TIME CMD
 root 29888   199  0 13:27:21 ?        0:00 in.telnetd
 root 29897 29890  0 13:27:24 pts/3    0:00 bash
 root 27594     1  0   Apr 14 ?        0:00 ./uxwdog -d /A1000/demo6789/ids51/admin-serv
/config
nobody 27711 27710  0   Apr 14 ?        0:01 ns-httpd -d /A1000/demo6789/iws60
/https-sparc5-1.central.sun.com/config
nobody 27649     1  0   Apr 14 ?        0:00 /A1000/demo6789/ims52/bin/msg/store/bin/imapd
 -d 5 -D 6
 root 27595 27594  0   Apr 14 ?        0:01 ns-httpd -d /A1000/demo6789/ids51/admin-serv
/config
 root 27703 27701  0   Apr 14 ?        0:01 ns-httpd -d /A1000/demo6789/iws60
/https-admserv/config
 root 20075     1  0   Apr 13 ?        0:00 /usr/sbin/cron
 root 27597 27595  0   Apr 14 ?        0:01 ns-httpd -d /A1000/demo6789/ids51/admin-serv
/config
nobody 29609     1  0 09:32:28 ?        0:00 /A1000/demo6789/ims52/bin/msg/imta/bin
/tcp_smtp_server
nobody 27587     1  0   Apr 14 ?        0:09 ./ns-slapd -D /A1000/demo6789/ids51
/slapd-sparc5-1 -i /A1000/demo6789/ids51/sla
 root 27701 27700  0   Apr 14 ?        0:00 ns-httpd -d /A1000/demo6789/iws60
/https-admserv/config]
```

As you can see, several things are running on the system in addition to the Messaging Server. You could filter the results looking only for the messaging components by user or group, but that requires that you know that all the daemons are properly installed and operating as the messaging user or group. For diagnostics, sometimes it is better to review everything running.

Another UNIX utility that you can use is top, which presents a nicer view of the processes running on the system as well as the ability to sort and organize the output (FIGURE 5-1).

Figure 5-1. top Command Output



Regardless of the method used to view the current processes on the system, you should have the following daemons running at this point when everything is installed on the same server:

- mshttpd— Web mail daemon
- stored— Mailstore daemon
- ns-slapd— LDAP daemon
- ns-httpd— web server for delegated administration and administration servers; you should have three
- enpd— event notification daemon
- dispatcher— Dispatcher
- tcp_smtp_server— SMTP daemon
- job_controller— Job controller daemon
- popd— POP daemon
- imapd— IMAP daemon

If the directory server were running on a separate system, you would not have the ns-slapd daemon running on this server or one of the ns-httpd for the Directory Administrator. If you turned off web mail, the ms-httpd daemon would not be running, and so forth. So you really must know the specifics of your installation, otherwise you will think something should be there that is not, or vice versa. Since we installed everything on the same server, all the daemons appear.

Keep in mind that this is a default installation on a small server. For larger, production servers, it is possible to have multiple daemons running too. For example, depending on the configuration parameters, there might be multiple ms-httpd daemons to support many web mail users.

To get a list of the current configuration settings, execute the following command:

```
# configutil
```

This command lists the current configuration of the Messaging Server, which includes information such as port numbers, protocol status (off/on), log file location, process settings, and so forth. It is a good idea to maintain an archive or repository of this information so when problems or issues arise, the current settings can be compared with the last known good settings. Print these settings out for future reference as you go through the remaining sections and exercises in this book.

Code Example 5-2 configutil Output—Current Configuration Settings

[View full width]

```
alarm.createtimestamp = 20030414042706Z
alarm.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
alarm.diskavail.createtimestamp = 20030414042706Z
alarm.diskavail.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement
 ,o=netscaperoot"
alarm.diskavail.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
 ,o=netscaperoot"
alarm.diskavail.modifytimestamp = 20030414042706Z
alarm.diskavail.msgalarmdescription = "percentage mail partition diskspace available"
alarm.diskavail.msgalarmstatinterval = 3600
alarm.diskavail.msgalarmthreshold = 10
alarm.diskavail.msgalarmthresholddirection = -1
alarm.diskavail.msgalarmwarninginterval = 24
alarm.diskavail.objectclass = nsmsgCfgAlarm,top
alarm.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
alarm.modifytimestamp = 20030414042706Z
alarm.msgalarmnoticeport = 25
alarm.msgalarmnoticercpt = postmaster
alarm.msgalarmnoticesender = postmaster
alarm.objectclass = nsmsgCfgAlarmContainer ,top
alarm.serverresponse.createtimestamp = 20030414042706Z
alarm.serverresponse.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement
 ,o=netscaperoot"
```

alarm.serverresponse.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
alarm.serverresponse.modifytimestamp = 20030414042706Z
alarm.serverresponse.msgalarmdescription = "server response time in seconds"
alarm.serverresponse.msgalarmstatinterval = 600
alarm.createtimestamp = 20030414042706Z
alarm.serverresponse.msgalarmthreshold = 10
alarm.serverresponse.msgalarmthresholddirection = 1
alarm.serverresponse.msgalarmwarninginterval = 24
alarm.serverresponse.objectclass = nsmsgCfgAlarm ,top
createtimestamp = 20030414042706Z
creatorsname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
encryption.createtimestamp = 20030414042706Z
encryption.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
encryption.fortezza.createtimestamp = 20030414042706Z
encryption.fortezza.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
encryption.fortezza.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
encryption.fortezza.modifytimestamp = 20030414042706Z
encryption.fortezza.nssslactivation = off
encryption.fortezza.objectclass = nsEncryptionModule,top
encryption.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
encryption.modifytimestamp = 20030414042706Z
encryption.nscertfile = alias/msg-sparc5-1-cert7.db
encryption.nskeyfile = alias/msg-sparc5-1-key3.db
encryption.nsssl2 = off
encryption.nsssl3 = on
encryption.nsssl3ciphers = rsa_rc4_40_md5
,rsa_rc2_40_md5
,rsa_des_sha
,rsa_rc4_128_md5
,rsa_3des_sha
encryption.nsssl3sessiontimeout = 0
encryption.nssslclientauth = 0
encryption.nssslsessiontimeout = 0
encryption.objectclass = nsEncryptionConfig,top
encryption.rsa.createtimestamp = 20030414042707Z
encryption.rsa.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
encryption.rsa.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
encryption.rsa.modifytimestamp = 20030414042707Z
encryption.rsa.nssslactivation = on
encryption.rsa.nssslpersonalityssl = Server-Cert
encryption.rsa.nsssltoken = internal
encryption.rsa.objectclass = nsEncryptionModule,top
gen.accounturl = http://%U@sparc5-1.central.sun.com:55555/bin/user/admin/bin/enduser
gen.configversion = 4.0
gen.createtimestamp = 20030414042707Z
gen.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
gen.folderurl = http://%U@sparc5-1.central.sun.com:55555/bin/user/admin/bin/mailacl
.cgi?folder=%M
gen.installedlanguages = en
gen.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
gen.modifytimestamp = 20030414042707Z
gen.objectclass = nsmsgCfgGen,top
gen.sitelanguage = en
local.defdomain = sparc5-1.central.sun.com
local.enduseradmincred = }3:0R77?xB
local.enduseradmindn = "uid=msg-admin-sparc5-1.central.sun.com-20020710153937, ou=People,
o=sparc5-1.central.sun.com, o=isp"
local.hostname = sparc5-1.central.sun.com
local.imta.imta_tailor = /A1000/demo6789/ims52/msg-sparc5-1/imta/config/imta_tailor
local.imta.ssrenabled = yes
local.installeddir = /A1000/demo6789/ims52/bin/msg
alarm.createtimestamp = 20030414042706Z
local.instancedir = /A1000/demo6789/ims52/msg-sparc5-1
local.lastconfigfetch = 1050433755
local.ldapbasedn = o=NetscapeRoot
local.ldapcachefile = /A1000/demo6789/ims52/msg-sparc5-1/config/local.conf
local.ldaphost = sparc5-1.central.sun.com
local.ldapport = 389
local.ldapsiecred = VCk3UUl38W
local.ldapsiedn = "cn=msg-sparc5-1, cn=iPlanet Messaging Suite, cn=Server Group (2),
cn=sparc5-1.central.sun.com, ou=sparc5-1.central.sun.com, o=NetscapeRoot"

local.ldapusessl = False
local.servergid = nobody
local.servername = sparc5-1
local.serverroot = /A1000/demo6789/ims52
local.servertype = msg
local.serveruid = nobody
local.service.pab.attributelist = pabattrs
local.service.pab.enabled = 1
local.service.pab.ldapbasedn = o=pab
local.service.pab.ldapbinddn = "uid=msg-admin-sparc5-1.central.sun.com-20020710153937,
ou=People, o=sparc5-1.central.sun.com, o=isp"
local.service.pab.ldaphost = sparc5-1.central.sun.com
local.service.pab.ldappasswd = }3:0R77?xB
local.service.pab.ldapport = 389
local.service.pab.maxnumberofentries = 500
local.supportedlanguages = "[en,de,fr,es,af,ca,da,nl,fi,gl,ga,is,it,no,pt,sv,ja,ko,zh-CN
,zh-TW]"
local.tmpdir = /A1000/demo6789/ims52/msg-sparc5-1/tmp
local.ugldapbasedn = o=isp
local.ugldapbindcred = }3:0R77?xB
local.ugldapbinddn = "uid=msg-admin-sparc5-1.central.sun.com-20020710153937, ou=People,
o=sparc5-1.central.sun.com, o=isp"
local.ugldapdeforgdn = "o=sparc5-1.central.sun.com, o=isp"
local.ugldaphost = sparc5-1.central.sun.com
local.ugldapport = 389
local.ugldapuselocal = yes
local.webmail.da.host = sparc5-1.central.sun.com
local.webmail.da.port = 88
local.webmail.sso.enable = 0
local.webmail.sso.singlesignoff = 0
logfile.admin.buffersize = 0
logfile.admin.createtimestamp = 20030414042707Z
logfile.admin.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
logfile.admin.expirytime = 604800
logfile.admin.flushinterval = 60
logfile.admin.logdir = /A1000/demo6789/ims52/msg-sparc5-1/log/admin
logfile.admin.loglevel = Notice
logfile.admin.logtype = NscpLog
logfile.admin.maxlogfiles = 10
logfile.admin.maxlogfilesize = 2097152
logfile.admin.maxlogsize = 20971520
logfile.admin.minfreediskspace = 5242880
logfile.admin.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
logfile.admin.modifytimestamp = 20030414042707Z
logfile.admin.objectclass = nsmsgCfgLog ,top
logfile.admin.rollovertime = 86400
logfile.createtimestamp = 20030414042707Z
logfile.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
alarm.createtimestamp = 20030414042706Z
logfile.default.buffersize = 0
logfile.default.createtimestamp = 20030414042707Z
logfile.default.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
logfile.default.expirytime = 604800
logfile.default.flushinterval = 60
logfile.default.logdir = /A1000/demo6789/ims52/msg-sparc5-1/log/default
logfile.default.loglevel = Notice
logfile.default.logtype = NscpLog
logfile.default.maxlogfiles = 10
logfile.default.maxlogfilesize = 2097152
logfile.default.maxlogsize = 20971520
logfile.default.minfreediskspace = 5242880
logfile.default.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
logfile.default.modifytimestamp = 20030414042707Z
logfile.default.objectclass = nsmsgCfgLog ,top
logfile.default.rollovertime = 86400
logfile.http.buffersize = 0
logfile.http.createtimestamp = 20030414042710Z
logfile.http.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
logfile.http.expirytime = 604800
logfile.http.flushinterval = 60
logfile.http.logdir = /A1000/demo6789/ims52/msg-sparc5-1/log/http
logfile.http.loglevel = Notice
logfile.http.logtype = NscpLog
logfile.http.maxlogfiles = 10

```
logfile.http.maxlogfilesize = 2097152
logfile.http.maxlogsize = 20971520
logfile.http.minfreediskspace = 5242880
logfile.http.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
logfile.http.modifytimestamp = 20030414042710Z
logfile.http.objectclass = nsmsgCfgLog ,top
logfile.http.rollovertime = 86400
logfile.imap.buffersize = 0
logfile.imap.createtimestamp = 20030414042710Z
logfile.imap.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
logfile.imap.expirytime = 604800
logfile.imap.flushinterval = 60
logfile.imap.logdir = /A1000/demo6789/ims52/msg-sparc5-1/log/imap
logfile.imap.loglevel = Notice
logfile.imap.logtype = NscpLog
logfile.imap.maxlogfiles = 10
logfile.imap.maxlogfilesize = 2097152
logfile.imap.maxlogsize = 20971520
logfile.imap.minfreediskspace = 5242880
logfile.imap.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
logfile.imap.modifytimestamp = 20030414042710Z
logfile.imap.objectclass = nsmsgCfgLog ,top
logfile.imap.rollovertime = 86400
logfile.imta.buffersize = 0
logfile.imta.createtimestamp = 20030414042710Z
logfile.imta.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
logfile.imta.expirytime = 604800
logfile.imta.flushinterval = 60
logfile.imta.logdir = /A1000/demo6789/ims52/msg-sparc5-1/log/imta
logfile.imta.loglevel = Notice
logfile.imta.logtype = NscpLog
alarm.createtimestamp = 20030414042706Z
logfile.imta.maxlogfiles = 10
logfile.imta.maxlogfilesize = 2097152
logfile.imta.maxlogsize = 20971520
logfile.imta.minfreediskspace = 5242880
logfile.imta.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
logfile.imta.modifytimestamp = 20030414042710Z
logfile.imta.objectclass = nsmsgCfgLog ,top
logfile.imta.rollovertime = 86400
logfile.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
logfile.modifytimestamp = 20030414042707Z
logfile.objectclass = nsmsgCfgContainer ,top
logfile.pop.buffersize = 0
logfile.pop.createtimestamp = 20030414042710Z
logfile.pop.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
logfile.pop.expirytime = 604800
logfile.pop.flushinterval = 60
logfile.pop.logdir = /A1000/demo6789/ims52/msg-sparc5-1/log/pop
logfile.pop.loglevel = Notice
logfile.pop.logtype = NscpLog
logfile.pop.maxlogfiles = 10
logfile.pop.maxlogfilesize = 2097152
logfile.pop.maxlogsize = 20971520
logfile.pop.minfreediskspace = 5242880
logfile.pop.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
logfile.pop.modifytimestamp = 20030414042710Z
logfile.pop.objectclass = nsmsgCfgLog ,top
logfile.pop.rollovertime = 86400
logfiles.admin.alias = |logfile|admin
logfiles.admin.createtimestamp = 20030414042707Z
logfiles.admin.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
logfiles.admin.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
logfiles.admin.modifytimestamp = 20030414042707Z
logfiles.admin.objectclass = nsmsgCfgAlias ,top
logfiles.createtimestamp = 20030414042707Z
logfiles.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
logfiles.default.alias = |logfile|default
logfiles.default.createtimestamp = 20030414042708Z
logfiles.default.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
logfiles.default.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
```

logfiles.default.modifytimestamp = 20030414042708Z
logfiles.default.objectclass = nsmsgCfgAlias ,top
logfiles.http.alias = |logfile|http
logfiles.http.createtimestamp = 20030414042710Z
logfiles.http.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
logfiles.http.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
logfiles.http.modifytimestamp = 20030414042710Z
logfiles.http.objectclass = nsmsgCfgAlias ,top
logfiles.imap.alias = |logfile|imap
logfiles.imap.createtimestamp = 20030414042710Z
logfiles.imap.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
logfiles.imap.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
logfiles.imap.modifytimestamp = 20030414042710Z
logfiles.imap.objectclass = nsmsgCfgAlias ,top
logfiles.imta.alias = |logfile|imta
logfiles.imta.createtimestamp = 20030414042710Z
logfiles.imta.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
alarm.createtimestamp = 20030414042706Z
logfiles.imta.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
logfiles.imta.modifytimestamp = 20030414042710Z
logfiles.imta.objectclass = nsmsgCfgAlias ,top
logfiles.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
logfiles.modifytimestamp = 20030414042707Z
logfiles.objectclass = nsmsgCfgContainer ,top
logfiles.pop.alias = |logfile|pop
logfiles.pop.createtimestamp = 20030414042710Z
logfiles.pop.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
logfiles.pop.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
logfiles.pop.modifytimestamp = 20030414042710Z
logfiles.pop.objectclass = nsmsgCfgAlias ,top
modifiersname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
modifytimestamp = 20030414042706Z
nsclassname = "com.netscape.management.msgserv.MsgServer@msgadmin52
.jar@cn=admin-serv-sparc5-1, cn=Netscape Administration Server, cn=Server Group (2),
cn=sparc5-1.central.sun.com, ou=sparc5-1.central.sun.com, o=NetscapeRoot"
objectclass = top
,nsConfig
,nsAdminObject
pipeprograms.createtimestamp = 20030414042708Z
pipeprograms.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
pipeprograms.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
pipeprograms.modifytimestamp = 20030414042708Z
pipeprograms.objectclass = nsmsgCfgContainer ,top
service.authcachesize = 10000
service.authcachettl = 900
service.createtimestamp = 20030414042708Z
service.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
service.dcroot = o=internet
service.defaultdomain = sparc5-1.central.sun.com
service.dnsresolveclient = no
service.http.allowadminproxy = no
service.http.allowanonymouslogin = no
service.http.createtimestamp = 20030414042710Z
service.http.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
service.http.enable = yes
service.http.enablesslport = yes
service.http.fullfromheader = no
service.http.idletimeout = 3
service.http.ipsecurity = yes
service.http.maxmessagesize = 5242880
service.http.maxpostsize = 5242880
service.http.maxsessions = 6000
service.http.maxthreads = 250
service.http.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"

service.http.modifytimestamp = 20030414042711Z
service.http.numprocesses = 1
service.http.objectclass = nsmsgCfgHttp ,top
service.http.plaintextmincipher = 0
service.http.port = 80
service.http.resourcetimeout = 900
service.http.sessiontimeout = 7200
service.http.smtpport = 25
service.http.spooldir = /A1000/demo6789/ims52/msg-sparc5-1/http
service.http.sslcachesize = 0
alarm.createtimestamp = 20030414042706Z
service.http.sslport = 443
service.http.sslusessl = yes
service.imap.allowanonymouslogin = no
service.imap.banner = "%h %p service (%P %V)"
service.imap.createtimestamp = 20030414042710Z
service.imap.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
service.imap.enable = yes
service.imap.enablesslport = yes
service.imap.idletimeout = 30
service.imap.maxsessions = 4000
service.imap.maxthreads = 250
service.imap.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
[redacted] ,o=netscaperoot"
service.imap.modifytimestamp = 20030414042711Z
service.imap.numprocesses = 1
service.imap.objectclass = nsmsgCfgImap
service.imap.plaintextmincipher = 0
service.imap.port = 143
service.imap.sslcachesize = 0
service.imap.sslport = 993
service.imap.sslusessl = yes
service.ldapmemcache = no
service.ldapmemcachesize = 131072
service.ldapmemcachettl = 30
service.listenaddr = INADDR_ANY
service.loginseparator = @
service.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
service.modifytimestamp = 20030414042708Z
service.objectclass = nsmsgCfgService ,top
service.plaintextloginpause = 0
service.pop.allowanonymouslogin = no
service.pop.banner = "%h %p service (%P %V)"
service.pop.createtimestamp = 20030414042710Z
service.pop.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
service.pop.enable = yes
service.pop.idletimeout = 10
service.pop.maxsessions = 600
service.pop.maxthreads = 250
service.pop.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
service.pop.modifytimestamp = 20030414042711Z
service.pop.numprocesses = 1
service.pop.objectclass = nsmsgCfgPop ,top
service.pop.plaintextmincipher = 0
service.pop.popminpoll = 0
service.pop.port = 110
service.pop.sslusessl = yes
service.readtimeout = 10
store.admins = admin
store.cleanupage = 1
store.createtimestamp = 20030414042710Z
store.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
store.dbcachesize = 16777216
store.defaultacl = "anyone lrs"
store.defaultmailboxquota = -1
store.defaultmessagequota = -1
store.defaultpartition = primary
store.diskflushinterval = 15
alarm.createtimestamp = 20030414042706Z
store.expirerule.createtimestamp = 20030414042710Z
store.expirerule.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement
[redacted] ,o=netscaperoot"
store.expirerule.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
[redacted] ,o=netscaperoot"

store.expirerule.modifytimestamp = 20030414042710Z
store.expirerule.objectclass = nsmsgCfgContainer ,top
store.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement,o=netscaperoot"
store.modifytimestamp = 20030414042710Z
store.objectclass = nsmsgCfgStore ,top
store.partition.createtimestamp = 20030414042710Z
store.partition.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
store.partition.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
store.partition.modifytimestamp = 20030414042710Z
store.partition.objectclass = nsmsgCfgContainer ,top
store.partition.primary.createtimestamp = 20030414042710Z
store.partition.primary.creatorsname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
store.partition.primary.modifiersname = "uid=admin,ou=administrators,ou=topologymanagement
,o=netscaperoot"
store.partition.primary.modifytimestamp = 20030414042710Z
store.partition.primary.objectclass = nsmsgCfgPartition ,top
store.partition.primary.path = /A1000/demo6789/ims52/msg-sparc5-1/store/partition/primary
store.quotaenforcement = on
store.quotaexceededmsginterval = 7
store.quotagraceperiod = 120
store.quotanotification = off
store.quotawarn = 90
store.serviceadmingroupdn = "cn=Service Administrators, ou=Groups, o=isp"
store.umask = 077

Provisioning

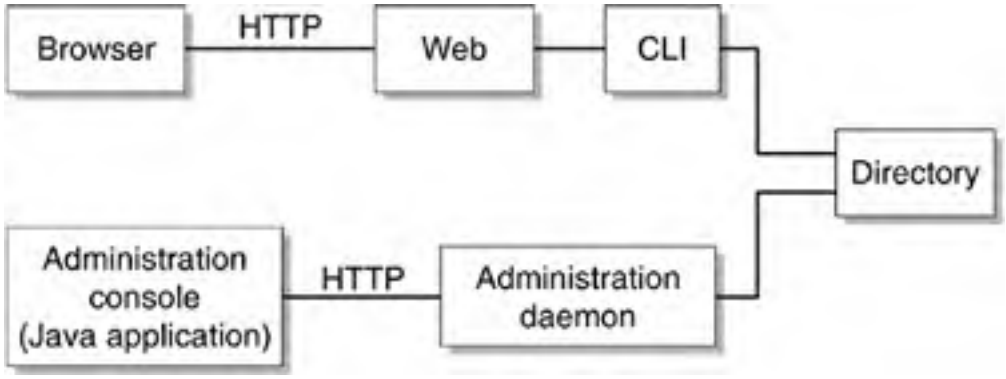
Once you know the system is installed and operational, the next major tasks is most likely how to best add and remove users, sometimes also called accounts or other terms. In the ISP world, the term used to describe this process is called *provisioning*. As with many things, there is often more than one way or method of provisioning accomplishing the task. Provisioning users or accounts is no different. Provisioning sometimes assumes starting a new user from scratch, but it may in fact be one of the steps in the migration process of an organization's older mail system. This chapter approaches it from the *new user* perspective, though the approach is not totally different when doing this as part of a migration. Some additional issues are discussed in Chapter 11, "Migration," on page 167."

Technically there may be many ways of provisioning, but there are really four main methods:

- Administration Console— Sun ONE Administration Console
- Web— delegated administration for messaging and collaboration or identity server
- CLI— command-line interface for the iPlanet Directory Administrator
- LDAP— direct interaction with the directory server via LDAP

Ultimately, these methods (FIGURE 5-2) all interact with the Messaging server in two places—the directory server and the mailstore. One thing to note is that creating user information in the directory does not create the physical mailbox in the mailstore nor does creating the mailbox (folder) in the mailstore create the user information in the directory. The tricky part is creating the user information in the directory with the appropriate attributes (fields) and having the users authenticate with the messaging system.

Figure 5-2. Administration Interfaces Architecture Overview



The following sections examine each of the four provisioning methods and outline some of the pros and cons of each method.

Administration Console

The Sun ONE Administration Console is a Java program that can be executed locally on the mail system or in a distributed (remote) fashion on any system supporting the Java™ Runtime Environment (JRE) 1.1.8. It connects to an administration process (daemon) through HTTP or HTTPS. The administration console provides a very low level access to most parts of the message system, including the Messaging Server, Directory Server, and Web Server. This access includes configuration data, user data, log data, and so forth. As such, it is primarily used for configuration and debugging purposes, not as a day-to-day way of administering the system.

Note

Occasionally problems exist when starting the Messaging Server Administration Console and remotely displaying the results. The splash logo can block the login entry box. To work around this issue you can start the Messaging Server Administration Console without the splash graphic by using the command startconsole -x nologo.

While it is possible to add users or change user information by using the administration console, it is typically reserved for configuration and diagnostic functions by the highest- level administrators.

An example of an appropriate use would be to configure the messaging server to accept IMAP over SSL connections. An example of poor usage for the administration console would be to add twenty user accounts or even one user account for that matter.

- Speed— Not zippy
- Ease of use— Good
- Access to functions— Very low level, most of the system
- Input checking— Little, if any
- Not customizable

You should equate the administration console to root or highest-level administration access, and as such it should be reserved for use by only those persons performing these duties on the messaging system.

Web

Another way to administer the Messaging System is via the Web. Actually, there are two ways to use the Web to administer the system—the Administration Web Interface and the Delegated Administrator for Messaging. The two methods are very different.

The Administration Web Interface provides basic administration function access such as starting, stopping, restarting, backing up, and restoration functions for some of the services of the Messaging Server. It is primarily designed for help desk personnel with minimum training and limited duties, as well as remote administration work when using the Java-based administration console is not possible.

The Delegated Administrator for Messaging (FIGURE 5-3)provides more user and domain management functionality for help desk and self-service end users. Functions for help desk personnel include adding, removing, and changing user information (if permitted). The functionality of the Delegated Administrator for Messaging can be restricted to provide specific help desk administrators for each domain within the messaging system as well as user administrators for each domain, which can ultimately reduce the burden of administration for the main administrators. For end users, this interface can be used to access functions such as mailing list management, vacation messages, mail filters, user information, and so forth, if permitted.

Figure 5-3. Delegated Administrator for Messaging

Overall these web interfaces are aimed at the help desk and end user population. They are not the fastest methods of administration, but they provide easy-to-use interfaces for occasional use. These interfaces can be customized because they are web pages (HTML, JavaScript™, and servlets) for your specific organization. So if you do not want people to be able to change their password or basic information via the Delegated Administrator interface, it can be modified to remove these options.

- Speed— Good
- Ease of use— Very good
- Access to functions— Help desk and basic administrator functions, end user self service
- Input checking— Some, but can be extended easily
- Customizable

Command-Line Interface

The functions to provision users and mail accounts in the Messaging Server are available from the command-line interface (CLI). This interface provides the ability to automate and program (script) your organization's business rules regarding user and mail account provisioning.

Note

The command line interface is really an interface into the Delegated Administrator, not directly into the Directory Server or the Messaging Server. You may see the terms CLI, Netscape delegated administrator (NDA) CLI, and IPlanet delegated administrator (IDA) CLI used interchangeably in this section.

This low-level interface is not designed for end users at all, and is typically reserved for the highest level of administrators or for help desk personnel via automation (scripts) only. It is very possible to add and delete thousands of users or accounts in minutes by using the CLI. It has the capabilities to quickly add and delete or otherwise mess up the Messaging Server, so treat it as you would root access on the server. This includes security issues such as ensuring to install the production server with appropriate users and groups as well as applying other security precautions.

Another use of the CLI is to perform debugging and diagnostic work. There are several commands as well as options that provide the ability to trace email paths and routing and look at the functioning of the various daemons. For more information, see Chapter 14 of the *iPlanet Messaging Server Administrators Guide* or go to:

http://docs.sun.com/source/816-6009-10/trblesho.htm#13833.

Code Example 5-3 Sample CLI Showing Creation of "testuser" Account

[View full width]

```
root@sparc5-1:/A1000/demo6789/ims52/ndacli/bin> ./imadmin user create -l testuser -W
    password -F Test -L User -D
serviceadmin@sparc5-1.central.sun.com -w bacon -n
sparc5-1.central.sun.com -H sparc5-1.central.sun.com
testuser@sparc5-1.central.sun.com: create user succeeded.
root@sparc5-1:/A1000/demo6789/ims52/ndacli/bin>
```

Overall, the command-line interface provides the one of the best ways to automate and perform bulk adds, deletes, and modifications quickly, plus implement an organization's policy regarding messaging.

- Speed— Good
- Ease of use— Good
- Access to functions— Very low level, scripts and senior administrators only
- Input checking— Some, but completely customizable
- Very customizable

Lightweight Directory Access Protocol

Ultimately, everything—the administration console, the web interfaces, and the CLI—interacts with the directory in some manner. So performing provisioning work by interacting with the directory by using the LDAP is not only possible, but very much like the NDA CLI in terms of capabilities.

Direct interaction using LDAP provides most of the benefits of the CLI without access to the very low level utilities and commands. It is also one of the best ways to automate and perform bulk adds, deletes, and modifications along with the CLI. And, just like the CLI, it can be used to enforce and implement an organization's policy regarding messaging. Direct interaction with the LDAP directory and its contents also means additional security precautions are necessary. Direct manipulation or access should be reserved for the highest- level administrators. Access for help desk personnel or lower-level administrators should be done only through scripts or other methods that can add additional checking and precautions. For more information see Chapter 10, "Security," on page 153."

- Speed— Excellent
- Ease of use— Good
- Access to functions— not as low level as the CLI, but still for scripts and senior administrators only
- Input checking— Some, but completely customizable
- Very customizable

[View full width]

```
User Commands                                   ldapmodify(1)

NAME
ldapmodify, ldapadd - ldap entry addition and modification tools

SYNOPSIS
ldapmodify [ -a ] [ -b ] [ -c ]  [ -r ] [ -n ] [ -v ]
[-F  ]   [  -d debuglevel  ]  [ -D binddn ] [ -w passwd ]
[-h ldaphost ] [ -M authentication ] [ -p ldapport ]
[-f file ] [ -l nb-ldap-connections ]

/opt/SUNWconn/ldap/bin/ldapadd [ -b ]  [ -c ]  [ -n ]
[ -v ] [ -F ]  [ -d debuglevel ] [ -D binddn ] [ -w passwd ]
[ -h ldaphost ] [ -p ldapport ] [ -f file  ]
[ -l nb-ldap-connections ]

DESCRIPTION
ldapmodify opens a connection to an LDAP server, binds, and modifies or adds entries. The
    entry information is read from standard input or from file, specified using the -f option.
    ldapadd is implemented as a hard  link to  the ldapmodify tool. When invoked as ldapadd
    the -a (add new entry) option is turned on automatically.
```

Code Example 5-4 Sample Template

```
dn: uid=<uid>, ou=people, o=<hostname_fqdn>, o=isp
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: inetUser
objectClass: ipUser
objectClass: nsManagedPerson
objectClass: userPresenceProfile
objectClass: inetMailUser
objectClass: inetLocalMailRecipient
mail: <uid>@<hostname_fqdn>
mailUserStatus: active
dataSource: NDA 4.5 Delegated Administrator
mailHost: <hostname_fqdn>
givenName: History
cn: <first_name> <last_name>
uid: <uid>
sn: <last_name>
mailDeliveryOption: mailbox
inetUserStatus: active
userPassword: <password>
creatorsName: uid=serviceadmin,ou=people,o=<hostname_fqdn>,o=isp
modifiersName: uid=msg-admin-<hostname_fqdn>-20020710153937,ou=people ,o=<hostname_fqdn>,
o=isp
createTimestamp: 20030414044513Z
modifyTimestamp: 20030414051012Z
nsUniqueId: d5cba701-1dd111b2-80cac302-81db34e7
nswmExtendedUserPrefs: meDraftFolder=Drafts
nswmExtendedUserPrefs: meSentFolder=Sent
nswmExtendedUserPrefs: meTrashFolder=Trash
nswmExtendedUserPrefs: meInitialized=true
pabURI: ldap://<hostname_fqdn>:389/ou=<uid>, ou=people, o=<hostname_fqdn>, o=isp,o=pab
```

Methods Analysis

After discussing the four methods of provisioning at a high level, the question "What is t*he one way* to do provisioning?" still remains. The answer is that it depends on the specific needs of your organization, including the end users, the administration staff, and the organization as a whole. Several factors, including which features such as vacation message or mailing list, may require the use of the Delegated Administrator for Messaging interface to some degree. Other factors include account turnover, skills of the administration staff, organizational policies, and so forth.

As a general rule, the best practices are:

Administrators console

- Top administrators only—for configuration
- Diagnostics and tuning only

Web

- Help desk and end-user self service
- Customize or write your own scripts. Use those provided as examples.
- Exceptions to bulk add and delete scripts. (There will always be exceptions.)
- Main provisioning interface for small organizations or others with minimal account additions and deletions—approximately one addition and deletion per day. (3,000 accounts with a one-percent change per year equals 300 accounts added or deleted per year.)

CLI/LDAP

- Automate as much as possible.

- Should handle 99+ percent of the work
- Exceptions handled by web or help desk personnel. (There will always be exceptions.)

Issues

There are many issues beyond which method is best or which method to use for provisioning. The remainder of this chapter uses the example of a university trying to automate their provisioning of the messaging system. Once automation has been decided upon, the most significant issues are:

- Authoritative Sources
- Data Feeds
- User ID

Each of these issues must be addressed and documented for the smooth provisioning of the messaging system. The following sections examine each of these issues in more detail and provide an example scripts to show how this all comes together.

Authoritative Sources

When provisioning users or accounts, some basic information is required—at a minimum, user ID, password, and email address to be used. The Messaging Server includes a directory server, so many organizations also use this to provide basic directory services to various LDAP-enabled applications such as Microsoft Outlook or Netscape Communicator. Realistically, you may want more information, such as full name and a phone number.

So, what is an authoritative source? An authoritative source is a source where the information or data is known to be accurate and up to date, and is most likely located where the information originates within the organization. In some cases it is not possible to integrate directly with the original source, but if something contains the same information and is as up to date, it can be said to be authoritative too. Simply put, information or data just does not suddenly appear from thin air; rather, it is the end result of a business process or part of a business process. A good example is the human resources (HR) system within an organization as an authoritative source on employees. Since you must go through the various HR processes and procedures to be an employee, it is highly likely that the HR system (database) is an authoritative source for employee information. Why? Legal and regulatory requirements as well as accounting (payroll). So it is pretty safe to say that a person who is not in the HR database is not an employee. Is this always true? No, but it does address at least 99 percent of the cases.

Regardless, think of this as a business workflow exercise, and ask the question "How does the organization get someone's identity, be they an employee, contractor, or whatever, before providing network or computing access?" Is it simply a matter of an email coming from a manager, or is it more formal, requiring that the person be in the payroll system too?

Typical authoritative sources might be:

- Human resource system (HRS)
- Student information system (SIS)
- Directory Server database
- Information systems database
- Contractor/vendor database
- Visiting guest database

In some organizations, there may actually be multiple sources from which information is available. Which source is valid depends upon a person's function or duties. Depending upon security requirements, you may also be required to check multiple sources to ensure that the person appears in all of them.

Data Feeds

Now that the authoritative data sources and their owners have been identified, the next issue faced when automating the provision process either via the CLI or direct LDAP integration is getting the data from the authoritative data sources. This requires input and agreement from the authoritative data source owners.

In a few situations, these authoritative systems or data sources can be directly integrated with the Messaging Server's directory via LDAP, to automatically provision accounts when new users are added to the payroll system or delete them when they are removed from the system. This requires some scripting or configuration on the authoritative source application. An example of this might be PeopleSoft 8's LDAP integration capabilities. However, in most organizations these data sources are not generally accessible, and so direct data extracts or integration is not possible.

Typically, though, there is a separation between the groups who control these systems and the group that administrates the messaging system. So many times tying the systems together directly is just not practical or possible. In these situations, the most practical integration is through a comma delimited file format sometimes referred to as a comma separated variable-length (CSV) file. Then, what information is needed and how this file is to be obtained and transferred and how often must be determined.

This file contains the information necessary to create the messaging account. It also typically contains a flag to indicate the action to be taken, whether this is a new account (add), or an existing account requiring deletion (delete) or updating (modify). There are also two other types of actions possible with the Messaging Server—activate and deactivate. One of the features of the Messaging Server is the ability to deactivate an account or entire domain while maintaining all of its associated information, including passwords, forwarding, address book entries, and so forth. This is a very useful feature in the University setting, where the policy might be to cut off services such as email if accounts are in arrears until such time as parking tickets or library fines are paid. Using this feature, an organization can simply deactivate an account and then easily reactivate the account without causing significant amounts of work such as adding the entire account back into the system.

So, the action flag or field in the CSV file might contain something to indicate the following actions:

- ADD
- DELETE
- MODIFY
- ACTIVATE
- DEACTIVATE

Our example uses the following designations to simplify this a bit, yet still represent all the actions:

- ADD
- DEL
- MOD
- ON
- OFF

Generally the file contains the basic information needed to create the record, as stated previously, plus some other information desired to make the directory useful. Often it is not necessary to be too literal, as some information or fields can be derived many times. An example might be full name where it is really a combination of first and last name. Every organization and authoritative data source is different, though. One thing to look for when using multiple authoritative data sources is fields that are common between them. Use the same fields consistently across all the data sources, if possible.

Since the purpose of this process is to automate the provisioning, the simplest method that does not require human intervention seems to work the best. In many organizations this could be a shared file system or using something such as File Transfer Protocol (FTP). This method is something that is worked out between the authoritative data source owner and the messaging administration group.

How frequently provisioning is done depends upon several factors. How quickly do you need messaging provisioned? Is there adequate CPU power to generate the CSV file as frequently as needed or desired? Some organizations do this nightly so that all new messaging accounts are created sometime between 10 p.m. and 4 a.m. for example. Other organizations desire more frequent updates, thus generating and processing the CVS hourly. Some modification of the scheduling can be done once the basics are working. Perhaps the biggest issue in this area is managing the end-user expectations. Were they informed that email accounts are created twice daily, or are they under the assumption that this is done in real time? Sometimes this issue is influenced by existing policy, while other times new policies must be set.

One final issue to consider regarding schedule or frequency is that not all functions have the same requirements. For example, if the policy is that new accounts are created once daily sometime between 10 p.m. and 4 a.m., that does not mean modifications or deletions must wait 24 hours. In some organizations, policy dictates that account deletions must happen within a short period of time for security reasons. So a policy such as this might drive the scheduling to something shorter than once daily.

User ID

Now that the issues of authoritative data sources and data feeds have been addressed, the next issue in provisioning is the user ID and everything that surrounds it. In some ways this is more difficult than the other issues to address because it is more of a policy issue. In some cases, it is necessary to maintain (grandfather) an existing standard for user ID creation for existing users while implementing a new policy for all new users.

Some points to consider when addressing the user ID issue are:

- User IDs must be unique.

  This can be within the entire messaging system or within each domain, that is, acme.edu. However, if IDs are unique only within each domain, users will be required to log in with their user ID plus domain, for example, jsmith@acme.edu, whereas if user IDs are unique across the entire messaging system users can log in using only their user ID, for example, jsmith. This can affect other issues, though.

  How do you ensure uniqueness?

  That can be difficult. Very few pieces of information about someone are completely unique. First name? No. Last name? No. First name plus last name? No. Social Security number? Pretty much, but there is a privacy issue.

  One possible answer is to make the user ID a derivative of the unique field from the authoritative source. This could be based on the social security number or employee ID number, but not actually use the number itself. Perhaps the person's first and last initials plus employee ID number could be used.

- Does not have to be tied to email address or name

  One common misconception is that a user's email address and user ID are the same thing. This is not necessarily true. While the two are linked together, they do not have to contain even the remotely the same information. For example, the user ID could be a12345 while the email address associated with that could be john.smith@acme.edu. The Messaging Server has the ability to assign multiple addresses to a single user ID. You must configure a primary email address at a minimum, but you can assign numerous alternate email addresses. So while user ID is a12345 and the primary email address is john.smith@acme.edu, it is quite possible to also assign jsmith@acme.edu, smitty@acme.edu, and johns@acme.edu. While there are other methods within the Messaging Server to do this, often a way to provide backward compatibility to older messaging environments is needed. An example might be that Acme University is consolidating two messaging servers into a single server. One was for faculty and staff (user@facstaff.acme.edu) and one was for students (user@student.acme.edu). Using the alternate address capabilities, it is possible assign user@acme.edu as the primary email address and user@student.acme.edu as the alternate address, so that email sent to the old address still arrives in the user's inbox.

- Organization's single sign on (SSO) strategy

  Since one part of the system is an LDAP-compliant directory server, it can be leveraged as the beginning of an organization-wide directory and authentication server. In some organizations, efforts to consolidate sign on and authentication sources are already underway. In still others, consolidation is only a notion, and in others, it is not even being considered. Rather than implement a messaging system with one set of user IDs today, only to have to rename the user IDs a short time later, it might be good to figure out if there is an SSO strategy or project underway.

- Goal and overall design of email system

  There is a significant difference between an email system designed for an ISP and one designed for an enterprise. One goal in some organizations might be email address for life. This means that the user ID and primary email address can never be reused. This very close to the next and final issue.

- Turnover (churn) in user population

  As with phone companies or cell phone companies, churn or turnover in user population is a key factor in operations in several ways. For a messaging system, it is not only an issue regarding provisioning method determination (that is, web interface versus CLI batch) but also reuse and provisioning of user ID and email addresses. Consider whether or not your organization has a 25 percent change in user population each year. That means in four years almost every user ID and email address in the messaging system will no longer be valid. If you have a jsmith user ID that is jsmith@acme.edu email address, how soon will it be before you reuse this address and user ID?

Sample Data File

For the samples, the user ID is derived from the person's first and last name—using a person's first initial plus their last name, so John B. Smith gets the user ID of jsmith. Should that user ID already exist in the system, you must add the person's middle initial. So, the user ID would be jbsmith if jsmith already exists. If this fails, the help desk can manually create the user ID in the system from the exception log. This example can easily be extended further with the first two initials of the first name, plus the middle initial, plus the last name, but we kept the example short to make it easier to understand.

So now you know that you need the action flags, the person's first name, the middle initial, and the last name. You can also add the person's phone number to the directory for the user to make the directory slightly more useful. As for the user's email address, use a simple view here and configure the email address with the user ID. So user jsmith's email address is jsmith@acme.edu. This could as easily have been set to john.smith@acme.edu.

```
ADD,Dave,,Pickens
ADD,Steve,D,Thomas
ADD,Steve,B,Thomas
ADD,Paul,B,Smith
OFF,pbunyan
ON,bblueox
DEL,ssimon
```

You must know the person's user ID when the person already exists in the messaging system, for operations such as DEL, ON or OFF. Assume that the user ID is available as part of the comma-delimited file in these cases.

Sample Provisioning Script

A provisioning script that actually automates everything based on the sample data file is located at:

http://ims.balius.com/.

Test User Generation Script

This script generates a sample user file that can be used as input in the previous provisioning script. It differs from the sample data file in that it will create any number of unique users from 1 to *n*. This script can be used to create hundreds or thousands of test accounts as needed.

Code Example 5-5 Test User Script Usage Example

```
# !/bin/csh
#
# This script adds demo accounts
# password set same as user_id
if ( $#argv != 1 ) then
    echo "Wrong number of arguments"
    echo "Usage: $0 {number}"
    echo ""
    echo "where {number} is the number of test accounts to add"
    exit 1
endif
set INSTALL_DIR=/A1000/demo6789/ims52
set mailhost=sparc5-1.central.sun.com
set passwd=bacon
set x=$1
cd $INSTALL_DIR/ndacli/bin
while ( $x > 0 )
    ./imadmin user create -l test${x} -W test${x} \
    -F Test${x} -L User -D serviceadmin@$mailhost \
    -w $passwd -n $mailhost -H $mailhost
    set x = 'expr $x - 1'
end
```

Edit the code for your specific installation. Replace the following variable:

- *INSTALL_DIR* is the install directory where you installed the Messaging Server into the mailhost. This is the fully qualified name of the Messaging Server.

Code Example 5-6 Add Test User Script Error Message

```
sparc5-1# root@sparc5-1:/> ./add_demo_users.csh
Wrong number of arguments
Usage: ./add_demo_users.csh {number}
```

Code Example 5-7 Add Test User Completion Message

```
root@sparc5-1:/> ./add_demo_users.csh 10
test10@sparc5-1.central.sun.com: create user succeeded.
test9@sparc5-1.central.sun.com: create user succeeded.
test8@sparc5-1.central.sun.com: create user succeeded.
test7@sparc5-1.central.sun.com: create user succeeded.
test6@sparc5-1.central.sun.com: create user succeeded.
test5@sparc5-1.central.sun.com: create user succeeded.
test4@sparc5-1.central.sun.com: create user succeeded.
test3@sparc5-1.central.sun.com: create user succeeded.
test2@sparc5-1.central.sun.com: create user succeeded.
test1@sparc5-1.central.sun.com: create user succeeded.
root@sparc5-1:/>
```

Chapter 6. Software Installation and Configuration

This chapter provides information and caveats that you may need during the installation phase of the overall messaging environment. It also discusses scalability issues. For additional details, refer to the *iPlanet Messaging Server Installation Guide for UNIX*. The chapter discusses the pros and cons of various answers to configuration questions and installation options so that you can avoid post-installation pitfalls, whether they are related to flexibility (that is, top domain name selection in directory), scalability, availability, performance, or ease of use. Thus, this chapter covers items not found in the current documentation and conveys information that can only be learned through experience.

Now that the system or systems have been prepared by following the instructions in Chapter 4, "Installation Preparation," on page 31," you can start the actual installation of the messaging server software. This process is relatively quick—more time is actually spent during configuration than installation.

Currently, the latest version of the Messaging Server software is 5.2, which will most likely change by the time you read this book. The Messaging Server software contains everything necessary to do a valid installation (Messaging Server, Directory Server, and Web Server software); however, it is advisable to install the latest version of the Directory Server software because the Messaging Server version 5.2 software contains an older version of the Directory Server. This adds a step or two to the install process, but it is not any more complicated than a normal installation.

This chapter covers the following topics:

- Simple Installation
- Automated Installation Script

Performing the procedures in this chapter installs the following software:

- Sun ONE Messaging Server 5.2 software
- Sun ONE Directory Server 5.1 Patch 1 software
- Sun ONE Web Server 6.0 software
- Sun ONE Messaging Server 5.2 Patch 1 software
- Sun ONE Calendar Server 5.1.1 software (optional)
- You can download this software from the Sun ONE web site at:

  http://wwws.sun.com.

  Follow the link for downloads.

Note

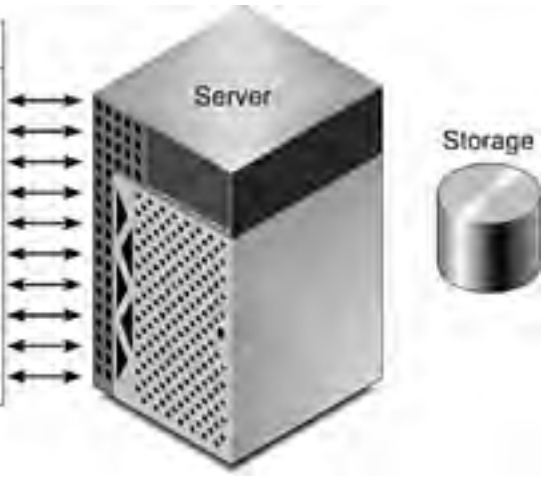
Read the release notes for any last-minute operating patches required for the software. Then download and apply these to the system.

The simplest configuration, as outlined in the Chapter 3, "Messaging Architectures," on page 15," is the "messaging in a box" or everything-all-on-one server architecture.

The Messaging Server software differs slightly from the preceding list due to the addition of several administration ports, as you can see from the diagram in FIGURE 6-1.

Figure 6-1. Simple Architecture With Administration Ports

Function	Port
WebMail	80
SMTP	25
POP	110
IMAP	143
LDAP	389
Directory administrator	55555
Messaging administrator	54321
Delegated administrator	88
Web server administrator	8888
Calendar server	81



One of the first steps in the installation process is to plan which ports will be used for the various connections. It is likely that you will elect to use the default ports for:

- SMTP— 25
- POP— 110
- IMAP— 143
- LDAP— 389

And it is likely that you will want to use:

- Webmail— 80

The default port of 80 may be in use on your system for a web server. Often, web servers such as Apache are installed and configured by default.

However, several administrator ports must be configured:

- Directory Administrator Console— Java-based GUI administrator for the messaging directory
- Messaging Administrator Console— Java-based GUI Administrator for messaging
- Delegated Administrator— web-based GUI for messaging
- Web Server Administrator Console— web-based GUI for the Web Server

You can use any port that is not currently in use and does not conflict with the others listed previously. For the procedure in this chapter use:

- Messaging Administrator Server— 55555
- Directory Administrator Server— 54321
- Delegated Administrator Server— 88
- Web Server Administrator Port— 8888

If you are unsure of whether a port is in use, on UNIX systems you can check using the netstat command:

```
# netstat -an
```

Now that you have selected the ports and determined that they are not in use, the actual installation process can begin.

Simple Installation

Simple installation installs a messaging server on a single system and makes it functional for other procedures in later chapters. The procedures are:

- Creating UNIX User and Group Accounts
- Disabling SendMail
- Installing a Master Directory Server
- Preparing the Master Directory Server for Messaging
- Installing the Messaging Server
- Installing the Delegated Administrator Server
- Setting Up Messaging Accounts and Testing the Server

TABLE 6-1 lists the values required for installation.

Table 6-1. Values Required for Installation

Domain Name	_____
Machine Name	_____
Machine IP address	_____
install-binaries	/temp/binaries—Create a subdirectory for each package.
/temp/binaries/iMS	Messaging Server 5.2
/temp/binaries/iDS	Directory Server 5.1p1
/temp/binaries/patch	Messaging Server 5.2 Patch 1
<server-root>	/opt/SunONE/ims52
<webserver-root>	/opt/SunONE/web4ida
<iDA-Root>	/opt/SunONE/ida4msg
Directory	Directory
Server User and Group	SunONE
Messaging Server User and Password	Mail
	SunONE
Web Server for Delegated Administrator GUI User	web4ida
Password	SunONE
Configuration Administrator ID	Administrator
Password	adminpass
Directory Manager DN	cn=Directory Manager
Password	adminpass
Messaging Server Service Administrator User ID	Service Administrator
Password	adminpass
Postmaster User Account	pma@*domainname*
Directory Server port	389
Messaging Server ports	

SMTP	25
HTTP	80
POP3	110
IMAP4	143
Administration Server port for Messaging Server	55555
Administration Server port for Directory Server	54321
Delegated Administrator running on Enterprise Server port	88
Administration Server port for Enterprise Server	8888

Creating UNIX User and Group Accounts

A best practice is to set up a UNIX user account and group for all Sun ONE servers, and then set the permissions appropriately for the directories and files owned by that user.

1. **Log in as root.**
2. **Issue the following command to create the Sun ONE Server group for the Solaris OE:**

   # **groupadd SunONE**

3. **Issue the following commands to create the messaging server user (for Solaris):**

   # **useradd mail**
   # **usermod -g SunONE mail**
   # **passwd mail**
   New password: **SunONE**
   Re-enter new password: **SunONE**

4. **Issue the following commands to create the Directory Server user for the Solaris OE:**

   The following examples assume csh:

   # **useradd directory**
   # **usermod -g SunONE directory**
   # **passwd directory**
   New password: **SunONE**
   Re-enter new password: **SunONE**

5. **Issue the following commands to create the web server user for the Solaris OE:**

   # **useradd web**
   # **usermod -g SunONE web4ida**
   # **passwd web4ida**
   New password: **SunONE**
   Re-enter new password: **SunONE**

   If your system is experiencing difficulty using these new accounts, you may have to create and specify home directories for them.

Disabling SendMail

A best practice is to stop and disable any programs running on needed ports before beginning a server installation. On most UNIX Solaris OE systems, the messaging program SendMail is running by default, which will interfere with the messaging server installation because both products want to use port 25 for SMTP. The Sun ONE Messaging Server installation program may or may not be able to disable SendMail for you. So you must manually stop SendMail and disable it from starting up on reboot.

1. **Log in as root.**
2. **Type the following:**

```
# /etc/init.d/sendmail stop
# ps -ef | grep sendmail
```

3. **Determine if the** sendmail **daemon is running by typing:**

```
# cat /etc/mail/sendmail.pid
```

This command sequence returns a process ID followed by the file path.

Example:

xxx /usr/lib/sendmail -bd -q15m

4. **Kill the** sendmail daemon **process by typing the command:**

```
# kill -9 xxx
```

Where *xxx* is the PID returned in Step 3.

5. **Issue the following command (Solaris OE) to move the SendMail configuration file to a safe place and prevent it from starting on the next system boot:**

```
# mv /etc/rc2.d/S88sendmail /etc/rc2.d/disabled.S88sendmail
```

Installing a Master Directory Server

In this section, you install a master directory server that your Messaging Server software will use to store the configuration and user account information.

The products you will install are the Sun ONE Directory Server 5.1 software and the Sun ONE Server Core Components software. Messaging Server 5.2 ships with Directory Server 4.16 Patch 1, which is an older version. It is recommended that the latest version of Directory Server 5.x be used.

You will install the new Directory Server, which is separate from the one that is packaged with the Messaging Server software. That Directory Server software has reached end of life (EOL) as discussed previously.

This procedure assumes you have downloaded, uncompressed, and unpacked the installation download files in the *install-binaries* directory.

In the instructions that follow, you must enter the values that appear in boldface. For all other values, just accept the defaults by pressing Enter.

1. **Change directories to the location of the directory server software:**

```
# cd install-binaries/iDS
```

Example:

```
# cd /temp/binaries/iDS
```

2. **Run the installer executable from the command line:**

```
# ./setup
```

3. **Install the directory server for messaging by answering the prompts as follows:**

```
Would you like to continue with installation? [Yes]:
Select the component you want to install [1]:
Choose an installation type [2]: 3
Install location [/usr/iplanet/servers]: /opt/SunONE/ldap
Specify the components you wish to install [All]:
Specify the components you wish to install [1, 2, 3]:
Specify the components you wish to install [1, 2]:
Specify the components you wish to install [1, 2]:
Computer name [hostname.yourdomain.com]:
System User [nobody]: directory
System Group [nobody]: SunONE
```

```
Do you want to register this software with an existing
iPlanet configuration directory server? [No]:
Do you want to use another directory to store your data? [No]:
Directory server network port [389]:
Directory server identifier [hostname]: hostname
administrator ID admin:
Password: adminpass
Password (again): adminpass
Suffix [dc=foo, dc=com]: o=isp
Directory Manager DN [cn=Directory Manager]:
Password: adminpass
Password (again): adminpass
Do you want to install the sample entries? [No]:
Type the full path and filename, the word suggest, or the word none [suggest]:
Do you want to disable schema checking? [No]:
Administration port [33530]: 54321
IP address [ ]: your_ip_address
Run Administration Server as [root]:
```

4. **You should see the following output:**

```
Extracting Sun One core components...
[......]
```

5. **Go to the** /opt/SunONE/ldap **directory and type** startconsole **to begin managing your servers.**

Preparing the Master Directory Server for Messaging

In this section, you will run the ims_dssetup.pl utility to prepare the master directory server for messaging. This perl script adds the additional directory objects necessary for the messaging server to store user preferences and so forth in the directory. Without this, the messaging server cannot operate correctly.

To prepare the Master Directory for messaging:

1. **Change directories to the location of the messaging server software:**

   # **cd** *install-binaries***/iMS**

   Example:

   ```
   cd /temp/binaries/iMS
   ```

2. **Run the** ims_dssetup **utility from the command line:**

   ```
   # ./ims_dssetup
   ```

3. **Prepare the directory server for messaging by answering the prompts as follows:**

```
Do you want to continue [y]:y
Directory server root [/usr/netscape/server4] : /opt/SunONE/ldap
Please select a directory server instance from the following list:
Which instance do you want [1]: 1
Will this directory server be used for users/groups for iMS [Yes]:Yes
Please enter the DC Tree base suffix [o=internet]:
Please enter the Users/Groups base suffix [o=your.domain.com] : o=isp
Do you want to update the schema files [yes]: yes
Do you want to configure new indexes [yes]: yes
Please enter the schema directory [msg/config]:
Please enter the directory manager DN [cn=Directory Manager]:
Password: adminpass
Do you want to continue [y]:y
```

You should see the following output:

[View full width]

Welcome to the iMS Directory Server preparation tool.

This tool prepares your directory server for iPlanet Messaging Server install.
Here is a summary of the settings that you chose:

```
  Server Root               : /sunone/demo/ids51
  Server Instance           : slapd-sparc5-3
  Users/Groups Directory      : yes
  Update Schema               : yes
  DC Root                   : o=internet
  User/Group Root             : o=isp
  Add New Indexes             : yes
  Schema Directory            : ./config
  Directory Manager DN          : cn=Directory Manager
```

Stopping Directory Server

Updating Schema files...

Starting Directory Server

Adding Suffixes... and turning off uid uniqueness plugins
Adding naming context o=internet
adding new entry cn="o=internet",cn=mapping tree,cn=config

adding new entry cn=internetdb,cn=ldbm database,cn=plugins,cn=config

Adding naming context o=pab
adding new entry cn="o=pab",cn=mapping tree,cn=config

adding new entry cn=pabdb,cn=ldbm database,cn=plugins,cn=config

Adding Indexes...
adding new entry cn=inetUserStatus,cn=index,cn=userRoot,cn=ldbm database,cn=plugins,cn=config

Welcome to the iMS Directory Server preparation tool.

adding new entry cn=db2index_2003_4_15_16_6_4, cn=index, cn=tasks, cn=config

modifying entry cn=mail,cn=index,cn=userRoot,cn=ldbm database,cn=plugins,cn=config

adding new entry cn=db2index_2003_4_15_16_6_7, cn=index, cn=tasks, cn=config

modifying entry cn=mailHost,cn=index,cn=userRoot,cn=ldbm database,cn=plugins,cn=config

adding new entry cn=db2index_2003_4_15_16_6_10, cn=index, cn=tasks, cn=config

adding new entry cn=inetMailGroupStatus,cn=index,cn=userRoot,cn=ldbm database,cn=plugins
,cn=config

adding new entry cn=db2index_2003_4_15_16_6_13, cn=index, cn=tasks, cn=config

adding new entry cn=modifytimestamp,cn=index,cn=userRoot,cn=ldbm database,cn=plugins,cn=config

adding new entry cn=db2index_2003_4_15_16_6_17, cn=index, cn=tasks, cn=config

adding new entry cn=mailUserStatus,cn=index,cn=userRoot,cn=ldbm database,cn=plugins,cn=config

adding new entry cn=db2index_2003_4_15_16_6_20, cn=index, cn=tasks, cn=config

adding new entry cn=createtimestamp,cn=index,cn=userRoot,cn=ldbm database,cn=plugins,cn=config

adding new entry cn=db2index_2003_4_15_16_6_23, cn=index, cn=tasks, cn=config

adding new entry cn=ou,cn=index,cn=userRoot,cn=ldbm database,cn=plugins,cn=config

adding new entry cn=db2index_2003_4_15_16_6_26, cn=index, cn=tasks, cn=config

adding new entry cn=cosspecifier,cn=index,cn=userRoot,cn=ldbm database,cn=plugins,cn=config

adding new entry cn=db2index_2003_4_15_16_6_29, cn=index, cn=tasks, cn=config

adding new entry cn=mailEquivalentAddress,cn=index,cn=userRoot,cn=ldbm database,cn=plugins
,cn=config

adding new entry cn=db2index_2003_4_15_16_6_32, cn=index, cn=tasks, cn=config

modifying entry cn=mailAlternateAddress,cn=index,cn=userRoot,cn=ldbm database,cn=plugins
,cn=config

adding new entry cn=db2index_2003_4_15_16_6_35, cn=index, cn=tasks, cn=config

adding new entry cn=dc,cn=index,cn=internetdb,cn=ldbm database,cn=plugins,cn=config

adding new entry cn=db2index_2003_4_15_16_6_38, cn=index, cn=tasks, cn=config

adding new entry cn=modifytimestamp,cn=index,cn=internetdb,cn=ldbm database,cn=plugins
,cn=config

adding new entry cn=db2index_2003_4_15_16_6_42, cn=index, cn=tasks, cn=config

adding new entry cn=createtimestamp,cn=index,cn=internetdb,cn=ldbm database,cn=plugins
,cn=config

adding new entry cn=db2index_2003_4_15_16_6_45, cn=index, cn=tasks, cn=config

adding new entry cn=inetDomainBaseDN,cn=index,cn=internetdb,cn=ldbm database,cn=plugins
,cn=config

Welcome to the iMS Directory Server preparation tool.
adding new entry cn=db2index_2003_4_15_16_6_48, cn=index, cn=tasks, cn=config

adding new entry cn=inetCanonicalDomainName,cn=index,cn=internetdb,cn=ldbm database
,cn=plugins,cn= config

adding new entry cn=db2index_2003_4_15_16_6_51, cn=index, cn=tasks, cn=config

adding new entry cn=mailDomainStatus,cn=index,cn=internetdb,cn=ldbm database,cn=plugins
,cn=config

adding new entry cn=db2index_2003_4_15_16_6_54, cn=index, cn=tasks, cn=config

adding new entry cn=mailRoutingHosts,cn=index,cn=internetdb,cn=ldbm database,cn=plugins
,cn=config

adding new entry cn=db2index_2003_4_15_16_6_58, cn=index, cn=tasks, cn=config

adding new entry cn=inetDomainStatus,cn=index,cn=internetdb,cn=ldbm database,cn=plugins
,cn=config

adding new entry cn=db2index_2003_4_15_16_7_1, cn=index, cn=tasks, cn=config

adding new entry cn=modifytimestamp,cn=index,cn=pabdb,cn=ldbm database,cn=plugins,cn=config

adding new entry cn=db2index_2003_4_15_16_7_4, cn=index, cn=tasks, cn=config

adding new entry cn=createtimestamp,cn=index,cn=pabdb,cn=ldbm database,cn=plugins,cn=config

adding new entry cn=db2index_2003_4_15_16_7_7, cn=index, cn=tasks, cn=config

adding new entry cn=memberOfPAB,cn=index,cn=pabdb,cn=ldbm database,cn=plugins,cn=config

adding new entry cn=db2index_2003_4_15_16_7_10, cn=index, cn=tasks, cn=config

adding new entry cn=memberOfManagedGroup,cn=index,cn=pabdb,cn=ldbm database,cn=plugins
,cn=config

adding new entry cn=db2index_2003_4_15_16_7_13, cn=index, cn=tasks, cn=config

adding new entry cn=memberOfPABGroup,cn=index,cn=pabdb,cn=ldbm database,cn=plugins,cn=config

adding new entry cn=db2index_2003_4_15_16_7_17, cn=index, cn=tasks, cn=config

adding new entry cn=un,cn=index,cn=pabdb,cn=ldbm database,cn=plugins,cn=config

adding new entry cn=db2index_2003_4_15_16_7_20, cn=index, cn=tasks, cn=config

Adding PAB and DC root...
adding new entry o=pab

adding new entry o=internet

root@sparc5-3:/stuff/test/messaging/solaris/iMS/msg #

Installing the Messaging Server

To install the Messaging Server:

1. **Change directories to the location of the Messaging Server software:**

   # **cd** *install-binaries***/iMS**

   Example:

   # cd /temp/binaries/iMS

2. **Run the installer executable from the command line:**

   # **./setup**

3. **Install the Messaging Server by answering the prompts as follows:**

   Would you like to continue with setup? [**Yes**]:
   Do you agree to the license terms? [No]: **yes**
   Please select the component you want to install [**1**]:
   Choose your installation type [**2**]:
   Server root [/usr/iplanet/server5]: **/opt/SunONE/ims52**
   Specify the components you wish to install [All]: **1,3,4**
   Specify the components you wish to install [**1, 2, 3**]:
   Specify the components you wish to install [**1, 2**]:
   Specify the components you wish to install [**1, 2**]:
   Specify the components you wish to install [**1, 2, 5**]:
   Computer name [<hostname>.<netscape.com>]: **hostname.<groupdomain>**
   System User [nobody]: **mail**
   System Group [nobody]: **SunONE**
   Do you want to register this software with an existing Netscape configuration directory server? [**No**]:
   Password (again): **admin**
   Suffix [o=<domainname>]: **o=isp**
   Directory Manager DN [**cn=Directory Manager**]:
   Password: **adminpass**
   Password (again): **adminpass**
   Administration Domain [**<domainname>**]:
   Administration port [25640]: **55555**
   Run Administration Server as [**root**]:
   User Name [SunONE]: **mail**
   Default Domain [<domainname>]: **<groupdomain>**
   Default Organization DN [o=<domainname>, o=isp]: **o=***groupdomain*, **o=isp**
   Host Name [*hostname.domainname*]:
   Port [80]:**80**
   Will the Messaging Server use a Smart Host [**2**]:
   Would you like to continue with setup? [**Yes**]:
   User ID [**ServiceAdmin**]:
   User Password: **adminpass**
   Confirm Password: **adminpass**
   Email Address: **pma@***groupdomain*

   The following messages are displayed:

   Extracting Netscape core components...
   Extracting Netscape Server Family Core components...

   [.....]

   Press Return to continue...

4. **Go to** /opt/SunONE/ims52 **and type** startconsole **to begin managing your Messaging Server.**

Installing the Delegated Administrator Server

To install the Delegated Administrator Server, perform the following procedures:

- Installing the Enterprise Web Server
- Installing the Delegated Administrator

Installing the Enterprise Web Server

You will install Sun ONE Enterprise Web Server 6.0 software. This software is required to run the Delegated Administrator.

Make sure the *webserver-root* value you use in the following procedure is different from the *server-root* you used previously for the messaging and directory servers.

1. **Change directories to the location of the Sun ONE Enterprise Web Server 6.0 software installation binaries:**

   # **cd** *install-binaries***/ES**

   Example:

   # cd /temp/binaries/iMS/solaris/ES

2. **Run the setup program:**

   # **./setup**

3. **Install the Web Server by answering the prompts as follows:**

   Would you like to continue with installation? [**Yes**]:
   Do you agree to the license terms? [No]: **yes**
   Choose an installation type [**2**]:
   Install location [/usr/netscape/server4]: **/opt/SunONE/web4ida**
   Specify the components you wish to install [**All**]:
   Specify the components you wish to install [**1, 2, 3, 4, 5, 6, 8**]:
   Computer name [**<hostname>.<domain>**]:
   System User [nobody]: **web4ida**
   System Group [nobody]: **SunONE**
   Run iWS Administration Server as [**root**]:
   iWS Admin Server User Name [**admin**]:
   iWS Admin Server Password: **adminpass**
   iWS Admin Server Password (again): **adminpass**
   iWS Admin Server Port [8888]: **8888**
   Web Server Port [80]: **88**
   Do you want to register this with an existing Directory Server [**No**]:
   Web Server Content Root [**/opt/SunONE/web4ida/docs**]:
   Do you want to use your own JDK [**No**]:
   Extracting Server Core...
   [.....]
   Press Return to continue...

4. **Start the Administration Server by typing:**

   # *webserver-root*/**https-admserv/start**

   Example:

   # /opt/SunONE/web4ida/https-admserv/start

5. **Start the Web Server by typing:**

   # **webserver-root/https-***hostname.domain***/start**

   Example:

```
# /opt/SunONE/web4dia/https-acme.edu/start
```

Installing the Delegated Administrator

You can now install the Delegated Administrator.

1. **Change the directory to the location of the Delegated Administrator installation binaries:**

   # **cd** *install-binaries*/**iDA**

   Example:

   ```
   # cd /temp/binaries/iMS/solaris/iDA
   ```

2. **Run the setup program:**

   ```
   # ./setup
   ```

3. **Install the Delegated Administrator graphical user interface (GUI) by answering the prompts as follow:.**

   Would you like to continue with installation? [**Yes**]:
   Do you agree to the license terms? [No]: **yes**
   Install location [/usr/netscape/ida10]: **/opt/SunONE/ida4msg**
   Manage Messaging Server [No]: **yes**
   Specify Host Name [*hostname.domainname*]:
   Specify Admin URL: **http://***hostname.domain***:88/**
   Specify CGI Path [**msg-<hostname>/Tasks/operation**]:
   Manage Calendar Server [**No**]:
   Specify Enterprise server config directory:
   <webserver-root>/https-*hostname.domain*/config
   Specify LDAP URL: **ldap://***hostname.domain***:389**
   Specify Directory Manager [**cn=Directory Manager**]:
   Password: **adminpass**
   Specify Suffix: **o=isp**
   This suffix is already present in the directory.
   Continue without installing iDA information in the directory? [No]: **yes**
   Specify DC Suffix [**o=internet**]:
   Specify Suffix [**o=isp**]:

The following messages will be displayed:

```
Extracting Netscape core components...
Extracting iPlanet Delegated Administrator for Messaging...
Restarting Enterprise Server
Connecting netscape browser to
        http://<hostname>.<domainname>:88/nda/start.htm
Press Return to continue...<Return>
```

Your Netscape browser may or may not actually start depending upon your specific installation. If it does not start, open your browser and manually enter the URL listed in the output.

Setting Up Messaging Accounts and Testing the Server

The purpose of this section is to test your Messaging Server. To do this, perform the following procedures:

- Creating a Postmaster User Account
- Creating Test Accounts
- Verifying Your Messaging Server Works Using WebMail

You must add and manage users through the Delegated Administrator, which you should now be running on port 88. You can either user the web interface, or the command-line utilities that ship with the messaging product.

Here, you will use the command-line utility imadmin. The minimum format for adding messaging users to specific messaging hosts is:

```
# imadmin user create -D admin_id -w admin_password -l users_uid -n users_domain -W users_password -
F users_firstname -L users_lastname -H users_messaging_server
```

Creating a Postmaster User Account

When you installed the Messaging Server, a postmaster group was automatically created in the directory for you. During installation, you specified a unique member of the group (pma@*domainname*) that will receive errors and other notices from the Messaging Server. Now you must actually create this user so these notices can be delivered and read. To set up this user account:

1. **From a shell window of any of your messaging machines, change to the Delegated Administrator command-line utilities directory:**

   # **cd** *server-root***/ndacli/bin**

   Example:

   ```
   # cd /opt/SunONE/web4ida/ndacli/bin
   ```

2. **Create the postmaster user account:**

   # **./imadmin user create -D serviceadmin@***domainname* **-w adminpass -l pma -n** *domainname* **-W adminpass -F Postal -L Worker -H** *hostname.domainname*

   Example:

   [View full width]

   ```
   # ./imadmin user create -D serviceadmin@mail.acme.edu -w adminpass -l pma -n mail.acme.edu
   -W adminpass -F Postal -L Worker -H acme.edu
   ```

You should see a message like the following:

```
pma@acme.edu: create user succeeded.
```

Creating Test Accounts

Using the command-line utility as in the previous section, create some test accounts that you can use to test your messaging system.

1. **From a shell, change directories to the Delegated Administrator command-line utilities:**

   # **cd** *server-root***/ndacli/bin**

   Example:

   ```
   # cd /opt/SunONE/web4ida/ndacli/bin
   ```

2. **Create a user account (test1):**

   # **./imadmin user create -D serviceadmin@***domainname* **-w adminpass -l test1 -n** *<groupdomain>* **-W testpass -F Test -L Account1 -H** *hostname.domainname*

   Example:

[View full width]

```
# ./imadmin user create -D serviceadmin@acme.edu -w AdminPass -l test1-n acme.edu -W
  userpasswd -F Test -L Account1 -H mail.acme.edu
```

You should see a confirmation message like the following:

```
test1@acme.edu: create user succeeded.
```

3. **Repeat the preceding process for test2 through test 5.**
4. **Create a user account (calmaster):**

```
# ./imadmin user create -D serviceadmin@domainname -w adminpass -l calmaster -n domainname -W
adminpass -F Calendar -L Account -H hostname.domainname
```

You have created this test user account for a supplemental lesson on installing a Sun ONE Calendar Server. The calmaster account is required for Calendar Server installation at a later time.

Verifying Your Messaging Server Works Using WebMail

You should now be able to log into the Messaging Server using the test accounts and send messages.

1. **Launch your web browser or bring up a new browser window.**
2. **Go to your server's web mail location:**

   http://*hostname.domainname*
3. **Enter the Username (test1, test2, test3, test4, or test5) and Password (testpass) for each server's test account and press return or click Login.**
4. **Click Compose and compose a message to test1, test2, test3, test4, and test5. Click Send when you are done.**
5. **Read the messages by clicking Get Mail.**

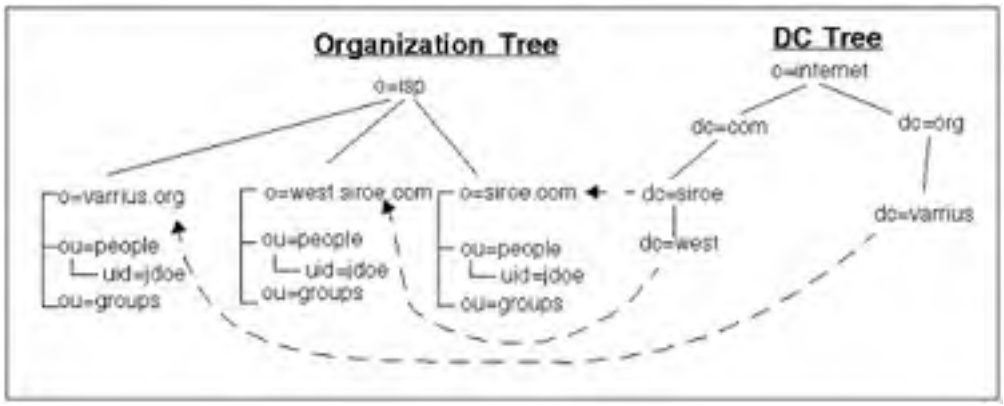
When you have successfully sent and retrieved messages from each messaging account on each server, you are done.

Congratulations, your Messaging Server works. For information regarding configuring your new Messaging Server, see Chapter 8, "Advanced Messaging Client Configuration," on page 103,".

Note

Referring to FIGURE 6-2, why were the organizations o=Internet for the DC tree and o=isp used as part of the User/Group tree? Using o=Internet at the top level allows you to host unrelated domains such as both acme.edu and baker.com. Including o=isp as part of the User/Group tree allows a flat name space (if desired) so Joe Smith's user ID of jsmith is used only once across all domains.

Figure 6-2. DC Tree and UG Organization Tree



Automated Installation Script

To short cut the preceding process and make things consistent, an installation script that automates the install process is available from the Sun. You can obtain this script and instructions from:

http://ims.balius.com/.

Note

No warranty is given; by downloading you accept this script as is.

You still must download the directory and messaging server binaries separately, and uncompress and unpack them.

Chapter 7. Message Transfer Agent Configuration

This chapter provides best practices and techniques regarding the setup and configuration of the Message Transfer Agent (MTA) component within the Messaging Server. Due to its complexity, this is an area that can cause significant issues related to security as well as basic functionality. This section dissects the default "out-of-the-box" MTA configuration file to provide a starting point for the reader. Many users of the previous versions of Sun Internet Mail Server (SIMS) or Netscape Messaging Server (NMS) had never seen an Innosoft PMDF product MTA configuration file. Therefore, this area is very intimidating and confusing. This chapter addresses some typical changes in plain English. For a more detailed discussion of issues such as antivirus checking and antispam processing, refer to and "Virus Scanning" on page 198 and "Antispam" on page 199.

This chapter contains a brief overview of the MTA and covers the following topics:

- Changing the Mappings
- Direct LDAP Lookup
- Adding New Domains to the MTA
- SMTP Authentication

First, a little history of the MTA that is within the Messaging Server. In March 2000, Sun Microsystems purchased a software company called Innosoft International. Innosoft International was the vendor of a mail product called PMDF. PMDF ran on a variety of platforms including the Solaris OE and VMS and was well respected with regard to performance, stability, scalability, and security.

During the course of the next two years, Sun integrated PMDF into the current version of the messaging product, starting with Messaging Server v5.0. But even before that Sun had OEMed the MTA portion of the PMDF product and it is used in SIMS version 3.5 and 4.0. So people have seen some of the PMDF configuration files in disguise. Administrators who are familiar with PMDF will feel right at home. Those who are not familiar with PMDF will have a little bit of a learning curve to climb.

PMDF was *more* than just the MTA, it had a message store (it was actually two message stores on VMS, and two on UNIX, one native, and one also based on the Carnegie Mellon University Cyrus mail program). It is a mail interconnect that talks many protocols, such as X.400, and talks to many PC mail systems. The MTA is where 50 percent of the configuration and options are within the mail system—it touches every single message that comes into or goes out of the messaging system.

Now for some basics……

Some people may be familiar with the term MTA. In reality, this is fancy terminology for message router.

According to the Telecom Glossary 2000:

> message transfer agent (MTA): An OSI application process used to store and forward messages as described in the X.400 message handling system synonym Internet, also known as a mail agent.

Just as an Ethernet router makes sure packets go where they are supposed to go and keeps them from going where they are not supposed to go, the MTA performs this function for messaging systems. One of the key points to note in the definition is "store and forward." The MTA does not simply forward or route, but stores a copy locally until it is sure that it has passed the message along or rejected it.

The basic MTA function of receiving and forwarding messages is performed in conjunction with information found in the directory. The MTA is a standalone daemon, and while required on the mail store, it can actually run by itself on a separate server. See Chapter 3, "Messaging Architectures," on page 15.

Out of the box the MTA is pretty plain, yet secure, in its configuration. However, there are several changes that organizations frequently make.

Typical changes include:

- Changing the definition (mapping) of what is local and what is not local
- Enabling direct LDAP lookup
- Accepting alternative domains
- Requiring SMTP authentication

- Rewriting domains

Changing the Mappings

This change opens up what is considered local and what is not local:

/*msgHome*/msg-*Instance*/imta/config/mappings

where *msgHome* is the directory where the messaging software was installed, and *Instance* is the name of the messaging instance (install), often the hostname (short host name).

If you look at the file, one section determines which IP addresses are to be considered internal:

```
INTERNAL_IP

 $(129.152.159.131/32)  $Y
 127.0.0.1  $Y
 *  $N
```

This file prevents people from using this server to relay messages via SMTP without authenticating as valid users.

> ## Note
>
> Mapping and other MTA configuration files are very picky regarding formatting, including line spacing and indentation. Consult the documentation for details.

The three lines indicate that:

- The subnet 129.152.159.131 with a bitmask of 32 (255.255.255.255) is considered internal, so nothing is on that subnet.
- The IP address 127.0.0.1 is considered internal ($Y = YES)
- Anything else (* wildcard) is *not* internal ($N = NO)

By changing the line for the subnet, you can open the ability to relay through this server. This change is useful for small environments and demonstrations, but must be carefully examined in large environments.

```
$(129.152.159.131/24)  $Y
```

The MTA must be restarted to pick up or initialize this change:

```
# su root
# cd /<msg-Home>/msg-<Instance>
# ./imsimta cnbuild
# ./imsimta restart dispatcher
```

Now the entire subnet of 129.152.159.*xxx* can use this MTA for relaying messages.

> ## Note
>
> Alternatively, you could use imsimta refresh, which combines the imsimta cnbuild and imsimta restart commands into a single command.

Direct LDAP Lookup

Prior to version 5.1 of the messaging software, the MTA did not have the ability to directly look up information in the Directory Server via LDAP. Rather, the MTA had a small cache of information, such as user IDs and mail addresses, that was periodically synchronized against the Directory Server. This was originally done when Directory Server performance was not as good as it is today. Now that the Directory Server is able to keep up with the requests from the Message Server, the MTA's cache is redundant and becomes a bottleneck, as well as adding administrative overhead. You may also hear the term *dirsync* used to describe the process or the daemon that is run to synchronize the data between the Directory Server and MTA's cache. One reason the use of the MTA cache was abandoned is that in some situations the information in the cache would become stale, causing some interesting problems—for example, users added to the system would not appear in the MTA cache until the next dirsync was run. Password changes would not necessarily be immediately reflected either. By using direct LDAP access, these problems are avoided.

So, it is highly recommended that the MTA be configured to utilize direct LDAP lookups.

Why is it so desirable to move to direct LDAP lookups?

Dirsync was used in iPlanet Message Server and in SIMS before that for a number of reasons that were good at the time. Dirsync provides a decoupling of the messaging server from the directory infrastructure, which, in the days of SIMS 3.5, was immature (by which we mean slow and not entirely reliable). It also reflected the ancestry of the product, which had been entirely independent of LDAP.

Dirsync represented a technical compromise. Given that LDAP was slow and unreliable, the approach taken was to predigest the directory information into databases for use by the MTA. In theory, this should give better performance and independence from the directory. In practice, however, these databases have been the bane of our lives. Wherever you have persistent structured data, there is always the concern that it can become inconsistent. And when you have a long update process like dirsync, you have a very unpleasant window where a failure can lead very quickly into a situation where manual intervention is required for a restart.

Dirsync also imposes a very abnormal load on the directory. Both the incremental dirsync and full dirsync queries are difficult for the directory server, very unlike the sort of query for which the directory was designed, which is "here is a attribute/value pair, find me the matching entry."

Since the days of SIMS, the directory technology has improved significantly. Now, the directory is very robust and much faster. The behavior of the directory is much better than the behavior of the databases used by the MTA. So the balance of the compromise is now very different. The speed of looking up a user in the directory is still too slow for the MTA to use the directory in a simplistic way, but with the inclusion of a pre-process cache for reading the directory information, we have found that the throughput in general goes up. To be fair, we can construct loads where the throughput goes way up or way down, but with a realistic load there is a net gain in throughput.

But the real win is in robustness. By going to the direct LDAP mode, you eliminate a whole set of complicated persistent data structures, replacing them with transparent ephemeral data structures. This not only eliminates a set of failure modes, but (and this is probably more important) means that the probability of needing manual intervention after any sort of incident is significantly reduced.

When Sun first introduced the direct LDAP mode, they did so more tentatively than was wise. Initial thoughts were to err on the side of caution by making dirsync the default mode for version 5.2 of the Messaging Server. In retrospect that was an error. The direct LDAP mode, after Sun had cleaned up a couple of weird corner cases, has proven far more robust and easy to deploy than they had ever hoped.

In the next release, Sun intends to make direct LDAP mode the only mode of operation now that it is known to work well. We are that satisfied with its behavior. It makes the directory deployment easier and the MTA much more stable, and makes it much easier to recover from any sort of hardware or software failure.

Already there is functionality in the area of mailing groups that is only supported in direct LDAP mode. Given that dirsync is now code with a very limited life expectancy, you can expect the developers to concentrate their efforts in the direct LDAP mode. Thus, dirsync is now more or less in maintenance mode only.

Beginning with version 5.1 of the Messaging Server, the ability to directly look up information from the Directory Server is available, though it was not well documented until version 5.2. The default, however, is that the MTA still caches information unless explicitly configured to perform this *direct LDAP lookup*. In future versions of the Messaging Server, the default will be direct LDAP lookup.

Four MTA configuration files must be modified to enable direct LDAP lookup:

- mappings
- job_controller.cnf
- option.dat
- imta.cnf

All of these files are in the config directory for the MTA:

/*msgHome***/msg-***Instance***/imta/config/**

where *msgHome* is the directory where the Messaging Server software was installed, and *Instance* is the name of the messaging instance (install), often the hostname (short host name)

Testing LDAP Lookup

A simple experiment can be done to demonstrate the value and verify that direct LDAP lookup works:

1. **With the messaging system running, add a user via the** imadmin **command.**

   See Chapter 6, "Software Installation and Configuration," on page 69.

2. **Send a message to this user from another user's account.**

   You should get a "user not found" message, or something to that effect.

3. **Sync the MTA with the directory:**

   a. **To initialize the Messaging Server MTA's databases with information from the directory, issue the commands:**

   ```
   # su root
   # cd /msg-Home/msg-Instance
   # ./imsimta dirsync -F
   # ./imsimta restart dispatcher
   ```

   where *msg-Home* is the directory where the messaging software was installed, and *Instance* is the name of the messaging instance (install), often the hostname (short host name).

   b. **Try to send a message again.**

   This attempt should be successful.

   c. **Stop the messaging system.**

   d. **Edit the four MTA configuration files.**

   Before you edit the following embedded instructions, make backup copies.

   Example (options.dat):

   ```
   ! VERSION=1.0
   ! Modified by IMS administration server on: Tue Nov 12 15:08:15 EST 2002
   !
   ! Uncomment out the next 5 lines to enable Direct LDAP mode
   ! ALIAS_MAGIC=8764
   ! ALIAS_URL0=ldap:///$V?*?sub?$R
   ! USE_REVERSE_DATABASE=4
   ! REVERSE_URL=ldap:///$V?mail?sub?$Q
   ! USE_DOMAIN_DATABASE=0
     MISSING_RECIPIENT_POLICY=1
     ALIAS_DOMAINS=6
   ```

   e. **Restart the messaging system.**

   f. **Add a user using the** imadmin **command.**

   g. **Send a test message as above.**

   This should now work without requiring the dirsync command.

Adding New Domains to the MTA

There are several ways to add new domains beyond the initial domain configured during the installation process. Each of these methods offers its own advantage and disadvantages. The recommended method to manage additional domains is via the LDAP directory as per the documentation. This provides the advantage that all MTAs in your messaging environment get the same information with one update rather than having go to each and every MTA to manually edit the files. Additional benefits include reducing the risk of typos (for example, one of your four MTAs has a typo of edfg.com rather than defg.com) and having to restart the MTA to recognize the change. So it pays to learn to use LDAP to manage your domain names.

The following section provides some basic lessons regarding manually editing the MTA configuration files for new domains. One other change that may be necessary when configuring a standalone MTA is the ability to accept messages destined for multiple domains. By default, the MTA is configured to accept mail for the domain that was entered at the time of install. To get the MTA to accept mail destined for other domains, either internally or perhaps as a legacy compatibility issue, you must modify the imta.cnf file in the **/***msgHome***/msg-***Instance***/imta/config** directory.

Assuming the messaging system was originally installed for domain abcd.com, but you want it to also accept messages for efgh.com because that is the old name of the company, you can configure the MTA to recognize efgh.com as a domain name it *owns.* Otherwise, the MTA thinks that addresses in this domain are *remote* addresses and it just sends them back out to the Internet rather than looking them up in the LDAP directory. The real problem is that the addresses are not being recognized as *local*. To get the addresses recognized as local (and looked up in LDAP), they must match the local (l) channel.

There are several ways to get new domain names to be recognized including simply using the Delegated Administrator interface to add a domain into the system. We will look at another way to do this manually at the MTA configuration file level for those situations where you either do not want to add the domain by using the Delegated Administrator or you cannot use the Delegated Administrator for some reason.

A rewrite rule must be added to the imta.cnf file (towards the top, among the other rewrite rules), such as:

efgh.com $U%$D@*name-of-your-l-channel*

where n*ame-of-your-l-channel* is the official host name (also known as *channel tag*) on your local (l) channel (for example, mail.abcd.com).

Make sure to issue the commands imsimta, cnbuild, and imsimta restart dispatcher to make this change take effect.

One alternate option is to completely rewrite the addresses. The local parts of the addresses must be identical (for example, dpickens@abcd.com = dpickens@edgh.com). The upside to this option is that you do not have to have two addresses or use the alternate address field for Dave Pickens in LDAP, just the normal dpickens@abcd.com. The downside to this is that you lose the information (data) regarding what domain this email was originally sent to (for example, you do not know if it was sent to dpickens@abcd.com or dpickens@efgh.com).

The MTA can easily rewrite efgh.com to abcd.com. So, instead of the preceding rewrite rule it would look something like:

efgh.com $U%abcd.com

If you want to change the efgh.com addresses even in the headers, or if you want to leave efgh.com visible in headers, use:

efgh.com EF$U%abcd.com

Modifying the imta.cnf file

Take a look at this in action:

1. **Edit the** imta.cnf **file in the /***msg-Home***/msg-***Instance***/config directory.**

   Make a backup copy first.

2. **Rewrite the domain** abcd.com **to the default domain you have installed:**

   abcd.com $U%name-of-your-l-channel

3. **Restart the MTA**.

```
# su root
# cd /<msg-Home>/msg-<Instance>
# ./imsimta cnbuild
# ./imsimta restart dispatcher
```

Alternatively, you could use imsimta refresh, which combines the imsimta cnbuild and imsimta restart commands into a single command.

4. **Send a test message to an existing user, but use the** abcd.com **domain now, and examine the message in the user's mailbox**

SMTP Authentication

By default on the Messaging Server, users need not submit a password when they connect to the SMTP service of the Messaging Server to send a message. (We do not force SMTP AUTH.)

Authenticated SMTP is an extension to the SMTP protocol that allows clients to authenticate to the server. The authentication accompanies the message. The primary use of authenticated SMTP is to allow local users who are traveling (or using their home ISP) to submit mail (relay mail) without creating an open relay that others can abuse. The AUTH command is used by the client to authenticate to the server.

You can use authenticated SMTP with or without SSL encryption.

The maysaslserver, mustsaslserver, nosasl, nosaslserver, switchchannel, and saslswitchchannel channel keywords are used to configure Simple Authentication and Security Layer (SASL) SMTP AUTH during the SMTP protocol by SMTP channels such as Transmission Control Protocol/Internet Protocol (TCP/IP) channels. The nosasl keyword is the default and means that SASL authentication is not permitted or attempted. It subsumes nosaslserver, which means that SASL authentication is not permitted. Specifying maysaslserver causes the SMTP server to permit clients to attempt to use SASL authentication. Specifying mustsaslserver causes the SMTP server to *insist* that clients use SASL authentication; the SMTP server does not accept messages unless the remote client successfully authenticates.

Examining the imta.cnf File

Examine the imta.cnf file found in the **/***msgHome***/msg** *Instance*/imta/config/directory as follows:

1. **Locate the section titled "! part II : channel blocks."**
2. **Look for the"! tcp_local channel."**

You might think that mustsaslserver would be appropriate to lock down the messaging system and require SMTP AUTH. However, this is not quite the case. Let us examine this from the Internet side of things. Do other messaging systems sending email to you have logins and passwords? No. So mustsaslserver will require everyone using the MTA to authenticate.

So, why is the MTA configured with the maysaslserver, and would that not leave the MTA open for relaying?

The keyword maysaslserver allows for both unauthenticated and authenticated SMTP connections and traffic. The key here is what happens *after* someone successfully authenticates. Previously, we discussed the concept of what is considered internal and what is not when looking at the mappings file. By authenticating, the MTA now treats this connection as internal. Unauthenticated connections and traffic are considered external unless something in the mappings file indicates otherwise (for example, they are on a specific subnet or from a specific IP address).

Does this leave the MTA open for relaying? No, you must be submitting a message for a valid user on the system or you must have authenticated to relay to external mail systems.

Chapter 8. Advanced Messaging Client Configuration

One of the most overlooked features of IMAP-based messaging systems is the ability to share folders between users. This feature provides the solution to several issues faced in organizations:

- Administrator needs access to boss's mailbox while his boss is traveling.
- Email must be covered while someone is on vacation.
- Group needs to coordinate files and emails for a project.
- System-wide template folders and miscellaneous mailboxes must be accessible by everyone.

The Messaging Server provides the ability to share folders. This feature can be used by most IMAP clients such as Netscape Communicator or Outlook Express. The native web mail interface that is part of the Messaging Server also provides the ability to use and access shared folders.

One interesting point is that direct delivery to a shared folder or user folder is permitted under the mail standards. The format for this is:

user_email_address+folder_name@domain_name

Example:

steve.student+math101@acme.edu

This command delivers the email directly to Steve Student's folder called math101.

The shared folders feature is enabled by default within the Messaging Server. However, many users are not familiar enough with their client program to configure them appropriately.

This chapter provides the necessary steps and procedures for configuring shared folders for some of the more popular mail programs.

Note

Currently the Messaging Server only supports the ability to share folders within the same server and does not have the ability to share across multiple servers. Sharing across multiple servers is being considered for future releases.

This chapter covers the following key concepts and topics:

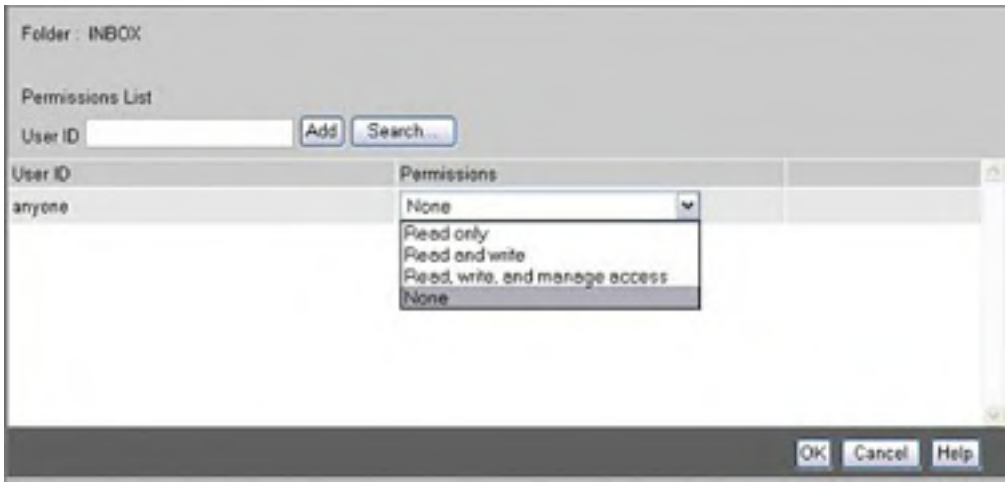
- What Is a Shared Folder?
- Supported Standards
- Limitations

What Is a Shared Folder?

A shared folder is one that you allow others to access. Several level of access control are available. For example, in web mail, you can allow others *Read only; Read and write; Read, write, and manage* access; or *None* to your folder. *None* is the default. FIGURE 8-1 shows the "Permissions" you can set in web mail.

- **Read only:** Allows users to only read the messages in the shared folder.
- **Read and write:** Allows users to read and set flags on messages in the shared folder. It also allows users to delete messages and subfolders.
- **Read, write, and manage:** Allows users to read messages, set flags (setg) on the messages in the shared folder, create subfolders under the shared folder, delete the subfolders, and share the folder with others.

Figure 8-1. Web Mail Shared Folder Permissions



Note that when a subfolder is created, it inherits the permissions of its parent folder. Once the subfolder is created, changing the permissions of its parent folder has no effect on the subfolder. FIGURE 8-2 shows that you can display the folder list by clicking Folders, then selecting a folder and clicking Share.

Figure 8-2. Getting to the Permissions Screen

Supported Standards

The Internet RFC 2086, *IMAP4 ACL Extension*,

http://www.ietf.org/rfc/rfc2086.txt?number=2086,

is the standard that defines the access control lists (ACLs) used in the IMAP4 protocol. The Message Server has supported RFC2086 since version 5.0.

RFC2086 describes the ACL as a set of identifier and rights pairs. For our purposes, the user ID for the IMAP user is the identifier.

The standard rights defined are:

- l - lookup (mailbox is visible to LIST and LSUB commands)
- r - read (SELECT the mailbox, perform CHECK, FETCH, PARTIAL, SEARCH, COPY from mailbox)
- s - keep seen/unseen information across sessions (STORE SEEN flag)
- w - write (STORE flags other than SEEN and DELETED)
- i - insert (perform DELETED, COPY into mailbox)
- p - post (send mail to submission address for mailbox, not enforced by IMAP4 itself)
- c - create (CREATE new sub-mailboxes in any implementation-defined hierarchy)
- d - delete (STORE DELETED flag, perform EXPUNGE)
- a - administer (perform SETACL)

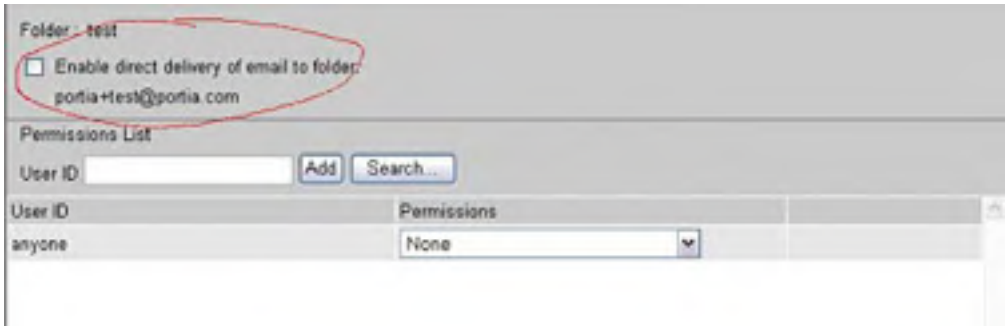
The web mail permissions correspond to the preceding as follows. The owner of the mail folder by default has all the rights (lrswipcda). Granting someone Read Only permissions gives that person the rights lrs. Read and Write permissions corresponds to lrswid; Read, Write, and Manage corresponds to lrswicda. TABLE 8-1 lists the mapping.

Table 8-1. Web Mail Permission and RFC2086 Rights

Web Mail	RFC2086
Owner gets these by default	lrswipcda
Read Only	lrs
Read and Write	lrswid
Read, Write, and Manage	lrswicda

If you are sharing a folder other than the Inbox, you will see an additional check box Enable direct delivery of email to folder, at the top of the permissions screen (FIGURE 8-3). When checked, this enables the post (p) privilege by anyone, so that mail addressed to *username+folder* @*host.domain* is delivered directly into this folder.

Figure 8-3. Sharing a Folder Other Than the Inbox



[Team LiB]

PREVIOUS NEXT

Limitations

You can only share a folder with another user who is on the same mailstore as you are.

Setup Procedures

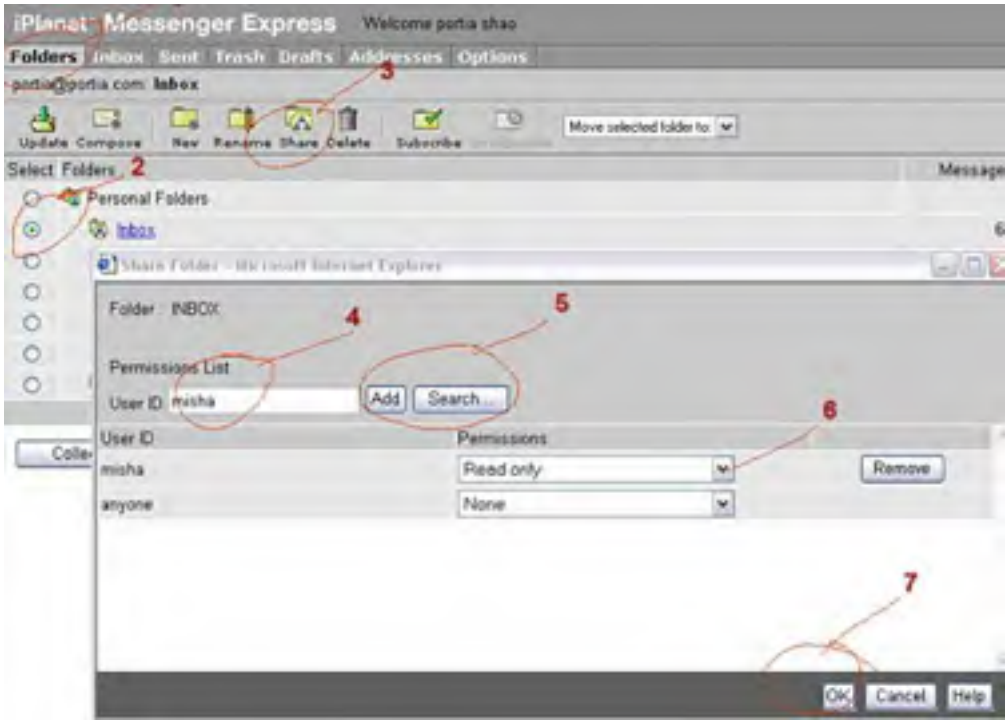
This section contains the following setup procedures:

- Letting Your Administrator Read Your Inbox
- Sharing Folders in MAP Clients
- Sharing a Folder in Mulberry
- Sharing a Folder in Netscape Messenger
- Using Outlook Express

Letting Your Administrator Read Your Inbox

Using web mail is pretty simple. We are assuming you (portia) and your administrator (misha) are on the *same* mailstore.

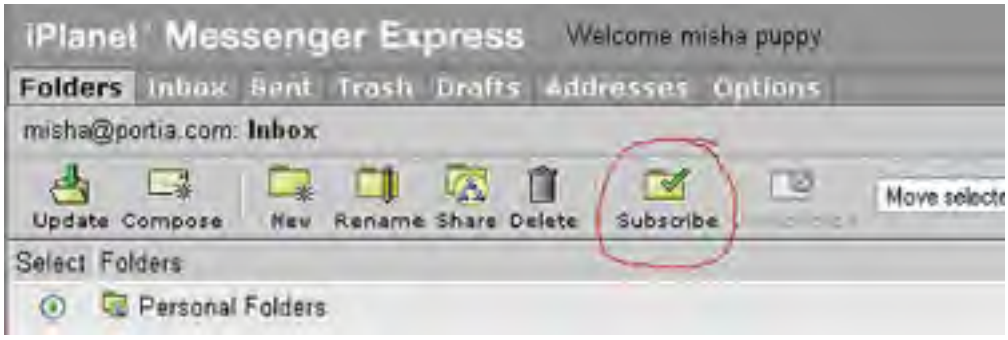
1. **Set the permission on your inbox to Read by your administrator.**



Note

The following steps are done as the administrator (misha). Ask your administrator to subscribe to your folder.

2. **Click the Subscribe button, then fill in the name. In this case, enter portia, who is sharing the folder with misha.**



3. **Click the Search button next to the name to find the right user and select the correct user.**

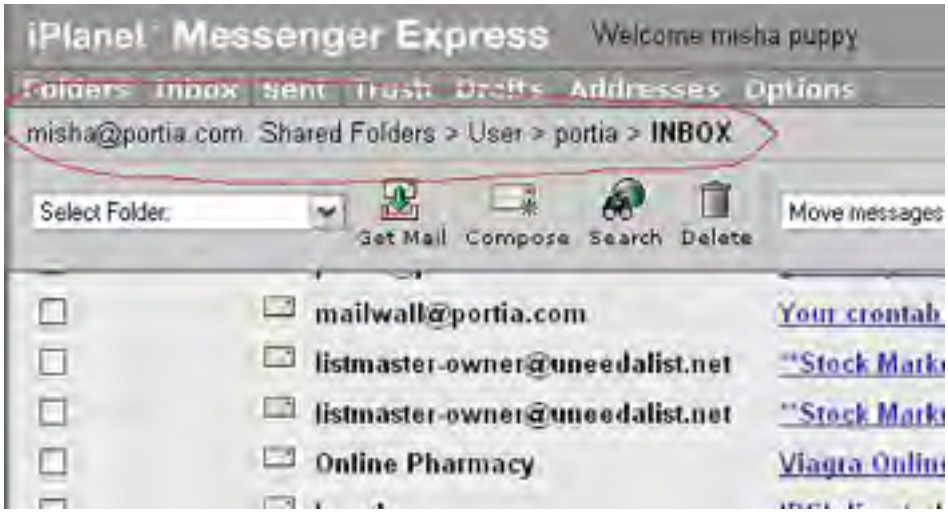
   You will be back on this screen where you will see the list of folders being shared.

4. **Click the Subscribe button to subscribe.**

   You should see the newly shared folder showing up in your list. If it does not, click Update to refresh the list.

5. **Double-click portia's inbox.**

   You see it as follows:

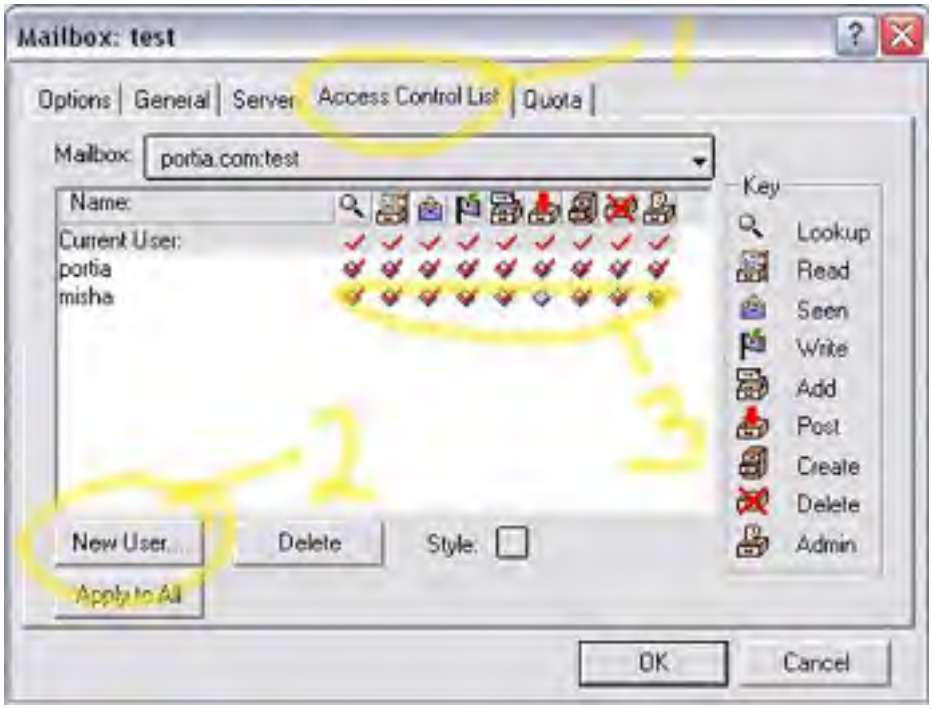


Sharing Folders in MAP Clients

Several IMAP clients allow you to share a folder with others, and allow you to view shared folders. Some examples include: Mulberry (www.cyrus.com), Netscape (4.7x shown here, but later versions also work), Mozilla, and Outlook Express. However, not all mail clients support this feature. Eudora 5.1 has its own version of shared folders.

Sharing a Folder in Mulberry

1. **To share one of your folders, right-click on the folder you want to share.**
2. **Select Properties to bring up the following window to edit the mailbox properties.**



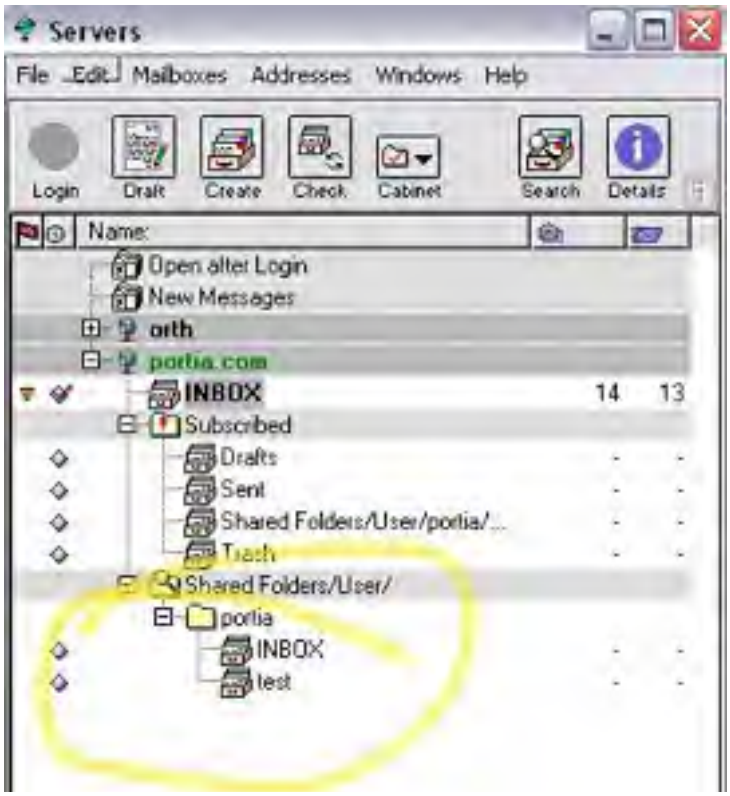
3. **Click the Access Control List tab.**

4. **Click the New User button and type in the login identifier. In this case, misha, the administrator, of the user who will be sharing your folder.**
5. **Select the appropriate check boxes for the access privileges you are granting.**

   The keys to the icons are shown on the same screen. (Note they are in the same order as described in RFC2086.)

When you first log in, you can tell Mulberry to show you the shared folders found on the server by selecting the Shared Folders/User/ mailboxes on the left side of the following window.

Now you will automatically see other people's shared folders.



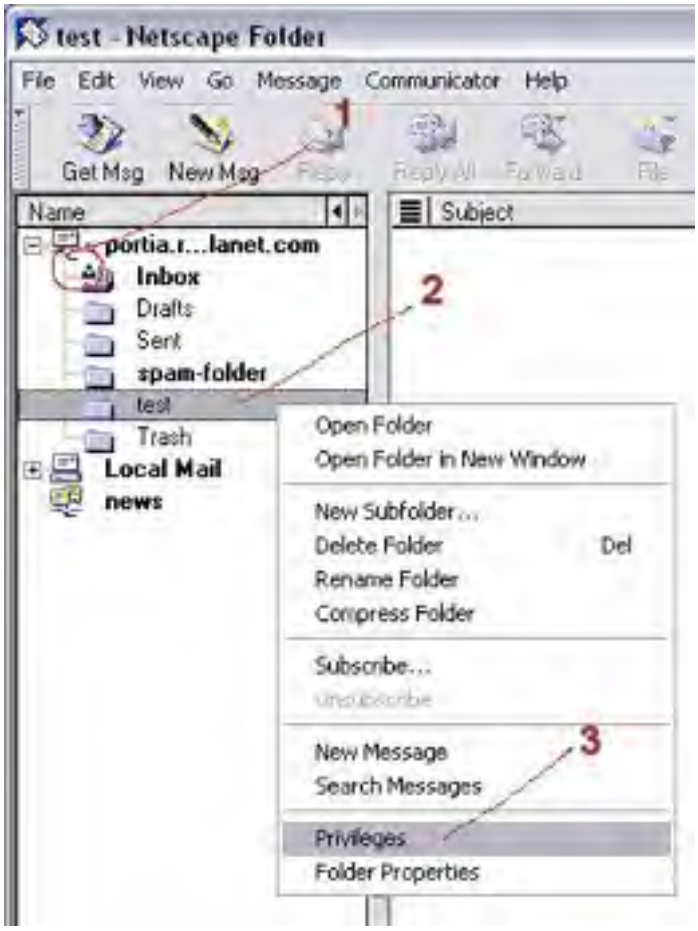
Sharing a Folder in Netscape Messenger

To share your folder with someone using Netscape Messenger:

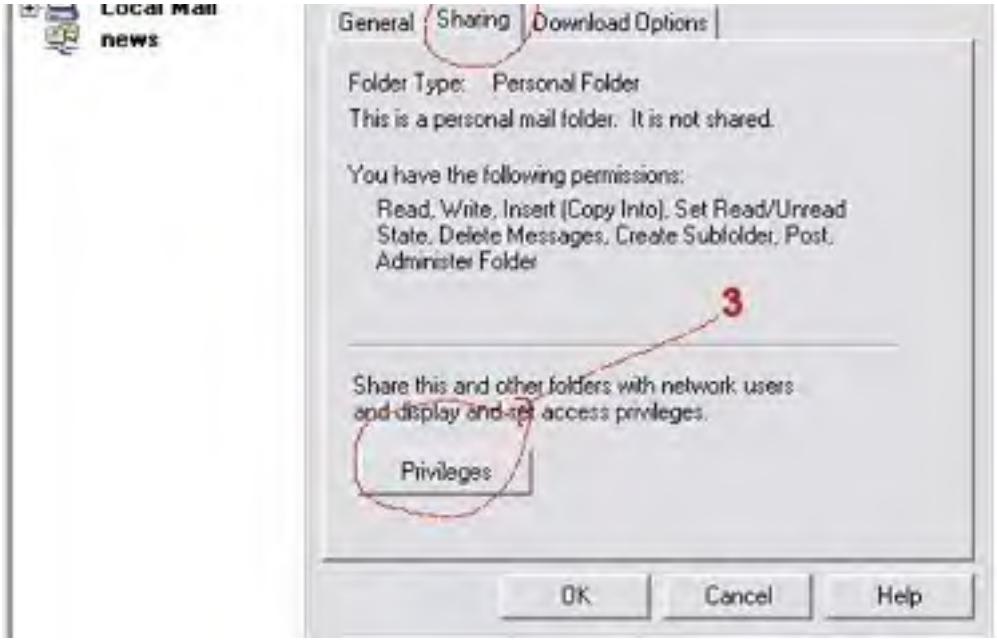
1. **Right-click on the folder to be shared to bring up the pop-up menu.**
2. **Select Privileges.**

Note

The little people icon on the folder shows that the folder is shared.



3. **If Privileges is grayed out, click on Folder Properties directly below it to bring up the Folder Properties window, then click on the Sharing tab and Privileges.**

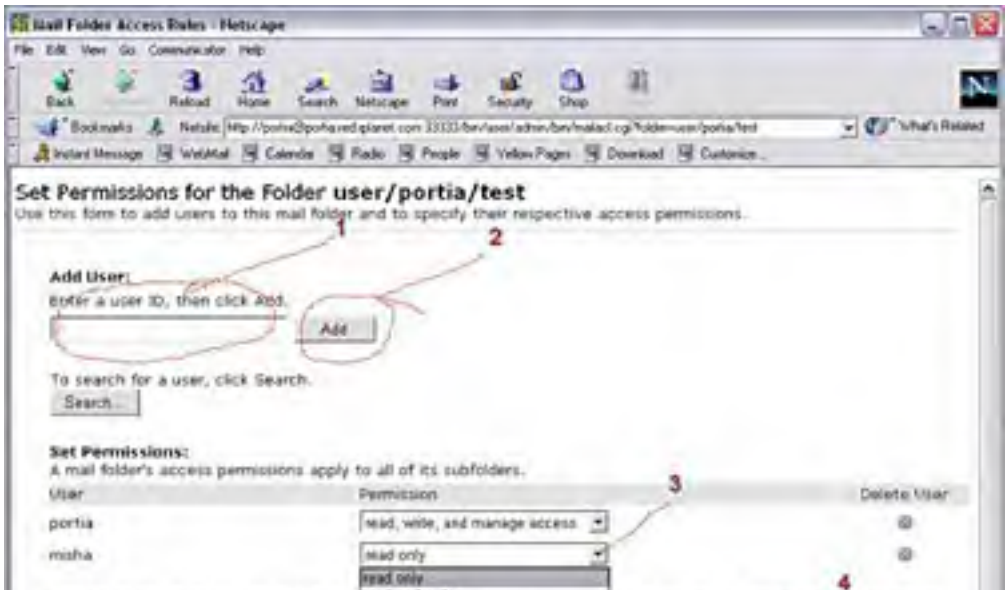


This brings up a separate Netscape browser window that asks you to login to the administration server.

4. **Use the same user name and password as for your mail account.**

   This window may not work well prior to Messenger Server version 5.2.

5. **After logging in, you will be shown a browser window where you can set the permissions for the folder.**



a. **Type in the user ID of the person who will share your folder. In this case type misha, and click the Add button.**

   Misha is shown as a user in the middle of the screen.

b. **Use the pull-down menu to select the permission you are granting.**

c. **Click OK to close the window when you are done.**

Now the administrator, misha, can log in using Netscape Messenger. Misha will see the following:

Using Outlook Express

With Outlook Express (2002), you can view folders others opted to share with you automatically if you have subscribed to them. However, there is no mechanism in Outlook Express to make a folder shareable, nor to subscribe to a shared folder. So you can either have the administrator subscribe to the folders using some other program, or the administrator can perform the following procedure.

Assuming you have made your folder readable by your administrator by using some other means, your administrator can set up an Outlook Express mail account as follows:

1. **Right-click on the mail server in question and select IMAP folders to display the dialog box.**

2. **Uncheck the box that determines whether only subscribed folders are seen, and click OK to close the window.**

3. **Exit Outlook Express and restart it to see the shared folders.**

   Now your administrator should see something like the following screen. Note that the Shared Folders hierarchy shows up in the middle of the folders list in alphabetical order.

This section of the book describes how to share a folder from an end user's point of view. It does not describe how to use the commands in RFC2086 directly. If you want to type the commands using Telnet, or are writing a program to do this, you should read the RFC2086 in its entirety. However, a very short dialog would look like this:

telnet *hostname* **143**
a login *username password*
b getacl inbox to see the acl on the inbox
c setacl inbox misha lrs to give misha "lrs" priv to my inbox
d deleteacl inbox misha to remove the acl set for misha on my inbox
z logout to log out when you are done

Chapter 9. Customization

Customers typically make several customizations right after getting the basic Messaging Server (Directory Server, Web Server, Delegated Administration, email, and perhaps even Calendar Server) installed. The most common of these include changing the "look and feel" of the web mail (IPlanet Messenger Express) interface and providing a single sign on (SSO) between the web mail, web-based calendar, and Delegated Administration interfaces. Some of the other common customizations that are done almost immediately include defining the welcome message for new accounts along with the over quota message for people about to go or already over quota. Some customers would also like to customize some of the return errors that the message system sends back to users.

The complete customization of the look and feel for Messenger Express is available in the manual (see http://docs.sun.com/source/816-6010-10/index.html for the *iPlanet Messenger Express 5.2 Customization Guide*). Most customers want to perform some very simple customizations for the look and feel of the Messenger Express:

- Changing and Adding a Logo
- Customizing the Login Screen
- Changing the Main Web Mail Screen Banner
- Removing and Adding Options on the Options Tab
- Single Sign On
- Setting the Initial Welcome Email
- Over-Quota Limits and Warning Email
- Customizing Return Errors

For additional details related to these changes, refer to the *iPlanet Messenger Express 5.2 Customization Guide*.

Changing and Adding a Logo

Most of the Sun ONE Messaging Express look and feel is controlled through HTML and JavaScript (also known as ECMA script), which is located in the following directory:

/*msg-Home*/msg-*Instance*/html

where *msg-Home* is the directory where the messaging software was installed, and *Instance* is the name of the messaging instance (install), often the *hostname* (short host name).

A quick look at the directory will tell you why the *iPlanet Messenger Express 5.2 Customization Guide* is necessary and a good thing.

[View full width]

```
root@sparc5-1:/A1000/demo6789/ims52/msg-sparc5-1/html> ls

applet_fs.html*         en/                  lookup.js*             searchmsg_fs.html*
spelltools.html*
ar/                     es/                  lower2.html*           searchusers.js*
spellword.html*
attach_fs.html*         fldr_fs.html*        main.js                setpermission_fs
.html*  srchresults_fs.html*
collect_fs.html*        form.js*             main.orig*             sk/
subscribe_fs.html*
colors.html*            fr/                  master-style.css*      sl/
th/
comp_fs.html*           frame.html*          mbox_fs.html*          spell.html*
tr/
compRecipient.js*       he/                  msg_fs.html*           spell.js*
upper.html*
cs/                     hr/                  opts_fs.html*          spell2.html*
util.js*
de/                     hu/                  pab.js*                spell2.js*
zh-CN/
editPabEntry.js*        imx/                 pl/                    spellchange.html*
zh-TW/
editPabGroup.js*        ja/                  receipt_fs.html*       spellresults.html*
el/                     ko/                  ro/                    spellSend.html*
emoticons.html*         ldap_fs.html*        sample.html*           spellsuggestions.html*
```

The first thing to note is the two-letter directories such as "en" or "de." These are language- specific directories, so "en" is English while "de" is German. The Messaging Server's Messenger Express interface is fully internationalized, supporting 20 or more different languages. Depending upon the customizations made and your audience, each of these *locales*, as they are called, will have to have the same customizations performed to them. This book only describes the main directory and the English (en) locale:

[View full width]

```
root@sparc5-1:/A1000/demo6789/ims52/msg-sparc5-1/html/en> ls

compRecipient_fs.html*  help.htm*            iplanet.jpg*           messageView.html*
searchusers_fs.html*
default.html*           help2.htm*           ix.htm*                pab_fs.html*
topics.htm*
editPabEntry_fs.html*   helpix.htm*          lookup_fs.html*        searchMessage.html*
editPabGroup_fs.html*   i18n.js*             mail.html*             searchOnly.html*
```

The customization guide provides solid information, but it often takes a very thorough and complete approach. This section provides a more practical and quick view of the changes, for several reasons:

1. I generally do not like to do more work than necessary.
2. Many of the changes in the customization guide require not only edits of the graphic files, but also of the HTML and JavaScript.
3. Every time a patch or update to the Messaging Server software is applied, your customizations must be redone because the update carries new HTML or JavaScript files for the Messenger Express. The method outlined here tends to survive better or at least is easier to apply after an update.

4. Size matters. Copying and editing the original graphics (gifs), rather than creating them from scratch or using something different, avoids issues where dimensions are hard-coded in the HTML or JavaScript, which avoids having to change these dimensions.

The downside to this approach is that the ALT tag fields do not get changed. However, these are fairly easy edits that even the most basic HTML coders can perform.

Only three optional graphics and one HTML file or one JavaScript file must be changed or customized. It is important that good change practices are followed—for example, keeping backups of the original files (versioning). While something like CSV is not quite necessary, if you are familiar with it and are using it for other programming projects, why not?

Graphics files that should be customized:

- /*msg-Home*/msg-*Instance*/html/imx/iplanet_logo.gif
- /<*msg-Home*>/msg-*Instance*/html/imx/WebMail_splash.gif
- /<*msg-Home*>/msg-*Instance*/html/imx/iplanetBanner.gif

Additional graphic files for the login page (the abstract graphic in the middle of the page):

- /*msg-Home*/msg-*Instance*/html/imx/left_strip_consumer_1.gif
- /*msg-Home*/msg-*Instance*/html/imx/center_strip_consumer_1.gif
- /*msg-Home*/msg-*Instance*/html/imx/right_strip_consumer_1.gif

HTML or JavaScript files that must be customized:

- /*msg-Home*/msg-*Instance*/html/en/default.html
- /*msg-Home*/msg-*Instance*/html/en/i18n.js

where *msg-Home* is the directory where the Messaging Server software was installed, and *Instance* is the name of the messaging instance (install), often the *hostname* (short host name).

Customizing the Login Screen

1. **Make backup copies of the original files you are going to edit:**

   # **cd /***msg-Home***/msg-***Instance***/html/imx/**
   # **cp iplanet_logo.gif iplanet_logo.gif.orig**
   # **cp WebMail_splash.gif WebMail_splash.gif.orig**
   # **cp iplanetBanner.gif iplanetBanner.gif.orig**
   # **cp left_strip_consumer_1.gif left_strip_consumer_1.gif.orig**
   # **cp center_strip_consumer_1.gif center_strip_consumer_1.gif.orig**
   # **cp right_strip_consumer_1.gif right_strip_consumer_1.gif.orig**
   # **cd ../en**
   # **cp default.html default.html.orig**
   # **cp i18n.js i18n.js.orig**

2. **Edit the three main graphics files using your favorite editor, such as GIMP.**

   Be careful to note that some of these files have transparent backgrounds, while others do not. You can easily transfer these files to your desktop by using ftp.

   a. The iplanet_logo.gif is an image 96 pixels wide x 66 pixels high on a white (255,255,255 RGB) background.

   b. The WebMail_splash.gif is an image of 450 pixels wide x 50 pixels high on a white (255,255,255 RGB) background.

   c. The iplanetBanner.gif is an image of 273 pixels wide x 27 pixels high on a transparent background.

   For consistency, you could have all of your new graphics on transparent backgrounds.

   Now that the files are edited, you can change some of the text; you can also edit this manually.

3. **Stop the Messaging Server before you make any changes.**

You do not have to do this, but it is generally a good idea. You must restart the services (server) to recognize the changes.

```
# cd /msg-Home/msg-Instance
# ./stop-msg
```

4. **Copy the** default.html **file to a scratch file:**

```
# cd /msg-Home/msg-Instance/html/en
# cp default.html default.tmp
# cp i18n.js i18n.tmp
```

5. **Using either an editor or a program like sed, change the occurrences of iPlanet to your organization. For example, the code for Acme University would be:**

[View full width]

```
# sed -e "s|www.iplanet.com|www.it.acme.edu|g" \
  -e "s|iPlanet e-commerce solutions|Acme University IT Group|g" \
  -e "s|iplanet.com|www.it.acme.edu|g" \
  -e "s|iPlanet Messenger Express|Acme University Web eMail    Service|g" \
  -e "s|iPlanet Messaging Server|Acme University Web eMail  Service|g" \default.tmp
  default.html
# sed -e "s|iPlanet Messenger Express|Acme University Webmail|g" \
  -e "s|Messenger Express|Webmail|g" < i18n.tmp > i18n.j
```

You changed a couple of URLs that refer to either www.iplanet.com or iplanet.com to the IT department's web site at Acme University. You also changed the title bar, the graphic ALT tag, and the main screen.

> **Note**
>
> The copyright notice must be changed manually.

6. **Restart the Messaging Server:**

```
# cd /msg-Home/msg-Instance
# ./start-msg
```

Changing the Main Web Mail Screen Banner

One often-requested item is an additional space on the main web mail screen—the screen you get once you have successfully logged in. Customers use this space to introduce their logos, banners, colors, and so forth. An easy way to include such information is to extend the basic frame set by adding an additional frame on top of the existing web mail frame.

A good example of this might be a partly transparent graphic that could be used in conjunction with coloring the frame's background in Acme University's school colors. Navigation buttons can be added too.

To add an extra frame, edit the mail.html file that contains the main layout of the web mail interface. This must be done in all versions of *lang*/mail.html. For the example, stick with the 'en' locale.

The mail.html is a very small file that plays a critical role because it controls the entire web mail interface.

1. **Stop the Message Server before you make any changes.**

   You do not have to do this, but it is generally a good idea. You must restart the services (server) for the system to recognize the changes.

```
# cd /msg-Home/msg-Instance
# ./stop-msg
```

2. **Back up the** mail.html **file:**

```
# cd /msg-Home/msg-Instance/html/en
# cp mail.html mail.html.orig
```

3. **Edit the** mail.html **file with your favorite editor, such as vi:**

vi mail.html

These are the lines that are of interest:

[View full width]

```
'<frameset border="0" frameborder="no" rows="0,*,0" onLoad="start()" onUnload="end()"
 onResize "change()">'+
'<frameset border="0" frameborder="no" cols="*,*,*,*,*">'+
'<frame name="cfgFrame" noresize scrolling="no" src="../frame.html?' + main.clientargs + '">'+
'<frame name="mboxFrame" noresize scrolling="no" src=     "../frame.html?' + main
 .clientargs + '">'+
'<frame name="cmdFrame" noresize scrolling="no" src="../frame.html?' + main.clientargs + '">'+
'<frame name="msgFrame" noresize scrolling="no" src="../frame.html?' + main.clientargs + '">'+
'<frame name="pabFrame" noresize scrolling="no" src="../frame.html?' + main.clientargs + '">'+
'</frameset>'+
'<frame name="mailFrame" marginwidth="0" marginheight="0" noresize src="../frame.html?' +
 main.clientargs + '">'+
'<frame name="appletFrame" marginwidth="0" marginheight="0" noresize src="../frame.html?'
 + main.clientargs + '">'+
'</frameset>'
```

To add an additional frame at the top of the page, add the following line or something similar:

'frame name="!-- *frame_name* **--" marginwidth="0" marginheight="0" noresize src="!--** *html source* **--">'+**

where *!-- frame_name --* is the name you want to add to the frame and *!-- html source --* is the HTML file you want this frame to include. So if you want to call the new frame "AcmeFrame" and its source is in the same directory but called acme.html, the additional line would look like:

'<frame name="acmeFrame" marginwidth="0" marginheight="0" noresize src="../acme.html">'+

This code is inserted right after the initial frame is defined, so the edited portion of the file looks like:

[View full width]

```
'<frameset border="0" frameborder="no" rows="20,*,0" onLoad="start()" onUnload="end()"
 onResize="change()">'+
'<frame name="acmeFrame" marginwidth="0" marginheight="0" noresize src="./acme.html">'+
'<frameset border="0" frameborder="no" cols="*,*,*,*,*">'+
'<frame name="cfgFrame" noresize scrolling="no" src="../frame.html?' + main.clientargs + '">'+
'<frame name="mboxFrame" noresize scrolling="no" src="../frame.html?' + main.clientargs +
 '">'+
'<frame name="cmdFrame" noresize scrolling="no" src="../frame.html?' + main.clientargs + '">'+
'<frame name="msgFrame" noresize scrolling="no" src="../frame.html?' + main.clientargs + '">'+
'<frame name="pabFrame" noresize scrolling="no" src="../frame.html?' + main.clientargs + '">'+
'</frameset>'+
'<frame name="mailFrame" marginwidth="0" marginheight="0" noresize src="../frame.html?' +
 main.clientargs + '">'+
'<frame name="appletFrame" marginwidth="0" marginheight="0" noresize src="../frame.html?'
 + main.clientargs + '">'+
'</frameset>'
```

Note that you must modify the "rows" value on the initial frame so that you can actually see the top frame. A nominal value should work, though some testing to determine the best value is warranted.

Once you have your changes saved and your new frame content "acme.html" completed, you can restart the messaging server:

```
# cd /msg-Home/msg-Instance
# ./start-msg
```

Caution

Often there are problems with loading pages (for example, blank screen once logged in) and other errors due to incorrect ownership and file permissions. Make sure the "owner" and "group" for the files you just modified or created are the same as the other files in the */msg-Home/*msg-*Instance*/html and other directories (for example, chown iplanet:email mail.html). The permissions should be set to 750 by using the chmod command (for example, chmod 750 mail.html).

Tip

A good diagnostic is to turn off caching in your browser so you always receive the latest changes from the server. Logging out of the web interface and back in again works sometimes. Stopping and restarting the messaging server works sometimes too.

Note

The HTTP engine that is bundled as part of the Messaging Server is not a full-fledged web server. So some of the advanced HTML and JavaScript server side directives are not supported or can lead to strange results. When in doubt, keep it simple—things that work with the older browsers such as Netscape 4.78 tend to work just fine.

Removing and Adding Options on the Options Tab

Removing and adding and options on the Options Tab and the ability to change the URL for the password change function are very closely related. Why is the ability to add or remove options important? Occasionally, institutions do not want users to change their personal information in the system directory; there may be a business or official process in the HR department or Registrar's Office to accomplish this so that the information gets updated everywhere. Ideally, applications and other software would rely upon the directory. However, that is not always the case.

Removing Options

Removing (commenting out) the existing options is the easiest of all the changes to make. To remove options from the options tab, find the function toggleFrameHTML (starts around line 150) in opts_fs.html file which is in:

/*msg-Home*/msg-*Instance*/html/

where *msg-Home* is the directory where the messaging software was installed, and *Instance* is the name of the messaging instance (install), often the *hostname* (short host name).

Comment out the getToggle() statement for each of the following options shown—the comment characters (//) start each line of the code that is to be commented out, for example:

```
// comment ** Copyright 2003 Sun Microsystems, Inc.
```

The removable options within the JavaScript code that can be commented out include:

```
Account Summary:

  getToggle(main.i18n['account summary'], 'summary',
  'javascript:parent.toggle(\'summary\')') +

Personal Information:

  getToggle(main.i18n['personal'], 'personal',
  'javascript:parent.toggle(\'personal\')') +
Change Password:
  getToggle(main.i18n['password'], 'password',
  'javascript:parent.toggle(\'password\')') +

Settings:

  getToggle(main.i18n['settings'], 'settings',
  'javascript:parent.toggle(\'settings\')') +

Appearance:

  getToggle(main.i18n['appearance'], 'appearance',
  'javascript:parent.toggle(\'appearance\')') +

Vacation Message:

  getToggle(main.i18n['vacation'], 'vacation',
  'javascript:parent.toggle(\'vacation\')') +
```

For example, to comment out the ability to change personal information:

1. **Stop the Messaging Server before you make any changes.**

   You do not have to do this, but it is generally a good idea. You must restart the services (server) to recognize the changes.

   # **cd /***msg-Home***/msg-***Instance*
   # **./stop-msg**

2. **Back up the** opts_fs.html **file:**

   # **cd /***msg-Home***/msg-***Instance***/html**

```
# cp opts_fs.html opts_fs.html.orig
```

3. **Edit the** opts_fs.html **file with your favorite editor, such as vi:**

```
# vi opts_fs.html
```

The lines of interest are:

```
getToggle(main.i18n['personal'], 'personal',
'javascript:parent.toggle(\'personal\')') +
```

Which will change to:

```
// commented out 01/09/03 by dbp
//
// getToggle(main.i18n['personal'], 'personal',
// 'javascript:parent.toggle(\'personal\')') +
//
```

Note

You do not have to go into a *language* directory such as "en" to change the options page. That is because this page is fully internationalized and uses variables that are set when a person logs in. So the actual text of Personal Information is not set within this JavaScript or HTML, it is set to whatever language you have configured for the default or a particular user. This also means you do not have to modify this page over and over again for each language that you use.

Once you have saved your updated opts_fs.html, you can restart the messaging server:

```
# cd /msg-Home/msg-Instance
# ./start-msg
```

Adding Options

Now that you have successfully removed (commented out) an option from the Options Tab, the next customization that many customers like to do is to add an option. Unfortunately it is not quite as easy as commenting out a few lines, but it is not difficult either.

1. **Stop the Messaging Server:**

```
# cd /msg-Home/msg-Instance
# ./stop-msg
```

2. **Back up the** opts_fs.html **file. You must be careful as you have already made some changes:**

```
# cd /msg-Home/msg-Instance/html
# cp opts_fs.html opts_fs.html.orig
```

This is where good change control management and using something like CSV pays off and really adds value.

3. **Edit the** opts_fs.html **file with your favorite editor, such as vi:**

```
# vi opts_fs.html
```

a. **Concentrate on two areas: the** toggleFrameHTML **function around line 150 and adding a custom action to be triggered by the** toggleFrameHTML.

Here is the toggleFrameHTML function after the previous edit:

```
function toggleFrameHTML() {
  return main.getBody(main.chrome2, true, main.black, main.link0,
  main.link1, main.chrome2) +
  '<center>\n<table border=0 cellspacing=7 cellpadding=0 width=100%>\n' +
getToggle(main.i18n['account summary'], 'summary',
'javascript:parent.toggle(\'summary\')') +
// getToggle(main.i18n['personal'], 'personal',
// 'javascript:parent.toggle(\'personal\')') +
getToggle(main.i18n['password'], 'password',
'javascript:parent.toggle(\'password\')') +
(main.cfgFrame.mbox.length == 0 ? '' :
getToggle(main.i18n['settings'], 'settings',
'javascript:parent.toggle(\'settings\')')) +
getToggle(main.i18n['appearance'], 'appearance',
'javascript:parent.toggle(\'appearance\')') +
getToggle(main.i18n['vacation'], 'vacation',
'javascript:parent.toggle(\'vacation\')') +
getToggle(main.i18n['NDA'], 'NDA',
'javascript:parent.toggle(\'NDA\')') +
'</table>\n</center>\n'
}
```

b. **Add an option called Yahoo.**

This option opens up a separate browser window by using JavaScript with the URL http://www.yahoo.com:

```
function toggleFrameHTML() {
   return main.getBody(main.chrome2, true, main.black, main.link0,
   main.link1, main.chrome2) +
   '<center>\n<table border=0 cellspacing=7 cellpadding=0 width=100%>\n' +
   getToggle(main.i18n['account summary'], 'summary',
   'javascript:parent.toggle(\'summary\')') +
// getToggle(main.i18n['personal'], 'personal',
// 'javascript:parent.toggle(\'personal\')') +
   getToggle(main.i18n['password'], 'password',
   'javascript:parent.toggle(\'password\')') +
   (main.cfgFrame.mbox.length == 0 ? '' :
   getToggle(main.i18n['settings'], 'settings',
   'javascript:parent.toggle(\'settings\')')) +
   getToggle(main.i18n['appearance'], 'appearance',
   javascript:parent.toggle(\'appearance\')') +
   getToggle(main.i18n['vacation'], 'vacation',
      'javascript:parent.toggle(\'vacation\')') +
// added the following option
    getToggle('Yahoo!', 'yahoo',
      'javascript:parent.toggle(\'yahoo\')') +
//
    getToggle(main.i18n['NDA'], 'NDA',
      'javascript:parent.toggle(\'NDA\')') +
      '</table>\n</center>\n'
}
```

You could have just as easily done this to point to the main institution web page or even a change password application (more on that later).

Note the three fields:

1. The label of the option as it appears—"Yahoo!"
2. The name of the option for tracking—"yahoo"
3. Action to take when clicked—javascript:parent:toggle(\'yahoo'\), which is normally passed to the following function which eventually runs the yahooHTML() function within opts_fs.html.

c. **Modify the** listFrameHTML() **function to trigger the choice:**

```
function listFrameHTML() {
  var s = main.getBody(main.white, true, main.black, main.link0,
  main.link1, main.link2, 6, 8)

    if (main.option_page == 'appearance') {
    s += appearanceHTML()
  } else if (main.option_page == 'password') {
    s += passwordHTML()
  } else if (main.option_page == 'personal') {
    s += personalHTML()
  } else if (main.option_page == 'settings') {
    s += settingsHTML()
  } else if (main.option_page == 'summary') {
    s += summaryHTML()
  } else if (main.option_page == 'vacation') {
    s += vacationHTML()
//
  } else if (main.option_page == 'yahoo') {
    s += yahooHTML()
//
  } else if (main.option_page == 'NDA') {
    s = ndaHTML()
  }
return s

}
```

d. **Build a** yahooHTML() **function. The easiest way is to copy the** ndaHTML() **function and modify it:**

[View full width]

```
function ndaHTML() {
 return '<HTML><HEAD></HEAD><BODY ONLOAD=\"location.href = \" +  main.NDAStartPage + '\'
 \"></BODY></HTML>'
}
```

This function becomes:

[View full width]

```
// added for yahoo by dbp
function yahooHTML() {
 return '<HTML><HEAD></HEAD><BODY ONLOAD=\"location.href = \'http:\/\/www.yahoo.com\'\"><
 /BODY></HTML>'
}
```

Note the use of the backslash (\) character which allows an escape so that JavaScript does not think the // part of http://www.yahoo.com is a comment as well as the backslashes (\) preceding the straight quotes (' and **"**). You could have just as easily made this any URL or web application.

This function will pull the web page into the existing frame if possible—for example, it will do this if the port or protocol changes, such as https instead of http. To get the web page in a separate window, you must use the JavaScript open.window command.

After you save your updated opts_fs.html, you can restart the messaging server and check your changes.

For a pop-up window:

[View full width]

```
// added to do popup window by dbp
function yahooHTML() {
 return '<HTML><HEAD></HEAD><BODY ONLOAD=\"window.open(\'http:\/\/www.yahoo.com\', \'test
 \', \'scrollbars=yes,menubar=yes,toolbar=yes,status=yes\')\"></BODY></HTML>'
}
```

As you can see, modifying these options is fairly easy. The last customization is to change how the *change password* functionality works—so rather than put up a page to change the password, you can call an external application.

The easiest way to do this is to comment out (**//**) the existing passwordHTML() function:

```
// function passwordHTML() {
//   return '<form name="form">' +
//     '<table border=0 cellpadding=3 cellspacing=0>' +
//     '\n<tr>\n<td colspan=2>' +
//     main.font(3) + '<b>' + i18n['password'] +
//     '</b></font>' +
//     '<br>' + main.font() + i18n['passwd exp'] +
//     '</td>\n</tr>' +
//     '\n<tr>\n<td colspan=2>' +
//     '<table border=0 cellpadding=0 cellspacing=0 width=100% ' +
//     main.cellBgString + '><tr><td>' +
//     '<img src="imx/spacer.gif" width=1 height=2>' +
//     '</td></tr></table>' +
//     '</td>\n</tr>' +
//     '\n<tr>\n<td' + main.base_line + ' width=1% nowrap>' +
//     main.font() + i18n['passwd old'] + nbsp +
//     '</td>\n<td>' +
//     '<input type="password" name="old">' +
//     '</td>\n</tr>' +
//     '\n<tr>\n<td' + main.base_line + ' width=1% nowrap>' +
//     main.font() + i18n['passwd new'] + nbsp +
//     '</td>\n<td>' +
//     '<input type="password" name="newpass">' +
//     '</td>\n</tr>' +
//     '\n<tr>\n<td' + main.base_line + ' width=1% nowrap>' +
//     main.font() + i18n['passwd confirm'] + nbsp +
//     '</td>\n<td>' +
//     '<input type="password" name="confirm"> ' +
//     '</td>\n</tr>' +
//     '\n<tr>\n<td colspan=2>' + nbsp +
//     '</td></tr>' +
//     '<tr align=center width=100%><td colspan=2>' +
//     '<table border=0 cellpadding=4 cellspacing=0><tr>' +
//     main.button(i18n['passwd submit'], 'parent.validate()') +
//     main.button(i18n['clear'], 'parent.clear()') + '</tr></table>' +
//     '</td></tr>' +
//     '</td>\n</tr>' +
//     '</table></form>'
//     '</td>\n<td>' +
//     '<input type="password" name="newpass">' +
//     '</td>\n</tr>' +
//     '\n<tr>\n<td' + main.base_line + ' width=1% nowrap>' +
//     main.font() + i18n['passwd confirm'] + nbsp +
//     '</td>\n<td>' +
//     '<input type="password" name="confirm"> ' +
//     '</td>\n</tr>' +
//     '\n<tr>\n<td colspan=2>' + nbsp +
//     '</td></tr>' +
//     '<tr align=center width=100%><td colspan=2>' +
//     '<table border=0 cellpadding=4 cellspacing=0><tr>' +
//     main.button(i18n['passwd submit'], 'parent.validate()') +
//     main.button(i18n['clear'], 'parent.clear()') + '</tr></table>' +
//     '</td></tr>' +

//     '</td>\n</tr>' +
//     '</table></form>'
// }
```

Substitute your own passwordHTML() function:

[View full width]

```
// added to call external password update-change webpage

function passwordHTML() {
 return '<HTML><HEAD></HEAD><BODY ONLOAD=\"window.open(\'http:\/\/changepwd.acme.edu\',
 \'chgpwd\', \'scrollbars=yes\')\"></BODY></HTML>'
}
```

Beyond the basic modifications of the Options menu, customers also provide additional validation criteria for passwords (for example, not in dictionary, must contain special characters). This customization involves modifying the validate() function—perhaps calling a special JavaScript function that you want to use over and over (see main.js).

Single Sign On

One change that most customers make initially is to enable single sign on between the Messenger Express (web mail) and the Delegated Administrator function. The out-of-the-box functionality is that the Delegated Administrator link from the Options Tab in web mail pops up a separate window with the login box. Configuring the messaging server for SSO still pops up a separate window, but bypasses the login screen because the server knows who the user is and that the user has been properly authenticated. The SSO is achieved by using cookies and session IDs generated by the Messaging Server or other application such as Delegated Administrator or even the Calendar Server.

Enabling Single Sign ON

The following steps are required when the Messaging Server is running:

1. **Use the** su **command to go to** *mailsrv,* **where** *mailsrv* **is the UNIX or system user ID under which the Messaging Server is running.**

   Since you used "nobody" as the system user ID during the install:

   ```
   # su - nobody
   ```

2. **Change to the messaging instance for which you want to enable SSO, /***msg-Home***/msg-***Instance***/:**

   ```
   # cd /msg-Home/msg-Instance/
   ```

   where *msg-Home* is the directory where the messaging software was installed, and *Instance* is the name of the messaging instance (install), often the *hostname* (short host name).

3. **Check the existing settings for web mail:**

   ```
   # ./configutil | grep webmail
         local.webmail.da.host = sparc5-1.central.sun.com
         local.webmail.da.port = 88
         local.webmail.sso.enable = 0
         local.webmail.sso.singlesignoff = 0
   ```

4. **Enable SSO and single sign off.**

   ```
   # ./configutil -o local.webmail.sso.enable -v 1
         OK SET
   # ./configutil -o local.webmail.sso.singlesignoff -v 1
         OK SET
   ```

   Single sign off cancels the SSO so that when someone clicks the logout link on any SSO-enabled application, the user's session ID and cookie go away.

5. **Configure the SSO prefix or group.**

   The SSO prefix or group provides a way for multiple SSO groups to all reside on the same system, which becomes part of the browser cookie.

   ```
   # ./configutil -o local.webmail.sso.prefix -v ssogrp1
         OK SET
   ```

   The ssogrp1 is the default for the Delegated Administrator and other applications, so you can use that, but you could also use something like *foobar*, however, you would have to change the default in the other Sun ONE products.

6. **Configure the application ID.**

   The application ID identifies the web mail to other applications.

   ```
   # ./configutil -o local.webmail.sso.id -v ims5
         OK SET
   ```

7. **Configure the domain of the cookie.**

   This domain must match the domain name used by the browser or client to access the web mail system—it must start with the period (.) and be a real domain, not a hosted or virtual domain.

   # **./configutil -o local.webmail.sso.cookiedomain -v ".central.central.com"**
   OK SET

8. **Configure the URL for verification of SSO for IDA.**

   The IDA is the application name, much like ims5.

   # **./configutil -o local.sso.ida.verifyurl -v "http://sparc5-1.central.sun.com:88/VerifySSO?"**
   OK SET

9. **Configure SSO for calendar (optional).**

   It will be called "ics50"—plus the port for calendar.

   # **./configutil -o local.sso.ics50.verifyurl -v "http://sparc5 1.central.sun.com:81/VerifySSO?"**
   OK SET

10. **Check the settings again:**

    # **./configutil | grep webmail**
    local.webmail.da.host = sparc5-1.central.sun.com
    local.webmail.da.port = 88
    local.webmail.sso.cookiedomain = .central.central.com
    local.webmail.sso.enable = 1
    local.webmail.sso.id = ims5
    local.webmail.sso.prefix = ssogrp1
    local.webmail.sso.singlesignoff = 1

    # **./configutil | grep sso**
    local.sso.ics50.verifyurl = http://sparc5-1.central.sun.com:81/VerifySSO?
    local.sso.ida.verifyurl = http://sparc5-1.central.sun.com:88/VerifySSO?
    local.webmail.sso.cookiedomain = .central.central.com
    local.webmail.sso.enable = 1
    local.webmail.sso.id = ims5
    local.webmail.sso.prefix = ssogrp1
    local.webmail.sso.singlesignoff = 1

11. **Restart the web mail Messaging Server as root.:**

    # **su -**
    # **cd /***msg-Home***/msg-***Instance***/**
    # **./stop-msg http**
    # **./start-msg http**

12. **Add a proxy user to the directory so SSO can look up users:**

    # **ldapadd -h sparc5-1.central.sun.com -D "cn=Directory Manager" -w eatbeef -v -f proxy.ldif**
    add objectclass:
    top
    person
    organizationalperson
    inetorgperson
    add uid:
    proxy
    add givenname:
    Proxy
    add sn:

    Auth
    add cn:
    Proxy Auth
    add userpassword:
    proxypassword
    adding new entry uid=proxy, ou=people, o=sparc5-1.central.sun.com, o=isp

```
        modify complete
```

where sparc5-1.central.sun.com is the host on which the directory server is running;

where *eatbeef* is the directory manager password;

where proxy.ldif is a file with the following:

```
dn: uid=proxy, ou=people, o=sparc5-1.central.sun.com, o=isp
        objectclass: top
        objectclass: person
        objectclass: organizationalperson
        objectclass: inetorgperson
        uid: proxy
        givenname: Proxy
        sn: Auth
        cn: Proxy Auth
        userpassword: proxypassword
```

where *proxypassword* is the password for this user.

13. **Add the access control information (ACI) for the proxy auth user:**

[View full width]

```
#  ldapmodify -h sparc5-1.central.sun.com -D "cn=Directory Manager" -w eatbeef -v -f aci1.ldif
      add aci:
            (target="ldap:///o=isp")(targetattr="*")(version 3.0; acl "proxy";allow
➥ (proxy) userdn="ldap:///uid=proxy, ou=people, o=sparc5-1.central.sun.com, o=isp";)
      modifying entry o=isp
      modify complete
```

where the file aci1.ldif contains the following:

[View full width]

```
dn: o=isp
     changetype: modify
     add: aci
   aci: (target="ldap:///o=isp")(targetattr="*")(version 3.0; acl "proxy";allow (proxy)
➥  userdn="ldap:///uid=proxy, ou=people, o=sparc5-1.central.sun.com, o=isp";)

     #  ldapmodify -h sparc5-1.central.sun.com -D "cn=Directory Manager" -w eatbeef -v -f
➥  aci2.ldif
     add aci:
         (target="ldap:///o=internet")(targetattr="*")(version 3.0; acl "Allow iDA User
➥  Proxy";allow (proxy) userdn="ldap:///uid=proxy, ou=people, o=sparc5-1.central.sun.com,
➥  o=isp";)
     modifying entry o=internet
     modify complete
```

where the file aci2.ldif contains the following:

[View full width]

```
dn: o=internet
    changetype: modify
    add: aci
    aci: (target="ldap:///o=internet")(targetattr="*")(version 3.0; acl "proxy";allow
➥ (proxy) userdn="ldap:///uid=proxy, ou=people, o=sparc5-1.central.sun.com, o=isp";)
```

14. **Go to the directory where the Delegated Administrator resource file is located.**

```
# cd /ida-Home/nda/classes/netscape/nda/servlet
```

where *ida-Home* is the location where Delegated Administrator was installed, for the demo system it is:

```
# cd /A1000/demo6789/ida12/nda/classes/netscape/nda/servlet
```

15. **Edit the** resource.properties **files as follows:**

```
# cp resource.properties resource.properties.orig
# vi resource.properties
```

Several changes must be made in this file:

```
> #LDAPDatabaseInterface-ldapauthdn=
# diff resource.properties resource.properties.orig

    514c514
    < NDAAuth-applicationId=ida
    ---
    > NDAAuth-applicationId=nda45

    526,528c526
    < verificationurl-ssogrp1-ida=http://sparc5-1.central.sun.com:88/VerifySSO?
    < verificationurl-ssogrp1-ims5=http://sparc5-1.central.sun.com:80/VerifySSO?
    < verificationurl-ssogrp1-ics50=http://sparc5-1.central.sun.com:81/VerifySSO?
    ---
    > #verificationurl-ssogrp1-nda45=http://localhost:80/VerifySSO?

    542,543c540,541
    < LDAPDatabaseInterface-ldapauthdn=uid=proxy,ou=people,o=sparc5-1.central.sun.com,o=isp
    < LDAPDatabaseInterface-ldapauthpw=proxypassword
    ---
```

The first change is the name to which the Delegated Administrator is referred within the SSO context—from nda45 to the value you gave it by using the configutil command (see Step 8).

Next, you added verification URLs for each of the applications you would like SSO enabled—mail (ims5), calendar (ics50), and delegated administrator (ida), as you called them in Steps 6, 8, and 9. These must match!

Finally, you uncommented out the *ldapauthdn* and *ldapauthpw* variables and used the user that you created in Step 12.

16. **Change the properties for the web server.**

    You must make the change because the Delegated Administator is really a web application.

    ```
    # cd /web-HOME/INSTANCE/config
    ```

    where *web-HOME* is the install directory for the web server and *INSTANCE* is the specific web server instance you are configuring. This is likely to contain the fully qualified name of the host. In the demo system it is:

    ```
    # cd /A1000/demo6789/iws60/https-sparc5-1.central.sun.com/config
    ```

17. **Edit the** servlets.properties **and the** context.properties **files:**

    ```
    # cp servlets.properties servlets.properties.orig
    # cp context.properties context.properties.orig

    # vi servlets.properties
    ```

    a. **Uncomment (remove the beginning # character from) each line in the** servlets.properties **file that contains** servlet.*.context=ims50.

       There should about 16 of these lines:

       ```
       #grep =ims50 servlets.properties
       #To enable single signon uncomment all the servlet.*.context=ims50 lines
       #servlet.Debug.context=ims50
       #servlet.Version.context=ims50
       #servlet.auth.context=ims50
       #servlet.cauth.context=ims50
       #servlet.getPage.context=ims50
       #servlet.getBin.context=ims50
       #servlet.cosMgr.context=ims50
       #servlet.userCosMgr.context=ims50
       #servlet.getLocation.context=ims50
       #servlet.TaskManager.context=ims50
       #servlet.logout.context=ims50
       #servlet.CLIMap.context=ims50
       ```

```
#servlet.CLISearch.context=ims50
#servlet.userSsrMgr.context=ims50
#servlet.ssoauth.context=ims50
#servlet.VerifySSO.context=ims50
```

b. **Edit the** context.properties **file:**

```
# vi context.properties
```

Add this line near the end of the file, just before the #IDACONF-Start:

```
context.ims50.sessionCookie=ssogrp1-ida
```

The ssogrp1-ida, must match the prefix and the name set in Steps 5 and 8.

18. **Restart the web server:**

```
# cd /web-HOME/INSTANCE
# ./stop
  shutdown: server shut down
# ./start
  iPlanet-WebServer-Enterprise/6.0SP2 B11/13/2001 00:49
  [LS ls1] http://sparc5-1.central.sun.com, port 88 ready to accept requests
  startup: server started successfully
```

Setting the Initial Welcome Email

Often you want to have an email that contains some basic information waiting for a new user. While this feature is available from both the command line and the administrator's console, the documentation often only provides examples for the administrator's console, as it is much easier to do from the console. Many customers want to configure this from the command line.

For more details, refer to Chapter 2 of the *Sun ONE Messaging Server Administration Guide*.

The following steps are required when the Messaging Server is running:

1. **Use** su **to go to** *mailsrv,* **where** *mailsrv* **is the UNIX or system user ID under which the Messaging Server is running. Since you used "nobody" during the install:**

   # **su - nobody**

2. **Change to the messaging instance for which you want to enable SSO,** **/***msg-Home***/msg-***Instance***/:**

   # **cd /***msg-Home***/msg-***Instance***/**
   # **cd /A1000/demo6789/ims5/msg-sparc5-1**

   where *msg-Home* is the directory where the messaging software was installed, and *Instance* is the name of the messaging instance (install), often the *hostname* (short host name).

3. **Check the existing settings for the welcome message:**

   # **./configutil | grep gen.newuserforms**

4. **Edit or create a welcome message (email format including headers—minimum of subject):**

   # **vi welcome.txt**

   You must at least have a "Subject: {subject}" header.

   Example:

   "Subject: Welcome!

   This is a welcome message."

   is OK, but not:

   "This is a welcome message."

5. **Set the welcome message.**

   # **./configutil -o gen.newuserforms -v < welcome.txt**

Over-Quota Limits and Warning Email

Often, an administrator wants to set a limit on users for how much storage the messages can consume and how many messages they can retain. This is referred to as quota. The Messaging Server offers the ability to limit a user on both how much storage (for example, bytes) and how many (quantity) messages are allowed. The system also provides a method for notifying users that they are running out of quota or are in danger of going over the limit, as well as providing a grace period so that even though they are over, they can still receive a little bit over their quota until such time as they log in and delete email.

You can set these limits by using the console or from the command line. Using the administration console is easier, but some customers prefer to do everything from the command line.

For details, refer to Chapter 11 of the *Sun ONE Messaging Server Administration Guide*.

Configuring Over-Quota Limits and Warning Email

The following steps are required when the Messaging Server is running to begin configuring the quota, warning message, and grace period:

1. **Use** su **to go to** *mailsrv,* **where** *mailsrv* **is the UNIX or system user ID under which the Messaging Server is running.**

   Since you used "nobody" as the system user ID during the install:

   ```
   # su - nobody
   ```

2. **Change to the messaging instance for which you want to change the user quota,** */msg-Home/msg-Instance/:*

   ```
   # cd /msg-Home/msg-Instance/
   # cd /A1000/demo6789/ims5/msg-sparc5-1
   ```

   where *msg-Home* is the directory where the messaging software was installed, and *Instance* is the name of the messaging instance (install), often the *hostname* (short host name).

3. **Check the existing settings for the quota:**

   ```
   # ./configutil | grep quota
   ```

4. **Configure a default user quota in terms of space:**

   ```
   # ./configutil -o store.defaultmailboxquota -v quota
   ```

   where *quota* is the quota expressed in bytes.

   - To set a default limit of 10 megabytes or 10,240,000 bytes:

     ```
     # ./configutil -o store.defaultmailboxquota -v 10240000
       OK SET
     ```

   - To configure a default user quota for the total number of messages:

     ```
     # ./configutil -o store.defaultmessagequota -v quota
     ```

     where *quota* indicates the maximum number of messages.

   - To set a default limit of 100 messages:

     ```
     # ./configutil -o store.defaultmessagequota -v 100
       OK SET
     ```

5. **Verify changes:**

```
# ./configutil | grep quota
```

6. **Configure quota enforcement and notification:**

```
# ./configutil -o store.quotaenforcement -v yes
      OK SET
# ./ configutil -o store.quotanotification -v yes
       OK SET
```

This step turns on the quota and the messages to users that they are exceeding the quota.

7. **Configure the actual message:**

```
# ./configutil -o store.quotaexceededmsg -v msg
```

where *msg* is the email message to the users when quota is exceeded. The message must have at least a subject line:

```
# ./configutil -o store.quotaexceededmsg -v < msg.txt
```

To configure how often a reminder is sent to the users:

```
# ./configutil -o store.quotaexceedmsginterval -v days
```

where *days* is the number of days between reminders.

To configure a daily reminder:

```
# ./configutil -o store.quotaexceedmsginterval -v 1
```

8. **Notify users in advance of the upcoming quota limit.**

Notification depends upon your company's protocol.

```
# ./configutil -o store.quotawarn -v percent
```

where *percent* is that threshold for the warning.

To configure a warning message at 90 percent of quota:

```
# ./configutil -o store.quotawarn -v 90
```

9. **Set the grace period—how long messages are held for users that are over quota.**

During the grace period, the messages are held in the queue. They are not delivered to the mailboxes.

```
# ./configutil -o store.quotagraceperiod -v hours
```

where *hours* is the number of hours over-quota messages will be held.

To configure a grace period of three days:

```
# ./configutil -o store.quotagraceperiod -v 72
```

10. **Check the existing settings for the quotas:**

```
# ./configutil | grep quota
```

Customizing Return Errors

Occasionally, customers like to configure or customize the return errors provided to other SMTP servers. While this is generally frowned upon, legitimate reasons do exist—to provide additional information, to provide system administrator contact information, and so forth.

The return messages are localized to a point—depending upon which version of the messaging software was installed and the level of customization, you might have German, Spanish, French, and English, so you may have to modify several files. This book only describes the English (en) locale.

The return codes are stored in the following directory:

/*msg-HOME*/msg-*Instance*/imta/config/locale/C/LC_MESSAGES

where *msg-Home* is the directory where the messaging software was installed, and *Instance* is the name of the messaging instance (install), often the *hostname* (short host name).

```
# cd /A1000/demo6780/ims52/msg-sparc5-1/imta/config/locale/C/LC_MESSAGES
```

If you look at this directory, you can see the files for the various responses:

[View full width]

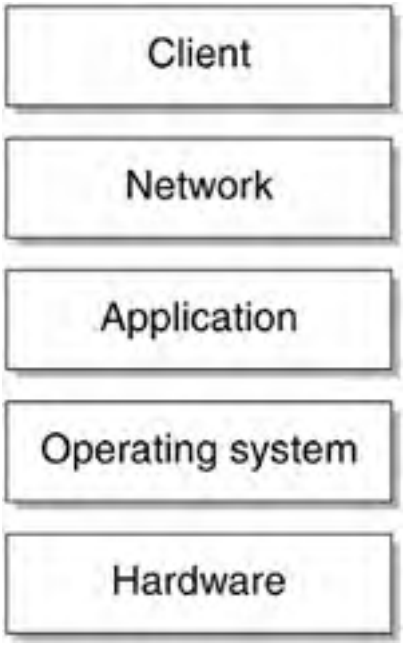
```
# ls

return_bounced.txt*    return_delayed.txt*    return_failed.txt*     return_header.opt*
return_suffix.txt*
return_deferred.txt*   return_delivered.txt*  return_forwarded.txt*  return_prefix.txt*
return_timedout.txt*
```


Chapter 10. Security

Security is integral to any mission-critical enterprise-wide system. To paraphrase a recent animated hit movie, "Security is like an onion...it has *layers*." Whether the system is a messaging system or a database system, there are many layers (FIGURE 10-1) when addressing security—each and every layer is integral to the overall security of the system. The question is how much effort is really appropriate for the level of security required.

Figure 10-1. Security Layers



This chapter discusses in detail the specific issues surrounding the security of a messaging server, including the server platform, the various protocols and their impact, and securing the contents of the messages.

This chapter divides the topic of security as it relates to a messaging system into three different layers or topics:

- Network
- System
- Messaging Software Protocols

Network

The network layer is the layer that is external to the physical host and operating system on which the Messaging Server or one of its components is running. It is surprising to see in this day and age of reasonable paranoia regarding basic network security how many customers are not actually deploying even the most basic network security measures such as firewalls. You may be saying, "of course we have a firewall!" OK, but is it just there for traffic between the Internet and your organization's network? Or do you have several layers of firewalls, including a layer protecting mission-critical systems such as your messaging system?

Why use firewalls for the Messaging Server? Even under the most complex configurations, roughly half a dozen ports must be exposed to users. Why allow users to access ports they do not have to access in the first place?

Some customers ask, "Why must I protect against my internal users?" Many computer security experts say that internal users pose a significant risk too.

Looking at the individual components of the Messaging Server such as the message store, MTA, directory, and proxies such as the MMP and MME, there are definitely components to which only internal users need access and then only on specific ports. In a typical enterprise, TABLE 10-1 shows what access might be required for messaging:

Table 10-1. Enterprise Messaging Access in a Typical Enterprise

	Internal	Internet
Directory	Y	N
Mail Store	Y	N
MTA-INBOUND	Y	Y
MTA-OUTBOUND	Y	N
MMP	Y	N
MME	Y	N

Corporations often use virtual private networks (VPNs) to allow external users to act as though they were part of the internal network, therefore limiting access from the Internet is fairly straightforward.

The same chart can look dramatically different for organizations such as universities (TABLE 10-2), many of which do not differentiate significantly between internal networks and Internet networks (although this is rapidly changing).

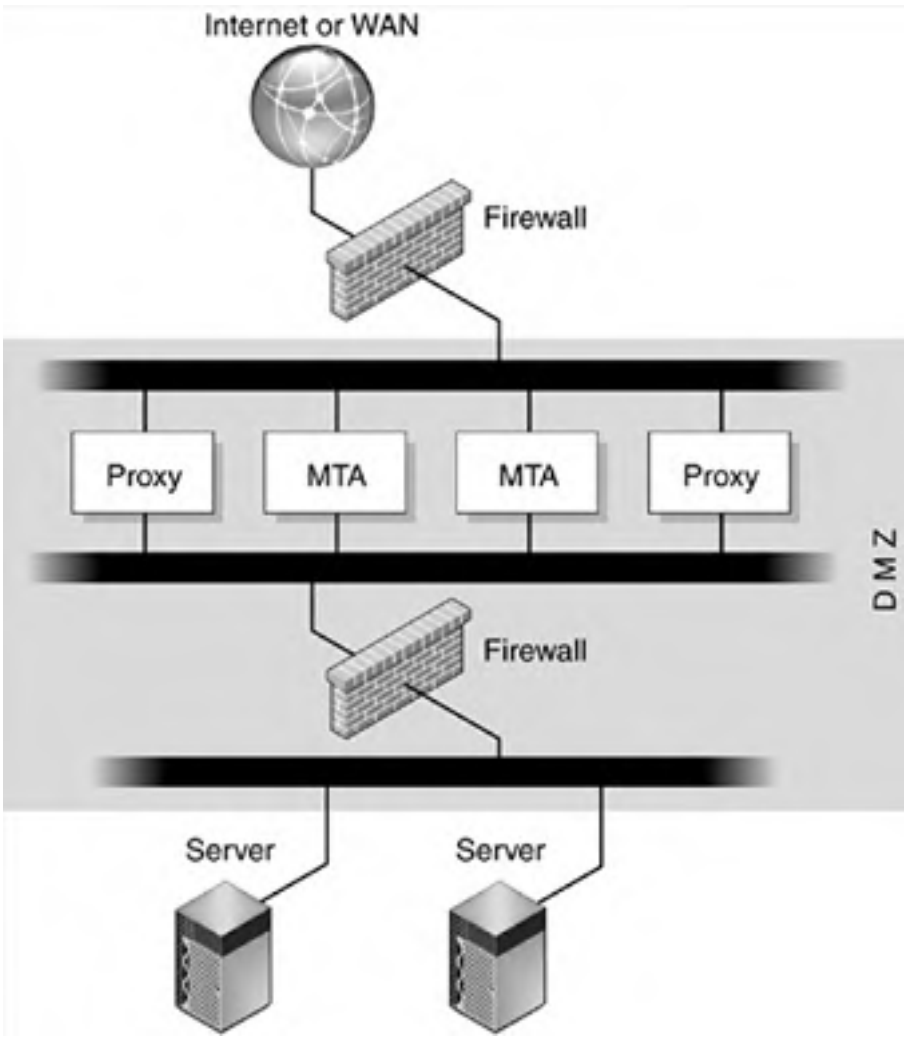
Table 10-2. Enterprise Messaging Access in a University

	Internal	Internet[*]
Directory	Y	Y
Mail Store	Y	Y
MTA-INBOUND	Y	Y
MTA-OUTBOUND	Y	Y
MMP	Y	Y
MME	Y	Y

[*] All Internet access except MTA-INBOUND—only authorized users and in a secure manner.

In some organizations, the concept known as a demilitarized zone (DMZ) (FIGURE 10-2) is used to establish systems with access to both internal and external networks—within reason and under very controlled circumstances. Many of the servers inside the DMZ are stateless and are limited to functions such as relays and proxies. Firewall rules can be explicitly configured to allow only specific internal network or Internet connections. Additional rules control how the proxies or relays are allowed to connect.

Figure 10-2. Secure Network Architecture for Messaging Environment



In reality, a significant degree of planning and forethought must be put into network security, including addressing issues such as network access to mission-critical systems such as messaging. This book does not address the issue of network security—the purpose is to make you aware of its requirements.

Some points on network security:

1. Put your server behind a firewall with packet filtering—stateful packet inspection (SPI) capabilities. Configure the firewall inspection packets to drop external packets with an internal source IP address, and forbid all connections from outside except those ports you explicitly need.
2. Do not put any Windows machines (especially Windows machines running XP, Outlook, or the Windows scripting host) on the network with your server.
3. The better the protection at the network level, the less the system security level has to deal with. Conversely, the poorer the protection at the network level, the more the system security has to deal with.

System

There are many aspects to system security. This book focuses on the Solaris OE. However, many of the concepts are easily applied to other UNIX operating environments, including derivatives such as Linux.

Basics of Solaris OE Security

Perhaps the easiest way to secure or harden the Solaris OE system is by using the Solaris™ Security Toolkit, informally known as the JumpStart Architecture and Security Scripts (JASS) toolkit. It provides a flexible and extensible mechanism to minimize, harden, and secure Solaris OE systems. The primary goal behind the development of these toolkits is to simplify and automate the process of securing Solaris OE systems.

The Solaris Security Toolkit focuses on Solaris OE security modifications to harden and minimize a system. *Hardening* is the modification of Solaris OE configurations to improve the security of the system. *Minimization* is the removal of unnecessary Solaris OE packages from the system. This removal reduces the number of components to be patched and made secure, which, in turn, has the potential to reduce entry points available to a possible intruder.

The Solaris Security Toolkit provides two methods for securing systems during initial Solaris OE installs by using JumpStart software technology or from the command line, which is called *standalone mode*. This standalone mode allows the Solaris Security Toolkit to be used on systems that require security modifications or updates. The standalone mode is particularly useful when rehardening a system after patches have been installed. The Solaris Security Toolkit can be run any number of times on a system with no ill effects. Patches can overwrite or modify files the Solaris Security Toolkit has also modified; by rerunning the Solaris Security Toolkit, any security modifications undone by the patch installation can be reimplemented. In production environments, patches should always be staged in test and development environments before installation.

The Solaris Security Toolkit is located at:

http://wwws.sun.com/software/security/jass/.

Other security related Sun BluePrints are located at:

http://www.sun.com/solutions/blueprints/browsesubject.html#security.

Note

The toolkit locks down the "nobody" account, so if you are using this account to run a prototype or demo messaging system, you must edit the /etc/passwd file and delete the /sbin/noshell at the end of the "nobody" entry. Alternatively, create a new group and users for the messaging system, as recommended in the *iPlanet Messaging Server Installation Guide*.

Additional system security measures include solid intrusion detection and monitoring. While the toolkit provides some hardening of the Solaris OE, it does not do intrusion detection.

A variety of commercial and open-source offerings are available for system intrusion detection and monitoring. Network intrusion detection and monitoring packages are also available. An example of this type of software is TripWire.

Some points on system security:

1. Start by installing the most recent Solaris OE security patch clusters and setting up a procedure to update the patches once every few months and in response to security alerts from the vendor.
2. Turn off all operating system services that listen on a port that you do not use. The toolkit does some of this.
3. Replace telnet, ftp, and so forth with sshd. The toolkit also does some of this—sshd is part of the install by default under the Solaris 9 OE.
4. Do not provide users with interactive accounts on the Messaging Server—only administrators should have accounts.
5. Do not change the default configuration of Solaris OE regarding the console; require administrators to log in and then become superuser and change the directory to root unless they are actually on a console port.
6. Implement sudo or its equivalent for administrators or the equivalent functionality (role-based access control), which is included within the Solaris Operating Environment. For more details, see the *Solaris System Administrators Guide on Security Services* at:

   http://docs.sun.com/db/doc/806-4078.

7. Read and understand the Sun BluePrints related to system security at:

   http://www.sun.com/solutions/blueprints/browsesubject.html#security.

8. Install intrusion detection and monitoring software, for example TripWire. Sun, in conjuction with Symantec, recently introduced a new intrusion detection appliance. For more information, go to:

   http://www.sun.com/smi/Press/2003-04/sunflash.20030414.1.html.

Messaging Software Protocols

As with the system security, the messaging software security also has layers of its own, which can be separated into the following components:

- Directory
- Message Store
- MTA
- Proxy

Directory

There are several aspects of securing the directory beyond securing the basic server and operating system. These aspects are:

- ACI— limiting permissions as to what people can see and do
- Search limits— how many responses and how much time can be spent searching
- SSL— enabling SSL support
- Non-standard ports— not using ports 389 or 636 for LDAP or LDAP over SSL

This is not an exhaustive list of directory security issues, but it covers most of the options to secure the Directory Server protocols and access using these protocols.

ACI

Access control instructions (ACIs) are basically permissions. The Directory Server provides a mechanism by which you define access. When the server receives a request, it uses the authentication information provided by the user in the bind operation and the ACIs defined in the server to allow or deny access to directory information. The server can allow or deny permissions such as read, write, search, and compare. The permission level granted to a user may be dependent on the authentication information provided.

Using access control, you can control access to the entire directory, a subtree of the directory, specific entries in the directory (including entries defining configuration tasks), or a specific set of entry attributes. You can set permissions for a specific user, all users belonging to a specific group or role, or all users of the directory. Finally, you can define access for a specific location such as an IP address or a DNS name.

Chapter 6 of the *iPlanet Directory Server Administration Guide* provides details on configuring and establishing ACIs.

The two or three most common changes or customizations are for customers to change permissions (ACIs) for:

- Self
- Anonymous
- General access

Some customers, for example, do not want anyone to be able to change their own entry information. For this, an ACI can be created to restrict (deny) change privileges to "self."

Other customers want anonymous (anyone) to see only the person's name, phone number, and email address—nothing else. Again, an ACI can be created for "anyone" to be able to only see the common name, phone number, and email address.

General access can control authenticated users' access to the directory, so even if they successfully log in, they can see more than "anyone" but less than "self," for example. An ACI modification or creation can do this too.

Other conditions that can be taken into consideration when creating ACIs include time of day, day of week, IP address, and DNS name.

ACIs are a powerful way to control access to the directory, if properly configured, but can also be a problem if poorly done since they can impede other software from working correctly.

Search Limits

One of the newer features of the Directory Server is the ability to limit search limits and time spent searching for different types of users. Previous versions only provided for one overall limit, not multiple limits.

These limit features provide the ability to configure both size limit (number of entries returned) and time limit (maximum amount of real time in seconds the server should spend performing a search request) as not only a system default, but also at a finer-grained level. For example, you can configure the Directory Server so that "anyone" or unauthenticated users can only retrieve five entries and spend 20 seconds searching, while "general access" users (for example, those who have authenticated successfully) can retrieve 50 entries and spend 180 seconds searching.

The Directory Server allows you to specify resource limits, including sizelimit, timelimit, lookthroughlimit, and idletimeout down to the per-user level. This is documented online at:

http://docs.sun.com/source/816-5606-10/password.htm#1085603.

Enabling SSL Support

Enabling SSL support for the Directory Server is the first step in providing secure access using LDAP over SSL for queries and responses. Enabling SSL by itself does not configure the other servers to take advantage of it, however. Additional configuration typically must be done. Enabling SSL simply turns on the Directory Server's ability to encrypt LDAP using SSL over the network, so where LDAP is normally on port 389, LDAP over SSL is typically on port 636 (either of which can be changed).

The caveat on enabling SSL for anything is certificate management. A certificate (key) must be generated and imported into the server. For specific instructions, see Chapter 11 of the *Sun ONE Directory Server Administrators Guide*. Also, the personal identification number (PIN) or password for the certificate must be entered to start the server. This PIN can be stored within a file, but it is done in cleartext, which provides some security issues and risks that must be assessed prior to doing so.

For details on managing SSL, see Chapter 11 of the *Sun ONE Directory Server Administrators Guide*.

Enabling SSL on the Directory Server will have some performance impact that must be taken into consideration when sizing. This depends specifically on the number of transactions and usage of the SSL-enabled LDAP ports. For example, if only ten percent of the transactions require SSL, use SSL only for these ten percent if possible. The Directory Server supports hardware acceleration of SSL workload, but this can add some additional configuration requirements and complexity.

Non-standard Ports

The Directory Server and Messaging Server provide the ability to use non-standard ports for LDAP and LDAP over SSL. While this prevents some basic default port scanning, it also means that all the software that access the directory must be configured to use the non-standard port numbers too, so this becomes slightly more difficult to manage and configure.

Message Store

From the Messaging Server software point of view, the security aspects on the message store are limited to the basic email protocols—POP, IMAP, SMTP, and HTTP (web mail), plus the administrative interfaces over HTTP.

SMTP

Configure the SMTP daemon on the mailstore (it is required to deliver mail to mailboxes) to only accept connections from the "official" MTAs. The MTA that on the mailstore is there just to deliver mail to mailboxes and users. The exception to this rule is for smaller configurations that want to have a consolidated or "all-in-one" messaging system where the MTA on the mailstore is the "official" MTA and the only MTA.

MTA

Several things can be done to make the MTA more secure, although this is really more of a configuration issue and depends upon the environment.

The biggest security feature for the MTA is to require authentication prior to sending an email. This is known as SMTP authentication or SMTP AUTH for short. This authentication requires that the sender have a valid login and password (account) on the messaging system, thus preventing users from sending email from anywhere, regardless of whether they are local (on net) or not. This also prevents people from sending thousands of emails out to the Internet using your MTA as a relay, though it does not prevent forged headers (see RDNS).

RDNS

Reverse DNS (RDNS) validates that the domain name from which the mail is purported to have been sent (sender's domain name) is at least registered, that is, valid. The setting within the Messaging Server that provides this capability is called mailfromdnsverify. This lookup only verifies the existence of the domain name in the DNS registry nothing else. Spammers and others can easily forge headers and the domain names can be registered/churned quickly. So, the debate is how useful RDNS really is at this point. In fact, it is only one feature of many that slows down spam and so forth.

Antivirus and Antispam

"Virus Scanning" on page 198 and "Antispam" on page 199 cover antivirus and antispam in more detail. Providing these services at the MTA level greatly enhances the overall security of the messaging environment.

Securing the Message Contents

Securing the message contents is usually the final step along the way in securing the mail system, and naturally the most difficult to implement for many reasons. PGP signing allows for the non-repudiation of a message—that is you can validate who it is from and the contents. SMIME is secure MIME, which actually encrypts the contents of the message.

While adding these options to your messaging system offers additional levels of security, it also adds significant levels of support and administration as well. Extra consideration must be given when implementing digital signing or message encryption.

Implementing PGP Signing

Pretty good protection (PGP) signing (digital signing) is simpler to do than message encryption (sometimes referred to as SMIME) by far, but it does not prevent access to the contents. It does allow you to confirm the identity (signature) of the sender and verify that the contents of the message (but not headers) have not been tampered with (non-repudiation).

The Online help for Mozilla v1.3a states:

> digital signature. A code created from both the data to be signed and the private key of the signer. This code is unique for each new piece of data. Even a single comma added to a message changes the digital signature for that message. Successful validation of your digital signature by appropriate software not only provides evidence that you approved the transaction or message, but also provides evidence that the data has not changed since you digitally signed it.

PGP is a public-private key system. That is, there are two keys, one private key that only the user knows and one public key that anyone can find out by looking it up in a directory. By using the combination of these keys, you can encrypt and sign documents meant for either public consumption or just one other individual. PGP signing operates slightly differently for each platform such as Windows, Solaris OE, or Linux and so forth, but overall it operates in a very similar manner.

For an overview of PGP, see:

http://www.pgpi.org/doc/overview/.

To implement PGP signing:

1. **Obtain PGP software.**
2. **Install PGP utility.**
3. **Generate key pair—your public and private key.**
4. **Create email.**
5. **Cut and paste email into PGP utility to generate PGP signature (checksum).**
6. **Cut and paste PGP signature to bottom of email.**
7. **Send email.**

   A good tutorial on the whole process is available at:

   http://www.haltabuse.org/pgp/index.shtml.

Some email clients support PGP signing natively, basically calling a PGP utility and performing the operation for the user. Some examples include:

- Ximian (http://www.ximian.org), KMail (http://devel-home.kde.org/~kmail/index.html),
- Postilion (http://www.postilion.org/) and
- Arrow (http://www.newplanetsoftware.com/arrow/).

These utilities may be slightly out of date. Plug-ins for Mozilla and Netscape 7, such as Enigmail, are also available.

A more complete list is located at:

http://email.about.com/cs/openpgpsoftware/.

The Messaging Server web mail interface can be customized to perform PGP signing automatically, but this takes some effort and requires that the PGP keys be stored inside the Directory Server so they can be accessible for both the web mail client and the public.

PGP or digital signing has little impact on the server itself because the client mainly performs the work. An exception is if the web mail is customized to perform the key calculation, this added workload must be considered when sizing the server. It does add some additional length to each message signed—roughly 512 characters or so. While this is not very much, it can increase the overall storage and throughput requirements if every message is signed.

SMIME

SMIME goes beyond simply computing a checksum based upon the message content and your private key. This includes encrypting the entire message so it cannot be read. Again, this requires encryption software and often a thick client, though the Messaging Server web mail client can be modified to perform SMIME (see also Implementing PGP Signing).

The main issue with SMIME versus signatures is that you *must* unencrypt the message and attachments to be useful with SMIME, whereas with signatures you may only have to validate the sender if you suspect the message has been tampered with or the source is not genuine.

SMIME has a significant impact on the sizing of the server if the web mail client is customized. If thick clients perform most of the work of encryption and decryption, the impact is typically with the overall increase in the size of the message.

SMIME or encrypted messages are the *only* methods of ensuring privacy from even system administrators—as much as possible, since no encryption is completely unbreakable given enough time and computing power.

This book refers to PGP or Open PGP, but the message contents can also be secured by using X.509 Certificates.

Conclusion

Some points on messaging server software security:

1. Require SMTP AUTH for mail submission and turn on appropriate logging, so abuse can be traced.
2. Set ACIs in the directory appropriately for your environment.
3. Enable SSL for LDAP, IMAP, POP, and web mail to provide secure transmission.
4. Configure and support PGP/digital signatures if non-repudiation and sender validation are required.
5. Configure and support SMIME or encrypted messages if absolute privacy required.
6. Keep in mind that each layer of security at this level adds administrative and support overhead.

Chapter 11. Migration

After you install the basic Messaging Server, one of the more difficult tasks is to migrate the existing user base and mailbox contents. Different techniques can be used, but only specific techniques are valid for specific migrations, Exchange for example. Additionally, other parts of the migration have specific issues, such as using the migration as an opportunity to standardize mail address formats while maintaining legacy addresses that can be addressed. This chapter describes the best practices for migration and identifies potential problems that may occur during the migration phase. The items that must be migrated are:

- Directory
- Mailbox (content)
- Mail list (aliases)
- Personal address books

This chapter covers the following topics:

- Basic Steps (Generic)
- Sendmail (UNIX Mail)
- Exchange, Novell Groupwise, and Lotus Notes

The process of installing a new messaging system can be divided into three phases:

- Installation
- Provisioning and maintenance of users
- Migration from the old system

Previous chapters covered the basic installation of the Messaging Server and the maintenance and provisioning of users. This chapter covers the final stage—getting users off the old system and onto the new system.

Few, if any, organizations will be starting a brand new deployment of messaging. Migration of an existing email system is not trivial. Migration often consumes half of the overall project effort, but you can minimize the time you spend by planning and using the knowledge this chapter provides.

Decisions regarding whether to migrate everything at once or user by user (self service) must be made. Each method has its pros and cons.

Basic Steps (Generic)

Migrating a messaging system has three steps:

- User Information— user ID, password, name, and so forth
- Messages and Folders— content
- Aliases and System-wide Mailing Lists— content and aliases

The techniques and methods used for migration of message and folder contents are different than those used for aliases or system mailing lists contents.

User Information

At first glance, the issue of migration of user information seems pretty trivial. However, much depends upon the format and source of the old mailing system. Some mailing systems use basic user stores such as text files, /etc/password for example. However, others might use an actual database. No problem, correct? To a point, yes, but the real issue lies in what information is there that you really cannot get to—specifically passwords.

Typically, gaining access to basic user information such as first name, last name, user ID, email addresses, and so forth is done easily enough. However, passwords are often stored in hashed or encrypted format.

Why Are Passwords Important?

During the migration of the actual content, system utilities may have to actually log in and act as if they were the user, unless they can read the mail directly off the file system or there is an administrative password option.

Password Handling Options

1. Temporarily reset the password to something known.

   This is easily enough done in many cases, but what else will it affect? Can you set it back to the previous password when you are done?

2. Decrypt (break) the password.

   This does not work in all cases. It is slow and not really feasible.

3. Use the administrator password (root or equivalent).

   This is not possible in all systems; may not actually act as user.

4. Set the password in cleartext.

   This is ideal if possible. It can be done through a web page if needed.

Messages and Folders

In many ways, populating the user information is the easiest part of the migration. At worst, you can simply provision all existing users in the same manner as if they were new users. This, however, leaves you with an empty inbox.

There are several ways of migrating the actual contents from the old messaging system to the Sun ONE messaging system. Ideally, the fastest method is that which can directly access the data on disk. However, due to the varied formats in which messaging software stores data, this is not always possible or recommended.

In most cases, the lowest common denominator provides the solution—POP or IMAP and perhaps SMTP. Why? Because the vast majority of messaging systems support these protocols and they are platform and operating system neutral.

> **Note**
>
> If you are using only POP3 on your mail server (that is, no folders or advanced functions), consider using the built-in feature of the web mail interface of the Sun ONE Messaging Server that provides the ability to *check other mail*. This allows users to simply configure the information such as user ID and password and

then click the button.

One of the easiest and most overlooked methods for migrating existing content is simply not to do it. Rather, let the users maintain their old accounts for some period of time and, should they desire to, simply drag and drop between the old and new accounts. Most of today's messaging clients, such as Netscape or Mozilla, have the ability to be configured for multiple messaging servers.

Letting Users Maintain Messages and Folders

Overall, the procedure is this:

1. **Install the Messaging Server.**
2. **Provision all existing users as though they were new users.**
3. **Configure the software so that all mail is delivered into new accounts.**
4. **Place instructions on configuring Netscape (or other browser) on the Web or in the old email account.**
5. **Provide instructions for moving email by using drag and drop.**
6. **Provide a deadline for moving off the old messaging system.**
7. **Decommission the old messaging system.**

This procedure avoids all password issues.

A variation on this procedure provides continued delivery to the existing messaging system until the customer wants to cut over, and eliminates the need for drag and drop. This procedure is:

1. **Install the Sun ONE Messaging Server software.**
2. **Provision all existing users as if they were new users.**
3. **Configure the MTA and directory to continue to route mail to the existing messaging system.**
4. **Create a simple web page that:**
    - Authenticates the user, capturing their password in cleartext
    - Sets the Messaging Server password to the captured password
    - Executes the MoveUser utility or something similar like fetchmail

      For MoveUser syntax details, see the *iPlanet Messaging Server Reference Manual* at:

      http://docs.sun.com/source/816-6020-10/ms_cmds.htm#15794.
    - Configures any additional settings required
5. **Allow the users to migrate at their leisure (within reason).**

Aliases and System-wide Mailing Lists

Unfortunately there is no automatic method of doing aliases and system-wide mailing lists. However, there are some significant opportunities to reduce future administrative workloads.

The three ways to migrate system wide mailing lists are:

- Aliases File
- Delegated Administrator
- Creating Dynamic Groups and Email Lists Using Direct LDAP Manipulation (Sun ONE Administrator Console)

Aliases File

As discussed in Chapter 6 of the *Sun ONE Messaging Server Administration Guide*, the aliases file is used to set aliases

that are not set in the directory. In particular, the postmaster alias is a good example. Aliases set in this file are ignored if the same aliases exist in the directory. One drawback in using the aliases file is that the MTA must be restarted for any changes to take effect.

A significant use of the aliases file is for expansion of large membership (quantity) aliases, such as by ISPs that must distribute individual messages to all 10,000,000 users quickly. An alias is created that expands into more aliases, and so forth. In a way this use is like throwing off threads during program execution, and it allows quicker processing of large mass mailings.

However, it is also a good way to easily import existing aliases (with some modification) during the initial migration. Once that is done, additional and more appropriate methods of mail list creation can be used.

Delegated Administrator

Most users and administrators create and administer aliases using the Delegated Administrator web-based user interface. Administrators can determine who, if anyone, has the ability to create mailing lists. A person with the ability to create and manage mailing lists has control over several things regarding a specific list:

- Additional owners
- Internal members
- External members
- Moderators (if any)
- Who can join the mailing list
- Who can see who is in the mailing list
- Whether a mailing list can be seen

Unfortunately, you cannot enter dynamic list criteria using the Delegated Administrator interface because these are not dynamic lists—either the user must subscribe to the list through the Delegated Administrator interface or the list administrator must add the person's email address to the list.

Creating Dynamic Groups and Email Lists Using Direct LDAP Manipulation (Sun ONE Administrator Console)

One feature that direct LDAP manipulation provides is the ability to create *dynamic groups* or email lists. This feature is based on LDAP queries that are then expanded upon at runtime. For example, a mailing list of Dave could be created specifying that anyone with "dave" or "david" as part of their common name (cn) in the directory would be part of the mailing list. Then, as users are added to the messaging system's directory, there is no need to administer this list because it is always up to date.

Unfortunately, the option to create a dynamic group-based mailing list through the Delegated Administrator interface or the aliases file is not possible. To do this, you must either access the Messaging Server through the Sun ONE Administrator Console or by direct LDAP manipulation.

The overall process is fairly simple:

1. **In the Administrator Console, access the Create Group or Edit Entry window, then click on the Mail and the Email-only Members tabs.**
2. **Click on the Add button under the Dynamic Criteria field.**

   Dynamic criteria are really just LDAP query strings, much like those you can enter against the directory from Netscape or other browsers.

   The following is an example of an LDAP search URL that filters for users who have "dave" or "david" as part of their common name:

   ldap:///o=isp??sub?(&(objectclass=person)(|(cn=*dave*)(cn=*david*)))
3. **Enter an LDAP search URL in the field or click the Construct button to open the Construct LDAP Search URL window.**

   Construct LDAP Search URL is a utility that aids in construction of the search URL.

4. **Click OK to add your entry to the "Dynamic criteria for email-only membership" field and dismiss the Add Dynamic Criterion window.**

For more detailed information, see Appendix D, "Managing Users and Mailing Lists of the Sun ONE Messaging Server," in the *iPlanet Messaging Server Administration Guide*.

Personal Address Books, Lists, and Bookmarks

The final step in the migration process tends to be the migration of each individual's own Personal Address books, lists, bookmarks, and so forth. This step is highly dependent upon the mail client people are using and to what client they are migrating. In most cases, there are at least a couple of ways to actually convert the content of one messaging client's address book and lists to the new one.

Migration Utility

Many of the newer email clients such as Eudora, Mozilla, and Netscape provide new clients the ability to read existing address books from other programs such as Outlook and Outlook Express. The email clients will often prompt you during the initial install to import any existing address books, and in some cases actually already know that they are there.

Export to Neutral Format From Old Client and Import Using New Client

In situations where an email client may not provide an import utility to directly read the address book of your old email client, many times you can simply export the old client and import the new client. It is important to look for a neutral format such as Lightweight Data Interchange Format (LDIF), comma separated variable-length file (CSV), or tab-delimited file. Check in the old email client and the new email client to see what format is available to both.

It is also important to know that in some cases the fields being exported from the old email client address book do not align directly or the same with what the address book in the new email client expects. Most import functions provide the ability to map fields upon import.

If your new email client does not do this, a good idea is to use a spreadsheet program such as the StarOffice™ software to import the CSV file, change the order of the fields, and save the file.

You can also write a script if you have a lot of users doing the migration.

Other Utilities to Convert Format Directly

Several companies and web sites have simple utilities for migration of address books from one format to another. A good web-based example is:

http://www.interguru.com/mailconv.htm.

Other companies that specialize in migration between proprietary mail systems (for example, Exchange) offer utilities as part of their services or migration utility software. For more information see "Exchange, Novell Groupwise, and Lotus Notes" on page 175.

Sendmail (UNIX Mail)

Sendmail is an MTA. It does not specifically provide methods for mail storage or retrieval (reading mail). However, it is often configured to use /var/mail type storage. Then additional programs such as the Washington University IMAP server (WashU IMAP) or Carnegie Mellon University's Cyrus POP server are added so users can retrieve their mail from /var/mail. This section deals with this specific type of generic configuration most often found when dealing with sendmail.

As stated previously, Sendmail is an MTA and as such much of the work is done at the MTA converting aliases, rules, and so forth. Converting users (see "User Information" on page 174) and /var/mail mailbox content (see"Mailbox Content" on page 175) is fairly straightforward.

Unfortunately there are no tools to migrate the MTA configuration from Sendmail to the Sun ONE Messaging Server's MTA—it is a manual process. Migration from the traditional Sendmail can be done by using the preceding generic method, but there are some advantages in doing a more direct (that is, not IMAP to IMAP) migration of content. Also, user information tends to be stored in /etc/password files, so it is easy to access.

Unfortunately, Sun ONE Messaging Server 5.2 does not come with as much assistance in migrating from Sendmail as previous versions did. Some of the appendixes in the Netscape Messaging Server 4.x documentation contain good information.

A good white paper specifically on this topic is "iPlanet Messaging Server Migration from UNIX® Sendmail" by John Twomey, dated July 2001. It is 31 pages, covers this subject in detail, and includes sample scripts. To obtain a copy of this white paper, contact your local Sun Sales Representative or System Engineer.

Some updating of the information in the white paper is required for use with Sun ONE Messaging version 5.2, though.

User Information

User information an be converted directly from /etc/password by using a simple Perl script called unix2ldif.pl. This script creates a properly formatted ldif file which can then be imported directly into the Messaging Directory. See the template in "User Information" on page 175.

Mailbox Content

Using additional scripts found in the John Twomey white paper, mailbox content can be imported via the imsimport utility. These scripts ensure proper formatting of the command as well as reiteration through the various mailboxes and folders. The white paper also includes details on Pine-formatted folders. To obtain a copy of this white paper, contact your local Sun Sales Representative or System Engineer.

Mailing Lists (aliases)

One could easily use the aliases file, as stated previously. However, the white paper provides a script to create mailing lists in the directory, which is a more appropriate and better way of doing things.

Personal Address Books

Only the previously described generic or general methods are available. The interguru.com web site link provides a good utility to migrate address lists from programs like Pine and elm used by interactive users in Sendmail environments.

Exchange, Novell Groupwise, and Lotus Notes

Given the proprietary nature of these messaging solutions both on the server side and on the messaging client side, migration away from them is somewhat difficult. Seek professional help. Several organizations have specialized migration software to assist in migrating away from Exchange, Groupwise, and Lotus Notes, including Sun Professional Services and a company called Wingra.

User Information

You must export the native format to something more malleable such as CSV or tab delimited files. Then, you can write a script to take this information and create a properly formatted LDIF file for import into the Directory Server.

A basic template to create a user in LDIF format is:

```
dn: uid=<uid>, ou=people, o=<hostname_fqdn>, o=isp
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: inetUser
objectClass: ipUser
objectClass: nsManagedPerson
objectClass: userPresenceProfile
objectClass: inetMailUser
objectClass: inetLocalMailRecipient
mail: <uid>@<hostname_fqdn>
mailUserStatus: active
dataSource: NDA 4.5 Delegated Administrator
mailHost: <hostname_fqdn>
givenName: History
cn: <first_name> <last_name>
uid: <uid>
sn: <last_name>
mailDeliveryOption: mailbox
inetUserStatus: active
userPassword: <password>
creatorsName: uid=serviceadmin,ou=people,o=<hostname_fqdn>,o=isp
modifiersName: uid=msg-admin-<hostname_fqdn>-20020710153937,ou=people
 ,o=<hostname_fqdn>,o=isp
createTimestamp: 20030414044513Z
modifyTimestamp: 20030414051012Z
nsUniqueId: d5cba701-1dd111b2-80cac302-81db34e7
nswmExtendedUserPrefs: meDraftFolder=Drafts
nswmExtendedUserPrefs: meSentFolder=Sent
nswmExtendedUserPrefs: meTrashFolder=Trash
nswmExtendedUserPrefs: meInitialized=true
pabURI: ldap://<hostname_fqdn>:389/ou=<uid>, ou=people, o=<hostname_fqdn>,
 o=isp,o=pab
```

Mailbox Content

Mailbox content migration is mostly limited to the generic method through POP or IMAP. See "Basic Steps (Generic)" on page 168.

Mailing Lists

See "Basic Steps (Generic)" on page 168.

Personal Address Books

See "Sendmail (UNIX Mail)" on page 174 and seek professional help.

Chapter 12. Performance Tuning

As with any system, performance is a key element to getting the most return on investment as well as maintaining happy users. This chapter contains practices and principles specifically related to performance tuning of a Sun ONE Messaging Server which can differ from or contradict conventional tuning wisdom. This chapter points out the areas on which a Sun ONE Messaging Server administrator should concentrate.

This chapter covers the following topics:

- Netscape Directory Server
- Solaris OE
- MMP
- MTA Tuning
- Notices
- Postmaster Mail

Netscape Directory Server

- **If you are using Netscape Directory Server 4.1x software, set the following (as root) on each of the servers running the LDAP server:**

  /usr/sbin/ndd -set /dev/tcp tcp_naglim_def 1

  The Solaris OE introduces a 100 ms delay in TCP/IP. This parameter tells the Solaris OE that any write that is smaller than *N* will be delayed. In Sun ONE Directory Server 5.0 software it is configurable, using the TCP_NODELAY flag, which is set by default.

Solaris OE

This section covers the following topics:

- Setting TCP/IP Parameters
- Setting tcp_local_option and tcp_ internet_option File Parameters
- Setting /etc/system Parameters
- Setting configutil Parameters

Setting TCP/IP Parameters

- **Apply the following TCP/IP tuning settings to all mail servers.**

  These settings may also be appropriate for LDAP servers. The values of these settings are for high-speed networks with lots of traffic to and from the servers.

```
# ** Performance related **
/usr/sbin/ndd -set /dev/tcp tcp_recv_hiwat 65536
/usr/sbin/ndd -set /dev/tcp tcp_xmit_hiwat 65536
/usr/sbin/ndd -set /dev/tcp tcp_conn_req_max_q 4096
/usr/sbin/ndd -set /dev/tcp tcp_conn_req_max_q0 8192
/usr/sbin/ndd -set /dev/tcp tcp_smallest_anon_port 8192
/usr/sbin/ndd -set /dev/ip ip_ignore_redirect 1
/usr/sbin/ndd -set /dev/tcp tcp_keepalive_interval 30000
/usr/sbin/ndd -set /dev/tcp tcp_naglim_def 1
# ** Security related **
/usr/sbin/ndd -set /dev/tcp tcp_mss_min 108
/usr/sbin/ndd -set /dev/ip ip_respond_to_echo_broadcast 0
/usr/sbin/ndd -set /dev/ip ip_forward_directed_broadcasts 0
/usr/sbin/ndd -set /dev/ip ip_strict_dst_multihoming 1
/usr/sbin/ndd -set /dev/ip ip_forwarding 0
/usr/sbin/ndd -set /dev/ip ip_forward_src_routed 0
/usr/sbin/ndd -set /dev/ip ip_respond_to_timestamp 0
/usr/sbin/ndd -set /dev/ip ip_respond_to_timestamp_broadcast 0
#
# Solaris guide says not to set lower than 60 seconds
# should investigate further, but the following has worked
/usr/sbin/ndd -set /dev/tcp tcp_time_wait_interval 15000
#
# Set according to local specifics.
#/usr/sbin/ndd -set /dev/tcp tcp_mss_def 1460
```

Setting tcp_local_option and tcp_ internet_option File Parameters

To prevent DOS attacks and protect overall system health, you should enable the following parameters in the msg *instance*/imta/config/tcp_local_option and msg *instance*/imta/config/tcp_intranet_option files. (They do not exist by default.)

```
!
!
DISABLE_ADDRESS=1
DISABLE_CIRCUIT=1
DISABLE_EXPAND=1
DISABLE_GENERAL=1
DISABLE_STATUS=1
HIDE_VERIFY=1
ALLOW_RECIPIENTS_PER_TRANSACTION=
ALLOW_REJECTIONS_BEFORE_DEFERRAL=
```

The last two parameters should be set according to site policy. They are listed here so you know they exist. Check the reference guide for more options.

Setting /etc/system Parameters

To set the /etc/system parameters:

1. **Set** tcp_conn_hash_size **to**:

   ```
   set tcp_conn_hash_size=262144
   ```

2. **Set the file descriptors to:**

   ```
   # set hard limit on file descriptors
   set rlim_fd_max=4096
   # set soft limit on file descriptors
   set rlim_fd_cur=4096
   ```

3. **Set** maxusers **to**:

   ```
   set maxusers=2048
   ```

   Ideally you do not have to set ncsize. Setting maxusers to the maximum value (2048) should allow the system to automatically tune itself. You can use the command vmstat -s | grep cache to see the percentage of hits against the directory name lookup cache (DNLC). You want this to be as high as possible.

   If, after setting maxusers, the cache hit rate against DNLC is not high enough, set ncsize using the following guidelines:

   ncsize = *(4 * (*max_nprocs + maxusers*))* +320

   max_nprocs = *10 +(16 * *maxusers*)*

   maxusers = physmem – *2*

   - If your system is using a VERITAS file system, you may have to adjust two important variables:

     vxfs:vxfs_ninode— VxFS inode structures held in memory

     vxfs:vx_bc_bughwm— the high water mark of the buffer cache's buffer

   - If your system uses VERITAS Volume Manager (VxVM), you should look at vxio:vol_maxio. This variable controls the maximum size of I/O requests that are sent down the SCSI chain without breaking the request up. This tunable parameter should not exceed 20 percent of kernel memory or physical memory (whichever is smaller), which should match the size of your widest stripe.

4. **Increase maximum physical I/O size. The following value should work for nearly all controllers:**

   ```
   set maxphys=8388608
   ```

Setting configutil Parameters

To set the configutil parameters:

1. **Set the number of processes (**service.[POP|IMAP|HTTP.numprocesses).

   The default is 1. You want to set this high enough to support your user load but never higher than the total number of CPUs in the system.

2. **Set the store database cache size (**store.dbcachesize) **equal to the sum of the *.db files in the** msg-*instance*/store/mboxlist **directory.**

   This is not a parameter that you set once and forget. You should adjust this parameter as your user base changes.

   ### Note

   This parameter has an upper limit of two gigabytes.

3. **Set the store database temporary directory (**store.dbtmpdir**) equal to** /tmp/msg-*instance*.

This parameter and store.dbcachesize are related. Make sure the /tmp/ partition has enough free space to hold the database cache, that is, the value of the store.dbcachesize set in step 2.

Note

Do not use just /tmp as the temporary directory. Be specific (for example /tmp/msg-*INSTANCE*/) as the file names placed in this location are the same as those used with the MTA parameters IMTA_SCRATCH and IMTA_TMP. Using a subdirectory underneath (for example, /tmp/msg-*INSTANCE*/) avoids this collision.

4. **Set authentication cache size (**service.authcachesize**) equal to a number larger than the maximum concurrency of your user base.**

   Setting this number higher than what your hardware can support opens up a denial of service (DOS) attack.

5. **Set authentication cache TTL (**service.authcachettl**) equal to the number of seconds you want entries kept in cache.**

   You must weigh the problem of user password changes being seen against the performance hit of the mail system queries against LDAP.

6. **Set user and group bind DN (**local.ugldapbinddn **equal to** cn=Directory Manager**).**

   This setting will improve the response time of queries.

   ### Note

   This logic has changed in Sun ONE Directory Server 5.0 software, so that binding as a normal user should have similar performance as binding as cn=Directory Manager in Netscape Directory Server 4.1x software.

7. **Set LDAP hosts (**local.ugldaphost**) equal to at least two dedicated LDAP consumers.**

   If the first host is recognized as down, new connections will be created. The new connections will be made to the first good host in the list.

8. **Set** local.ldapconnecttimeout **to a value in seconds. This will enable a different connect function in the LDAP library. Choose the timeout value carefully.**

   Should an LDAP connection fail, the default LDAP timeout is three minutes. This can create a large overhead in failovers.

The web mail spool directory is the directory where web mail places out going messages from clients. If you have lots of web mail users you may want to consider setting this variable to a fast file system.

MMP

In addition to applying the TCP/IP parameters listed previously in the AService.cfg file you should adjust the parameter default:NumThreads. That section of the configuration file is:

[View full width]

```
#
# number of worker threads allocated for the AService daemon.
# Optimally, it should be equal to the number of processors on the # machine, unless an
  AService DLL does synchronous handling of
# connections (Imap and Pop Proxy do not).
#
default:NumTeads 2
```


MTA Tuning

This section covers the following topics:

- Dispatcher
- Job_Controller
- Option.dat
- IMTA_TAILOR File

Dispatcher

You want to allow enough connections to support your load, while not allowing so many concurrent connections that your system is not able to respond quickly and prevent a DOS attack. The maximum concurrency number is equal to MAX_PROCS * MAX_CONNS, defaults are 10 and 20, respectively. Once you know the maximum concurrency rate for your configuration, you can determine the number of processes required to support your total load.

Job_Controller

You must be concerned with two files, imta.cnf and job_controll.cnf. The job_controll.cnf file defines the pools and the maximum number of jobs that can be run at a given time in those pools. The imta.cnf file defines what pool a channel uses and the maximum number of processes the channel can run in that pool at a given time.

In large deployments, machines are dedicated to specific roles. On a MTA-IN machine you can configure more jobs to the tcp_intranet channel, and thus deliver mail faster to your mailstores.

Large sites may want to adjust MAX_MESSAGES. Set in the global section of the job_controll.cnf file, this variable is not present by default. The default value is 100,000.

ims_master channel

If the mail store is handling lots of mailboxes and messages per second, you may want to increase the number of ims_master processes that the job_controller will start to process the ims-ms queue. To increase the number of processes you must make two changes, one in the imta.cnf file and the other in the job_controller.cnf file. In the imta.cnf file you must change the maxjobs keyword, in the job_controller.cnf file you must adjust the job_limit for the IMS_POOL.

Message Dequeue

Four parameters are related and interact together to tell the job_controller the number of processes that are responsible for delivery of messages.

job_limit— This is the maximum number of processes that can run in a given pool simultaneously. There is no method to view the number of processes in pool A versus pool B. To view all SMTP client processes use the following:

CODE box = ps -aef | grep tcp_smtp_client

maxjobs— This is a channel keyword. You can apply this parameter to each channel to set the maximum number of processes (tcp_smtp_client) that job_controller will start to process messages in this channel.

MAX_CLIENT_THREADS— The default value is 10. This option is set in tcp_*channel-name*_option though these files are not present by default. This option controls the number of threads per process.

threaddepth— The maximum number of messages per thread. To view the number of threads that a tcp_smtp_client process is currently using, you can use the command top. The default value is 128. This only applies to multithreaded channels. Channels like reprocess and conversion are single threaded. You should make adjustments to these parameters slowly and gradually until you find the right combination for your environment.

ims-ms Channel-Specific Information

The ims-ms channel is a multithreaded channel, but does not respect the threaddepth channel keyword. Instead this channel uses a hard-coded value of five (maximum of five messages handled per thread). The ims-ms channel does have a channel option to control how many threads will be used within the process DELIVER_THREADS. This option would be placed in msg-*instance*/imta/config/ims-ms_option. The default value for this channel option is 15. Like other channels the channel option file is not present by default.

If you need to decrease the amount of time it takes for delivery of messages from your MTA, you should adjust the preceding parameters.

Option.dat

The max_internal_blocks setting controls how much memory the SMTP server uses to store a message before it creates files in IMTA_SCRATCH. If you have enough memory in your server, you might consider setting this variable to a value that allows the SMTP server to store your average message size or more in memory.

MAX_INTERNAL_BLOCKS

This setting controls how much memory the SMTP server uses to store a message before it creates files in IMTA_SCRATCH. If you have enough memory in your server, you might consider setting this variable to a value that allows the SMTP server to store your average message size or more in memory.

Reverse Database

If the MTA does not need to rewrite backward pointing addresses then you can set USE_REVERSE_DATABASE=0 in the option.dat file. Use this parameter when trying to get every last millisecond of performance out of an MTA while working with a potential customer. Most ISPs will not require the MTAs to rewrite backward pointing addresses.

IMTA_TAILOR File

You can set the IMTA_TCP_FLAG_RETENTION option in your imta_tailor file to 1 so that the old *.data-failed files get purged after a day. You can find these files in msginstance/imta/queue/*channel-name*/spool/.

Note

This directory will only exist if the MTA needed to write files to it. Many sites will never see this directory on their systems.

If you are using direct LDAP lookups instead of dirsync, you can set the IMTA_TMP variable to a value that maps to a memory mapped file system, like /tmp/. Also, you can set the IMTA_SCRATCH variable to a value that maps to a memory mapped file system, like/tmp/.

Notices

The default values for notices in the Message Server software are 1 2 4 7. In large deployments, you may want to reduce these defaults.

Postmaster Mail

The defaults for delivery status notifications (DSNs) for postmaster are copywarnpost and copysendpost. In large deployments or environments where the postmaster does not want to get DSNs about users' mail, these keywords should be changed. For a complete explanation of the values, see the Sun ONE Messaging Server documentation. The possible keywords are *XYZ*warnpost and *XYZ*sendpost, where *XYZ* is either copy, warn, or err.

Chapter 13. Advanced MTA Configuration

One of the most powerful components of the Messaging Server is its MTA or message transfer agent. As described in the first part of this book, the message transfer agent is basically a router for email. If you are familiar with the PMDF product by Innosoft, the Messaging Server's MTA is basically the same product. Sun purchased Innosoft and its PMDF product in early 2000 and incorporated this technology into version 5.x of the Messaging Server, providing a highly scalable, reliable, and feature-rich MTA. This chapter does not go into details regarding the applications programming interfaces (APIs) available to programmers to access the lowest and most detailed portions of the messaging system. However, Sun preserved the applications programming interfaces from both PDMF and SIMS, so they are both available, should your installation require advance customization that goes to that level.

The MTA is so feature rich that an entire week-long course could be taught or an entire BluePrint article written on configuration and integration alone.

Conversion Channel

The conversion channel feature of the MTA provides a method of processing a message and its attachments. By default, nothing is configured in the conversion channel and nothing is configured to route through the conversion channel.

When you work with the conversion channel, it is important to have a good understanding of Multipurpose Internet Mail Extensions (MIME) messages and their structure. The setup and implementation of conversion routines require that you access the MIME-type information of message parts as they are presented to your conversion routines, so a general description of a MIME message is required to understand how to do this. Consider the following analogy.

A train is made up of one or more engines coupled to a series of railcars. The railcars on the train are of different shapes and sizes and hold different kinds of cargo. The whole assembly is thought of as a single train.

You can describe a MIME message in a similar way. A MIME message consists of a header and one or more message parts. Each part can be of a different MIME type, and the parts are joined together to form a single message.

Each part of a MIME message has content-type information which identifies the nature of that message part, perhaps the name that was given to that message part when it was created, and perhaps information on how to dispose of that message part (inline or as an attachment).

The content-type header identifies the major type and subtype of the message part. The major type describes a family of related MIME types, such as IMAGE, APPLICATION, or AUDIO. The subtype describes the specific member of that family such as JPEG, WORDPERFECT5.1 or WAV. Common content-types are IMAGE/JPEG, APPLICATION/WORDPERFECT5.1, and AUDIO/WAV.

A list of common MIME types is located at:

http://www.isi.edu/in-notes/iana/assignments/media-types/media-types, http://hostutopia.com/support/s058.html, or http://www.bc.edu/bc_org/tvp/email/helpers.shtml.

When the message enters the conversion channel, it is like a train that has all of its railcars and engines decoupled from each other. Each decoupled component is held in a temporary holding area and its MIME information is catalogued.

The function of the conversion channel is to reassemble a message from its components and, during that reassembly, apply site-supplied criteria to decide whether or not that component should be altered before it is recoupled.

The level of examination of each message part is determined by the setup of the IMTA_CONVERSION_FILE, where the system administrator can set criteria as to which message parts should be examined. Such criteria can include, but are not limited to, the content-type of the message part.

Note

Any of the features and functions of the messaging server rely upon proper formatting and playing by the rules, so to speak. If, for example, you have configured the conversion to virus scan only executable documents such as application/zip or application/* files, nothing prevents the sender from misappropriating or otherwise disguising the attachment from an application/* to a video/mpeg file. So applications, mail clients, or other messaging systems that do not correctly format MIME messages can cause issues.

The following paragraphs contain some easy examples of what you can accomplish with the conversion channel.

Adding a Disclaimer

One thing that customers often want to do is append a disclaimer to every message sent from the messaging system. This could be for legal reasons (for example, this email message is not a legal document...) or advertisement (for example, email delivered by TED—the electronic delivery guy!).

As previously described, the conversion channel can be used to perform arbitrary processing on messages down to the attachment level. In this case, the processing appends a disclaimer to text messages passing through the conversion channel.

The TCP channels handle mail coming in from or going out to external mail servers. Thus, you must modify the MTA configuration files so that all the mail routed through the main TCP channels passes through the conversion channel. Then, you must configure the conversion channel to select only the first part of a multipart message and only append the disclaimer if the message part is text.

To accomplish this, you must edit two of the messaging configuration files:

- the mappings file, which is located at /*INSTALL_DIR*/msg-*INSTANCE*/imta/config/mappings; where *INSTALL_DIR* is the directory where you installed messaging and *INSTANCE* is the name of the specific messaging server, most likely the host name. The mapping table in the mappings file, which is also known as the IMTA_MAPPING_FILE, tells the MTA which messages should detour through the conversion channel.

- the CONVERSIONS file, which is located in /*INSTALL_DIR*/msg-*INSTANCE*/imta/config/ directory and is called conversions.

In a new install, the mappings file will not exist. The conversions file, which is also known as the IMTA_CONVERSION_FILE, contains instructions as to what commands are executed when messages pass through the conversion channel.

For more information regarding channels, see:

http://docs.sun.com/source/816-6009-10/mtacncpt.htm#22760 and http://docs.sun.com/source/816-6009-10/channel.htm#43150.

The specific steps to make the conversion channel append a disclaimer are:

1. **Create a scripts directory, for example, /***install_dir***/**scripts.

   Make sure that it is owned by the user that the Messaging Server runs as. For the exercise, "nobody" was used as the user for the server:

   ```
   # mkdir -p /install_dir/scripts
   # chown nobody:nobody /install_dir/scripts
   ```

2. **Create a shell script that appends the disclaimer.**

   Create a script called append_disclaimer.sh and make sure that the script is executable (for example, chmod +x append_disclaimer.sh).

   The append_disclaimer.sh might look like:

   ```
   # !/bin/sh
   #
   # File: append_disclaimer.sh
   #
   # Usage:
   #
   # append_disclaimer.sh [-debug] "name-of-disclaimer-text-file"
   #
   # References:
   #
   # http://docs.sun.com/source/816-6009-10/channel2.htm#42323
   # http://docs.sun.com/source/816-6009-10/channel2.htm#42402
    if [ "$1" = "-debug" ]
    then
          shift
          set -x
    fi

    DISCLAIMER_FILE=$1
    DISCLAIMER_FILE=/install_dir/scripts/${DISCLAIMER_FILE}

    TAG="Standard Disclaimer Appended 'date'"

    cp $INPUT_FILE $OUTPUT_FILE # copy original message part to output
    destination.

   # See if the message was already tagged.
    grep "Comments: Standard Disclaimer Appended" $MESSAGE_HEADERS
    >/dev/null
    if [ $? -ne 0 ]
    then
   # add a blank line
    echo "" >> $OUTPUT_FILE

   # append the disclaimer
    cat $DISCLAIMER_FILE >> $OUTPUT_FILE

   # Set a directive so the message will be tagged
    echo "OUTPUT_DIAGNOSTIC=\"${TAG}\"" > $OUTPUT_OPTIONS
    fi
   ```

```
# end script.
```

3. **Put a text file containing the disclaimer in the** /*install_dir*/scripts **directory.**

   For this exercise, call it footer.txt.

   [View full width]

```
The opinions expressed above are those of the individual and not necessarily Sun
 Microsystems, Inc.
This email is not a legal document.
```

4. **Modify or create the** mappings **file to trigger a trip through the conversion channel.**

   Unlike the conversions file, the mappings file is there from the initial install. The IMTA_MAPPING_FILE is *install_dir*/imta/config/mappings on UNIX systems.

5. **Create a backup of the original file prior to making any changes:**

```
# cd /INSTALL_DIR/msg-INSTANCE/imta/config
# cp mappings mappings.bak
```

6. **Add a section to this file that says to run both *in* and *out* messages through the conversion channel:**

```
!
CONVERSION

 IN-CHAN=tcp_*;OUT-CHAN=tcp_*;CONVERT         Yes

!
! Make all messages going from any tcp channel going to any tcp channel take a
! detour through the conversion channel.
!
```

> **Note**
>
> Mapping and other MTA configuration files are very picky regarding formatting including line spacing and indentation. Consult the documentation for details.

7. **Modify or create the** conversions **file to include entries that call the** append_disclaimer.sh **script.**

   This file does not exist upon initial installation. It only exists if the conversion channel has been configured already. The IMTA_CONVERSION_FILE is *install_dir*/imta/config/conversions on UNIX. In the following example you can see where you can further identify where the message came from or is to be routed to (for example, in-channel or out-channel), as well as the type (for example, text) and subtype (for example, every subtype), and the message part (for example, 1 or 1.1).

   Why do you need two entries? Well, you must append the disclaimer to either a single-part message (for example, no attachments) *or* the first part (for example, main body) of a multipart message (message with attachments). If you had put only the first entry, MTA would not append the disclaimer to anything that had attachments.

```
!
  in-channel=tcp_*; out-channel=tcp_*;
    in-type=text; in-subtype=*; part-number=1;
    parameter-symbol-0=APPARENT_NAME; parameter-copy-0=*;
    dparameter-symbol-0=APPARENT_FILENAME; dparameter-copy-0=*;
    message-header-file=2; original-header-file=1;
    override-header-file=1;
    command="/install_dir/scripts/append_disclaimer.sh footer.txt"

!
! Append disclaimer only to the first part of a multipart message
! if that part is a text message part. (part-number=1.1 is the
! first part of a multipart message).
!
   in-channel=tcp_*; out-channel=tcp_*;
     in-type=text; in-subtype=*; part-number=1.1;
     parameter-symbol-0=APPARENT_NAME; parameter-copy-0=*;
! Append disclaimer to single part messages if the body part is text.

!
```

```
    in-channel=tcp_*; out-channel=tcp_*;
     in-type=text; in-subtype=*; part-number=1;
     parameter-symbol-0=APPARENT_NAME; parameter-copy-0=*;
     dparameter-symbol-0=APPARENT_FILENAME; dparameter-copy-0=*;
     message-header-file=2; original-header-file=1;
     override-header-file=1;
!
! Append disclaimer only to the first part of a multipart message
command="/install_dir/scripts/append_disclaimer.sh footer.txt"
! if that part is a text message part. (part-number=1.1 is the
! first part of a multipart message).!
!
  in-channel=tcp_*; out-channel=tcp_*;
   in-type=text; in-subtype=*; part-number=1.1;
   parameter-symbol-0=APPARENT_NAME; parameter-copy-0=*;
   dparameter-symbol-0=APPARENT_FILENAME; dparameter-copy-0=*;
   message-header-file=2; original-header-file=1;
   override-header-file=1;
   command="/install_dir/scripts/append_disclaimer.sh footer.txt"

! if that part is a text message part. (part-number=1.1 is the
! first part of a multipart message).
!
  in-channel=tcp_*; out-channel=tcp_*;
   in-type=text; in-subtype=*; part-number=1.1;
   parameter-symbol-0=APPARENT_NAME; parameter-copy-0=*;
   dparameter-symbol-0=APPARENT_FILENAME; dparameter-copy-0=*;
   message-header-file=2; original-header-file=1;
   override-header-file=1;
   command="/install_dir/scripts/append_disclaimer.sh footer.txt"

!
```

8. **After you have made the changes to the** MAPPINGS **and** CONVERSIONS **files, you must rebuild the configuration files and restart the dispatcher.**

```
# cd /INSTALL_DIR/msg-INSTANCE
# ./imsimta cnbuild
# ./imsimta restart dispatcher
```

Note

On large systems with lots of messages in queue restarting the job_controller unnecessarily causes a load on the system. Avoid restarting the job_controller if possible.

Converting PostScript to Acrobat

Add a PostScript (PS)-to-Acrobat conversion utility that takes all incoming messages (those actually delivered to the message store) with PS attachments and converts the PS attachments to PDF (Acrobat) format, replacing the original PS attachment.

There are a couple of things to note:

1. You have already configured the conversion channel to process incoming and outgoing messages and created a mappings file, but this file is not quite right for what you will do here.
2. Since you are calling a ready-made utility, you do not have to create a script.

To create the PS-to-Acrobat conversion utility:

1. **Modify the** mappings **file first. Here is the section you modified from before**.

[View full width]

```
!
CONVERSION
!
 IN-CHAN=tcp_*;OUT-CHAN=tcp_*;CONVERT              Yes
!
! Make all messages going from any tcp channel going to any tcp channel take a detour
 through the conversion channel.
!
```

2. **Add a section to route anything stored to the local mailstore:**

```
!
 IN-CHAN=tcp_*;OUT-CHAN=ims-ms;CONVERT             Yes
!
! make all messages being stored to the mailstore go through the conversion channel.
!
```

3. **Add an entry in the** conversions **file which is** *INSTALL_DIR*/msg-*INSTANCE*/imta/config/conversions.

   The entry might look something like:

```
!
! convert postscript to pdf
!
 out-chan=ims-ms; in-type=application; in-subtype=postscript;
 out-type=application; out-subtype=pdf; out-mode=block;
 command="/opt/sfw/bin/ps2pdf 'INPUT_FILE' 'OUTPUT_FILE'"
!
```

   This entry assumes that your system has the ps2pdf utility loaded from the Sun freeware CD. Be careful with the quoting here: INPUT_FILE is encased in single straight quotes('), as is OUTPUT_FILE, while the entire command is enclosed in double straight quotes (").

4. **Rebuild the configuration files and restart the dispatcher.**

```
# cd /INSTALL_DIR/msg-INSTANCE
# ./imsimta cnbuild
# ./imsimta restart dispatcher
```

Note

Restarting the job_controller unnecessarily causes a load on the system. Avoid restarting the job_controller if possible.

Virus Scanning

The Sun ONE Messaging Server has a facility for allowing sites to hook in third party software to perform arbitrary body-part processing. Examples could include software that performs document conversion from text to Postscript, content filtering, or other desired processing. This facility can also be used in conjunction with a third party virus scanning software to conduct email virus screening. The only typical requirement is that the virus scanning engine provide a command line interface so the Messaging Server can pass content and receive result codes back.

For additional details see:

http://docs.sun.com/source/816-6092-10/index.html.

Antispam

The Sun ONE Messaging Server provides the ability to integrate with third party software to perform special processing of messages. This includes uses such as virus scanning as well as antispam scanning. The Messaging Server has been tested in a lab environment with both SpamAssissin, an open source software for antispam processing, and Brightmail, which is a commercial antispam scanning offering. It is anticipated that many of the antivirus vendors such as Symantec, Interscan, and so on, will begin offering antispam capabilities or add ons in the near future. For additional details see:

http://docs.sun.com/source/816-6829-10/index.html.

Other Possibilities

There are almost limitless possibilities for what the MTA can be configured to accomplish. Some additional functions that customers and Sun Professional Services have added include integration with fax gateways, which require document conversion to TIFF format, as well as the addition of a custom channel (for example, FAX).

Other functions include outbound paging and support for Blackberry RIM devices.

As you would expect, using the conversion channel and performing advanced MTA configuration can open up a significant number of possibilities with the Messaging Server. The caveat here is that anything you add to the process requires additional CPU and memory resources, and potentially additional storage. As with any program or scripting, interpreted languages such as shell scripts or Perl will not be as efficient as low-level programming languages such as C, but they offer the ability to easily change and modify things without recompiling the source. Count on spending a good portion of time testing and debugging anything you do, as a small mistake such as the wrong quote mark or incorrect channel name (for example, ims_ms versus ims-ms) can create a non-functioning conversion channel routine.

Chapter 14. Highly Available Messaging Deployment

Not all organizations see messaging as a mission-critical service or for some reason decide not to implement highly available messaging. This chapter reinforces why messaging is mission-critical and needs high availability. It addresses specific issues (pros and cons) with various high-availability architectures that customers have implemented as well as some of the caveats when planning and installing messaging in a high availability environment. These lessons have been learned the hard way at various customer's sites and are found nowhere else in the documentation or technical notes.

Every year we seem to rely more and more upon our email systems. Many of the reasons were outlined in the beginning of this book. Typically the only time it becomes clear that email is mission critical is when there is a major outage or problem. If you do not think email is mission critical, try either pulling the plug on the messaging system or not adding new accounts when requested. Today we are using email systems to store more than email. They are becoming the storage folder for fax and voice mail traffic too. Around the corner are additional usages for messaging systems that we have not yet begun to imagine. So not planning for high availability (HA) seems to be asking for trouble, if not today then certainly in the future.

High Availability Architecting Differences

One of the biggest misconceptions regarding failover software, also sometimes referred to as clustering or high availability, is that it guarantees availability. Failover software cannot eliminate all outages or problems, but it can provide additional availability when it comes to hardware failures.

There are two main aspects to architecting a high availability solution:

MTBF—Mean Time Between Failures— How much time elapses on average between each failure.

MTTR—Mean Time To Repair (or Recover)— How quickly the system comes back up and is available for users after a failure occurs.

Unfortunately, many people put too much emphasis on the MTBF figure at the expense of the MTTR figure. Take for example two scenarios where the MTBF is three months, meaning the system is likely to experience a failure four times each year. In the first scenario the MTTR is one hour, while in the second scenario the MTTR is eight hours. In scenario one, the total downtime for the year is four hours while the second scenario results in 32 hours of downtime. Reducing the MTBF to 12 months still results in eight hours of downtime per year, which is more than in the first scenario.The key to availability is addressing both MTBF and MTTR, and sometimes focusing more in reality on MTTR than MTBF.

What does failover software guard against? The failover software addresses server hardware-related issues such as failed network adapters, CPUs, and so on. Some failover software goes further and protects against hung or non-responsive software processes (for example, the LDAP daemon) by querying the process every now and then. Should it be non-responsive, the failover software then restarts the daemon. If this fails a set number of times, a complete failover is then triggered. Both VERITAS and Sun Cluster software perform the process restart attempt.

What does failover software not protect against? Failover software does not protect against everything, and specifically does not address:

- Operator error (for example, rm -rf *)
- Software problems (for example, bugs)
- Storage failures (for example, drive failures or controller failures)

You can put a cluster in place and actually have more downtime due to operator error if you do not adequately provide for system administrator training on the clustering software. You can have downtime due to defective software. You can have a cluster that will not failover because the storage system, which is a shared resource, fails catastrophically or because it was not protected (for example, not on a UPS like the server was—yes, this has happened to customers). Failures can still occur. Even after addressing issues such as operator errors through training and formal procedures, software problems by an internal testing process and storage failures by having protected the storage (for example, RAID 5e and UPS, and so on).

What then? It is really a matter of *planning to fail*, that is, how you will handle a failure, even with a clustered environment. As the old commercials for American Express Traveler's Checks said, "What will you do, what will you do?" By closely examining the restoration process for your messaging environment, you can develop specific steps that will result in the fastest restoration of service time. They may include everything from the basics of re-indexing the mail contents to a complete restore, including the Solaris OE.

Questions that must be asked in your environment are:

- What is the procedure for doing this?
- How can it be improved?

Each environment is slightly different, but there are some basic techniques such as JumpStart and Flash Archive usage for rapid restoration of the operating system and software, as well as period snapshots of the database and directory, to complete mail content backups. For more details, refer to Chapter 15, "Managing Messaging Services and Preventive Maintenance," on page 209."

High Availability Architectures

The *iPlanet Messaging Server Installation Guide for UNIX* outlines several HA architectures and discusses a few of the pros and cons of each. The installation guide can be found at:

http://docs.sun.com/source/816-6014-10/.

This manual lists the following HA architectures:

- Symmetric (hot standby)
- Symmetric (Active-Active)
- N + 1 (N servers + one standby)

However, there are other HA architectures to be considered once you understand all of the parts of the architecture and how (or whether) HA affects them. As discussed earlier in the book, sometimes there are advantages to keeping things simple.

The Parts

- Directory— can be protected by using failover or multiple master replication
- Mail Store— stateful and requires failover
- MEM— stateless, requires multiple physical servers, no failover agent available
- MMP— stateless, requires multiple physical servers, no failover agent available
- MTA— stateless, can be made available by either failover or multiple physical servers. MTA is considered stateless because, due to the nature of store-and-forward, there is nothing stateful in memory, it is all written to disk—so appropriate storage protection (for example, RAID 5e or RAID 0+1) is a good idea.

Directory

With the advent of Multiple Master replication technology in the Directory Server 5.1 and higher, customers have the option of making their directory server highly available. They can use the tried-and-true method of using Sun Cluster or VERITAS Cluster software. Or, they can use Multiple Master replication that is now built into the Sun ONE Directory Server.

Mailstore

The mailstore provides the basic storage of messages as well as the native HTTP, IMAP, and POP services. Due to the stateful storage of the header information in a database, it becomes necessary to use failover software such as Sun Cluster or VERITAS to obtain high availability.

MEM and MMP

The MEM and MMP function as proxy servers. So long as the configurations and files are the same on all systems, you can have as multiple servers performing the same function. This does require a network- based load balancer such as Resonate, Cisco Load Director or F5, or Alteon to work.

MTA

Since messaging by nature is a store-and-forward architecture, it allows for some flexibility regarding availability. That is, should an MTA be unavailable, other parties will hold their messages for some period of time, periodically retrying. Typically most environments can easily configure multiple MTAs and appropriate DNS entries to provide for redundancy at the MTA level. The failover time, however, is not instantaneous, so many organizations also provide a virtual IP and network-level failover as you would for MEM or MMP. So while the MTA has some information, it can generally start up and continue where it left off without many issues—forwarding the mail it has in the queue, albeit somewhat delayed.

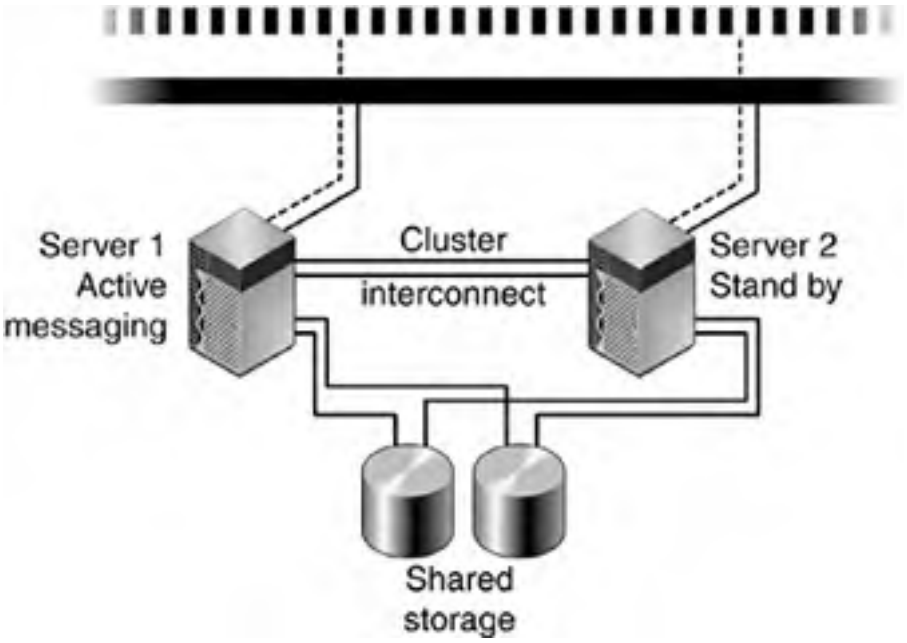
Other Architectures

When you consider what items within the messaging architecture require failover or can take advantage of failover, plus any addition services (such as the Calendar Server) that are often integrated into such an environment, the possible number of architectures increases.

Alternative No. 1

In any environment, having to provide a server for a *hot standby* architecture is wasteful use of computing resources. The alternative configuration (FIGURE 14-1) that some customers have implemented has the Sun ONE Messaging Server environment (mailstore and MTA) running on one system and the main LDAP server running on the other node. This configuration provides for high availability of both messaging and directory, while allowing independent failover of each, plus utilizes both nodes. Additional directory replicas, called *consumers*, can be configured to replicate from the main LDAP server.

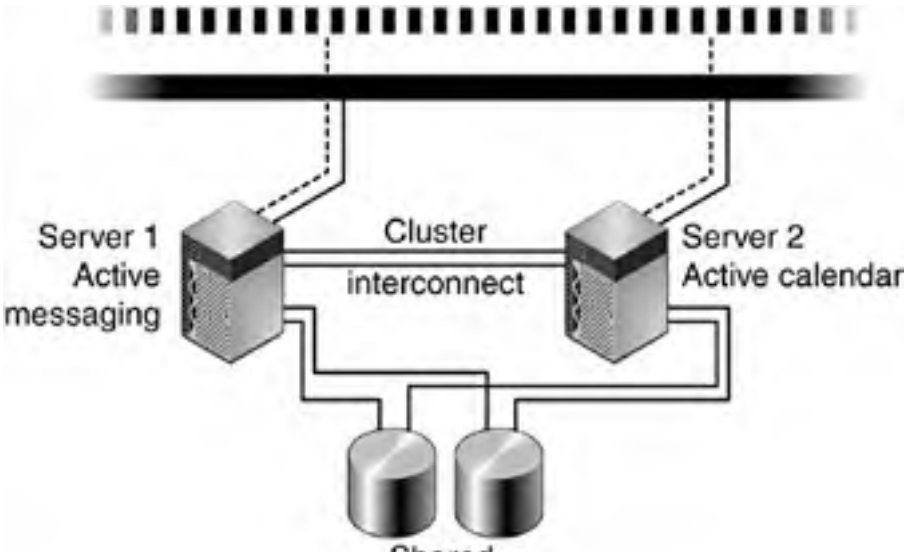
Figure 14-1. High Availability Configuration Failover

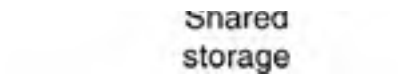


Alternative No. 2

Customers often implement the Sun ONE Calendar Server in addition to the Sun ONE Messaging Server, since they can be purchased in a money-saving package called the Web Communication Bundle. This alternative configuration (FIGURE 14-2) provides for a highly available calendar system in addition to the messaging system. As in Alternative No. 1, the directory server is made highly available on the second server, but now the calendar server is added to the system. This configuration provides for high availability for messaging, calendar, and directory while allowing independent failover of each, plus it utilizes both nodes. As in Alternative No. 1, additional directory replicas can be configured off the main LDAP server.

Figure 14-2. Failover Using Both Nodes in a High Availability Configuration





Differences in Planning for High Availability Messaging

Planning for use of failover software involves obtaining and managing additional IP addresses and hostnames.

Differences in Installing HA Messaging

The obvious difference is that you have to install, configure, and manage the failover software such as Sun Cluster or VERITAS. Beyond that, the largest differences in installing messaging on a clustered system involving dealing with the logical host. *Always* use the fully qualified logical host name and IP address. Do not use the physical host. You must also use the logical storage devices. The differences include references to things such as the LDAP server when installing messaging. Do not refer to the physical host of the LDAP server, but rather to the logical host. There are also some edits to configuration files that must be performed as you will see.

Best Practices and Caveats

Caveat—While everything works well with Sun Cluster for failover on an ACTIVE-ACTIVE cluster configuration, there is one slight issue. Specifically, the Simple Network Management Protocol (SNMP) monitoring daemon is not able to understand that you now have two message servers running on the same physical host, and it goes away so you no longer have a monitoring daemon.

Installation Procedure and Notes

For complete details, see Chapter 4, "High Availability" in the *iPlanet Messaging Server Installation Guide for UNIX* located at:

http://docs.sun.com/source/816-6014-10/ha.htm#11284.

This section came about from a situation where one of our customers was having significant difficulty getting the Sun ONE Messaging Server installed with Sun Cluster 3.0 software and the EMC storage units. The customer we were doing this lab work for spent about four weeks dealing with hardware installation issues that were related to their EMC storage system. So do not underestimate the time it takes to install and physically configure the hardware.

Note

When installing the Solaris OE you must select the Entire Distribution, and you really should select NONE for naming service and manually configure DNS. For example, edit the /etc/nsswitch.conf file and configure the /etc/resolv.conf file.

Conclusions

1. Verifying that the hardware configuration is correct and supported is very critical. The fact that a fiber channel card or driver is not quite correct can lead to significant delays and errors. Messages like "SCSI resets" or "SCSI reservation" problems are indicative of storage issues. Check, recheck, and escalate when all else fails.
    - Check Sun for the latest supported configurations.
    - If using EMC, check with EMC local technical resources to verify that specific interface cards and drivers are supported. EMC does the certification work for Sun Cluster hardware, not Sun.
2. It is critical to do a complete install of the operating system, using the latest version of Solaris OE available.

    In our case, this was Solaris 8 OE Update 10/01. We discovered this the hard way on the customer site, where they had used the JumpStart feature to load their "standard" data center load. This was a mistake for two separate reasons—their JumpStart image used an older version of Solaris 8, Update 01/01, which even with patches is not the same as starting with Solaris 8 OE Update 10/01; and their JumpStart image, while containing innocuous settings for things like DNS, host tables, and so forth, was not the full install of Solaris OE—they had removed packages to "tighten" security and save space. So we lost about a day and a half struggling with some issues, eventually reloading the operating systems on both cluster nodes from a Solaris 8 OE Update 10/01 CD.
3. Overall, the software installation process is not difficult.

    In our lab, we completed the whole installation in roughly nine hours, including breaks for lunch, other meetings, and conference calls. But this was with two people, one with Sun Cluster 3.0 software knowledge and one with iPlanet Messaging Server knowledge, both of whom know the Solaris OE well.

    Once EMC hardware issues were resolved at the customer site, after a total of five weeks of diagnostic and troubleshooting efforts, the installation of the Sun Cluster software and iPlanet Messaging 5.1 proceeded normally.
4. The Delegated Administrator can definitely be made part of the resource group for messaging, using the Sun Cluster Netscape Webserver agent.

    In our lab, we installed the Webserver agent and made it dependent upon the messaging server being up and running (which also made it dependent upon LDAP and storage being there). It worked just fine.
5. The Sun Cluster 3.0 software is significantly different (in many ways better and easier) than Sun Cluster 2.2, so there are some things when architecting it that must be taken into consideration.

    While at least two interconnect links are still required, storage is abstracted from the actual applications, meaning that the application can failover to the other nodes, but storage may not actually migrate. Therefore, it is critical to ensure that you have sufficient bandwidth between the nodes to handle the situation where the messaging server might failover but for some reason storage continues to work on the original node.

    Get the regional Sun Cluster pre-sales engineer and post-sales support engineer involved in approving the configuration. Be extremely detailed about how things will be configured. Little mistakes can take weeks to correct, and waste weeks of time. Planning how things are architected before you begin saves considerable time and reduces the number of situations where you might have to start over or have additional logical hosts. This includes deciding whether you are running the LDAP within the messaging resource group, or whether it is an independent resource group by itself.

Chapter 15. Managing Messaging Services and Preventive Maintenance

As with any system, your Messaging Server requires routine maintenance. This chapter outlines the best practices and issues surrounding day-to-day and routine maintenance involved in managing a messaging server, specifically the Sun ONE Messaging Server. While the current documentation explains the basic commands, it does not address automation or scripting of these functions, nor does it adequately cover techniques that can improve backup and recovery time.

Periodic maintenance is a necessary part of the operation of a messaging system, and the Sun ONE Messaging Server is no exception. By keeping up with maintenance tasks, you can avoid issues that would otherwise occur. This benefit was alluded to in Chapter 14, "Highly Available Messaging Deployment," on page 201."

Keep in mind also that the following are only suggestions. Each organization must develop its own checklists and schedules according to its specific requirements. Additionally, new best practices and maintenance utilities will be developed from time to time, so do not expect your checklists to be completely static either—you must periodically revisit your checklists to take advantage of new developments.

Periodic Maintenance Checklists

It is a good idea to create and maintain checklists for your periodic maintenance. Documented procedures, policies, and checklists are always more consistent than trying to recall whether the system has been patched or backed up.

This section contains descriptions of daily, weekly, monthly, quarterly, and annual checks.

Daily Checks

The items that you should check daily are:

- Review log files for abnormalities.

  Yes, lots of data is logged and it is a pain to review the log files. It is very easy to skip this, but reviewing the log files is also one of the easiest ways to detect errors or abnormalities before they become problems. Often, errors or abnormalities can get buried in the log. The key is to look for specific keywords or filter out lines that are normal. Some people simply write a Perl script or shell script to filter log files for the exceptions. Some organizations have a log scanning utility that they use for other purposes (for example, operating system log file scanning). Some even go to the additional step of adding notification (for example, paging) for a more robust and active method of problem detection.

  By reviewing the log files daily or automating it, you can catch abnormalities or security issues before they cause major problems.

  Do not let automation make you complacent—look yourself sometimes. There is no substitute for the best pattern recognition system—you. No utility or script can be as adaptive as the human brain.

- Check for core files.

  A core file is an indication that a fatal error has happened on the server. A program or process could generate a core file, or if the problem is very serious, the operating system itself could generate a core file. One issue that many system administrators do not address is the space necessary to store a system core of a machine (server) that has four gigabytes or more memory—if the default is kept, often a system core will not be captured due to lack of disk space. It is generally a good idea to configure a separate volume with enough disk space for 2.5 to 3 times the amount of physical memory. There are other settings that usually must be made to enable large files greater that two gigabytes and tell the operating system where to put system cores. See your operating system administrator's manual for specific details.

  Core files for programs and processes generally are not as large as system cores and wind up where the program or process resides.

- Review queues.

  You want to examine all the queues in the system, both on the MTA and the mailstore, to ensure that all the mail is being delivered (passing through) rather than being stuck. This means *all* the queues—not just the ones you normally use, but every queue that is configured. A configuration change or a strangely formatted email may cause one or two messages to be diverted to a queue that is not normally used. A significant number of messages stacked up in the normal queues might also indicate that there is a problem. So by checking all the queues daily, you can get an indication of any problems before they cause major issues.

- Back up the messaging database online.

  One of the nice features of version 5.2 of the Sun ONE Messaging Server is that the administrator now has the ability to perform an online backup of the messaging database. This database stores the header information and folder index information for messages in a particular mailstore (each mailstore has one database). While you can re-create this entire database from messaging contents, this backup can take a significant amount of time.

  To avoid having to re-create the database from scratch, performing a periodic backup (daily or several times per day) can reduce the recovery time immensely. The system can then simply perform an update to the database, which is many times faster than performing even a parallel re-create (re-index).

- Back up the mailstore.

  Depending upon your specific environment and policies, backing up the mailstore (messages) is a good thing. The utilities provided as part of the Sun ONE Messaging Server product can perform a complete backup or a backup based by groups of mailboxes. It can back up to tape or to disk, and it can be integrated with third-party backup utilities such as Legato or VERITAS.

  Keep in mind that the backup utilities maintain single message copy integrity.

- Back up the directory.

  It is necessary to back up the directory (LDAP server) contents separately at the same time. Utilities provided as part of the directory server to do this backup. Since the directory is generally much smaller than the backup of the mailstore, it is rather quick to perform a directory backup. Often the backup is made to disk and then copied onto tape since it does not take up a lot of space.

  There is some debate as to the best method of backup. Two utilities are provided with the directory server, db2bak and db2ldif. The db2bak utility creates a backup in the backup format, and the db2ldif creates a backup in LDIF format. The reason for using db2bak for backups is that it is quicker to restore; the reason for using db2ldif is that it is a neutral, human-readable format so you can import it into most directory servers and directly manipulate it if necessary. Some organizations actually perform backups using both methods, just in case.

- Review new OS or program security patches.

  It is *very* important to keep up to date regarding security patches for both the operating system and programs (messaging, directory, and so on), even above other recommended patches. I put this in the daily category because you should subscribe to the CERT mailing list for security (http://www.cert.org/) to receive notices regarding security-related issues. Then *read and understand* how each applies to your messaging environment —many may not, but there is always the one that will. You can also find the latest Solaris OE security bulletin for Solaris OE-specific security issues at:

  http://sunsolve.sun.com/pub-cgi/secBulletin.pl?mode=latest.

Weekly Checks

The items that you should check weekly are:

- Back up the operating system.

  Ideally, the operating system does not change very much from day to day or even from week to week with a Sun ONE Messaging System installation. User information is not stored at the operating system level, nor is most configuration information, so generally a weekly full backup is sufficient.

- Do a full backup of the mailstore.

  Notice that this is listed twice. Many times, customers perform an incremental backup of the mailstore nightly and then only perform full backups weekly. And yes, some customers do not back up email at all. Imagine 1,000,000 mailboxes each with 10 megabytes of mail. That is 10,000,000 megabytes or 10 terabytes of data to back up. Also, some customers believe that backing up their email opens the door for search warrants and sets expectations with customers of individual email recovery.

- Review new OS recommended clusters.

  Sun releases recommended operating system patches in a bundle or cluster roughly once every two weeks. This includes any security patches, plus kernel and other system programs. It is a good idea to review the report for the latest Recommended and Security Patch Report file for your particular version of Solaris. This report is located at:

  http://sunsolve.sun.com/pub-cgi/show.pl?target=patches/patch-license&nav=pub-patches.

  You can also be notified of patches and get additional updates emailed to you weekly. This service is called the Patch Club—subscribe at:

  http://www.sun.com/newsletters/.

  You can also sign up for Sun Alert Weekly which provides alerts regarding additional issues that may affect the availability of your system.

Monthly Checks

The items that you should check monthly are:

- Review hotfixes and patches for messaging.

  There are two different types of patches for the messaging and directory software. One is called a *hotfix* and the other is called a *patch*. Hotfixes are designed to address one specific issue or problem that a particular customer or small group of customers is experiencing. Hotfixes are not tested against one another—so hotfix 12 is not tested with hotfix 11, for example. This is not always the case, but in general is what happens.

Hotfixes may also address any specific security or corruption issues, so it is important to understand why the latest hotfix has been developed and what it fixes. Note that the description will indicate also whether this might be a cumulative hotfix.

Patches are cumulative of most of the hotfixes since the last patch or release of the messaging and directory software. They have gone through the entire QA cycle and are designed for general application and usage by all customers.

Both hotfixes and patches are applied in the same way, and both change the messaging or directory binaries. A readme file that details the necessary installation steps is available. Details regarding post-installation steps and backout are in this readme file. The installation script creates a backup of any binary replaced

Caution

Hotfixes and patches can undo customizations and changes that have been made. This is especially prevalent when customers have made customizations or changes to the web mail GUI. Reconciling these customizations (redoing) with the updated files can take effort and time—it is not something to be rushed. It is a good idea to apply hotfixes and patches to a test system, determine any reconciliation required, redo the customizations on the test system and thoroughly test them, and then move the reconciled files onto the production mail server.

- Export the directory.

  Directory export (for example, using the db2ldif utility) is listed as a monthly checklist item because it is a good idea overall and should be done periodically if you are not using this method for backups.

- Review sum of database file sizes

  You should periodically review the sum of the database file sizes so you can properly tune the store.dbcachesize parameter. See Chapter 12, "Performance Tuning," on page 179" for more details.

Quarterly Checks

The items that you should check quarterly are:

- Practice recovery of the messaging system from scratch.

  Practicing the recovery of the messaging system from scratch is often overlooked. Yet as stated before, it is often the time that it takes to recover that really impacts downtime, not the actual hardware failover. Often the first time an organization actually does this is during a real outage—not the best time to be trying new procedures or not knowing exactly what you should be doing.

  One argument made against this practice is always the lack of time. Well, would you rather spend eight hours practicing and when something happens be able to recover in two hours, or would you rather skip the practicing and spend 16 hours recovering the messaging system?

  Another argument made is: We have one terabyte of email and do not have a test system with enough storage. OK, well what about using a subset of the mailboxes? Create a specific *test* backup tape with every tenth mailbox so you only need 100 gigabytes of storage.

  Yet another argument is not having enough servers. Fine. Put everything all on the same system, directory, MTA and mailstore; at least it is better than nothing. You will lose something in the translation but the majority of the steps and procedures will still have to be done.

  On your practice system, intentionally corrupt or drop the database on the mailstore. Now try specifically to re-index or recover from a backup of the data to practice this part of the recovery. The database on the mailstore will be corrupted more frequently than the entire system, so practice recovery and rebuild of just the database and figure out where to reduce time (for example, using backups of the database and/or parallel rebuilds).

  By doing a recovery of the messaging system from scratch quarterly, you will understand the overall process and be comfortable executing the necessary steps.

Annual Checks

The items that you should check annually are:

- Review procedures and checklists.

- Evaluate the latest version of the messaging software for possible upgrade.

Chapter 16. Monitoring a Sun ONE Messaging Server

Monitoring your systems and the Sun ONE Messaging Server software that comprises your email infrastructure is an important part of the overall management effort. Tools can range from simple monitoring of the basic hardware and network infrastructure to more complex monitoring such as response time and error logging. They can be homegrown, open source, or commercial products. You can implement one or many.

The important part of the management effort is to understand that such tools exist, map out your specific needs with regard to what you want to monitor and what data you want to keep or graph over X period of time, then examine the tools available to see what meets your needs (or most of your needs).

SNMP

Since version 5.1 of the Sun ONE Messaging Server product, support for the SNMP protocol has been available. Using an SNMP client (sometimes called a network manager) such as Sun Net Manager or HP OpenView (not provided as part of the messaging server product), you can monitor certain parts of the Sun ONE Messaging Server.

The Messaging Server implements two standardized management information bases (MIBs), the Network Services Monitoring MIB (RFC 2788) and the Mail Monitoring MIB (RFC 2789). The Network Services Monitoring MIB provides for the monitoring of network services such as POP, IMAP, HTTP, and SMTP servers. The Mail Monitoring MIB provides for the monitoring of MTAs. The Mail Monitoring MIB allows monitoring of the active and historical state of each MTA channel. The active information focuses on currently queued messages and open network connections (for example, counts of queued messages or source IP addresses of open network connections), while the historical information provides cumulative totals (for example, total messages processed, total inbound connections).

SNMP is not enabled by default and must be configured. See "Appendix A " of the *Sun ONE Messaging Server Administrator's Guide* at:

http://docs.sun.com/source/816-6009-10/snmp.htm#23526.

SNMP monitoring is fine for organizations that already have this in place, but it can be awfully burdensome to implement SNMP just for monitoring a single application—though it does perform a most valuable service.

Some of the following applications are listed as alternatives or additions to the SNMP method of system and application monitoring.

Alternative Tools

This section describes the following alternative tools:

- What's Up Gold
- Sun Management Center
- Orca
- Big Brother
- BMC Patrol

What's Up Gold

What's Up Gold is a *very* basic commercial monitoring tool with some nice features. Its main advantage is that it is easy to install, configure, and get working very quickly. It monitors whether a server (via TCP or UDP) is available via the network—in other words, "what's up?" Another thing in its favor is that it requires no agents or anything loaded on to the servers themselves. A downside is that it has no specific hardware or software knowledge, so it does not monitor specific applications or hardware—there is no performance or throughput data. The biggest downside is that it requires a Windows system (98/ME/NT/2000/XP).

With Sun™ Management Center 3.0 Basic being offered at no charge for Sun systems, the need for something as basic as What's Up Gold has been greatly diminished.

There might also be some usable open-source offerings in lieu of What's Up Gold, which is located at:

http://www.ipswitch.com/Products/WhatsUp/index.html.

Sun Management Center

Sun Management Center software is an open, extensible system monitoring and management solution that uses the Java software protocol and SNMP to provide an integrated and comprehensive enterprise-wide management of Sun products and their subsystems, components, and peripheral devices. Sun Management Center technology provides a solution to extend and enhance the management capability of Sun's hardware and software solutions.

Sun Management Center Basic Edition is available at no charge, and provides basic monitoring features for a single server. The Sun Management Center Enterprise Edition provides the ability to monitor a large number of servers and systems in a client-server configuration, with an agent running on each server and system to be monitored. The data gathered by these agents is then collected by a central server and viewed by the Sun Management Center main management console. Additional components can be added on to the Sun Management Center Enterprise Edition for various enhanced capabilities and applications.

Sun Management Center works with accompanying software packages: Service Availability Manager, a set of modules that test and measure the availability of network services, as well as a System Reliability Manager—a component that enhances reliability, helping to increase service levels and decrease administrative costs, and a Performance Reporting Manager—software that adds analysis, reporting, and graphing capabilities.

The newest addition to the Sun Management Center component list is Change Manager. Based on the concept of managing and provisioning entire software configurations as a single, integrated software stack, the Sun Management Center Change Manager software delivers a fast and easy way to install, configure, upgrade, provision, and audit the integrated software application payloads running on your systems.

An important point to note is that there are application specific modules that plug into the Sun Management Center. These modules are developed by Halcyon Inc. and are compatible with the Sun ONE Management Center 3.0 product. The modules of interest are:

- PrimeAlert for Sun ONE Directory Server
- PrimeAlert for Sun ONE Messaging Server
- PrimeAlert for Sun ONE Web Server

If you are using the VERITAS Cluster Server for failover, you might be interested in:

- PrimeAlert for Veritas Cluster Server

For details see:

http://wwws.sun.com/software/solaris/sunmanagementcenter/index.html and

http://www.halcyoninc.com/downloads/home.html.

Orca

Orca is a general web-based graphing package. However, combined with the SE Toolkit (which collects system data) it is a nice foundation for keeping track of overall system performance data over a period of time. Performance and throughput data from the Sun ONE Messaging Server can be easily incorporated and graphed using Orca.

While Orca (Orcallator) might not be flashy, it does provide a good start for simple monitoring of the system and application data.

For details see:

http://www.orcaware.com/orca/ and

http://www.setoolkit.com/.

Big Brother

Big Brother monitors system and services for availability. It is a web-based tool with the status of your various systems and services displayed on a color-coded web page in near-real time. When problems are detected, administrators can be notified by email, pager, or text messaging. Big Brother has a pretty good following for many reasons, including their licensing policy and availability of source. The original Big Brother is free for non-commercial use, as defined by its license. Big Brother is provided in source code format for UNIX and Linux, and precompiled for Windows NT and Windows 2000.

Big Brother extensions for Sun ONE Messaging and Sun ONE Calendar are available from Sun "as is" upon request from Sun. Other extensions can be found at the Big Brother archive site. For details see:

http://bb4.com/ and

http://www.deadcat.net/.

BMC Patrol

BMC Patrol is a commercial software application monitoring product. While it can be configured to monitor some of the basic system functions, to get the most out of BMC Patrol implementation you must have two *knowledge modules*. Knowledge Modules are additional extensions and pre-configured thresholds for warnings and alerts for the BMC Patrol software. The two specific modules of value in a Sun ONE messaging environment are:

- Solaris Knowledge Module
- Sun ONE Messaging Server Knowledge Module

The Sun ONE Messaging Server Knowledge Module specifically provides proactive monitoring of key messaging server components including LDAP Server, Messaging Server including SMTP Server, IMAP/POP Server, WebMail Server, Administration Server, and the underlying message store.

Additional modules for some specific hardware (for example, Sun Fire™ F15K) and configurations (for Sun Cluster software) are also available.

Information about these Sun-specific modules is located at:

http://www.sun.com/service/sunps/systemsandnetworkmanagement/bmcpatrol/ and

http://www.bmc.com/.

or contact your local Sun Sales Representative.

Appendix A. Case Studies

It is always useful for customers to see real-world examples of Sun ONE Messaging Server implementations and architectures. Sometimes this is critical due to implementation time constraints or it may be simply a matter of gathering reference points. The case studies in this appendix serve this purpose.

The following sections contain a series of case studies to illustrate several points made throughout this book as well as highlight some specific lessons learned. Architecture diagrams and timelines are provided for reference. These cases occurred over the past few years and are actually a composite of the case studies of several different customers.

This appendix contains the following case studies:

- Acme University
- Baker Tech
- Community City College

Additional case studies will be gathered in the future.

Acme University

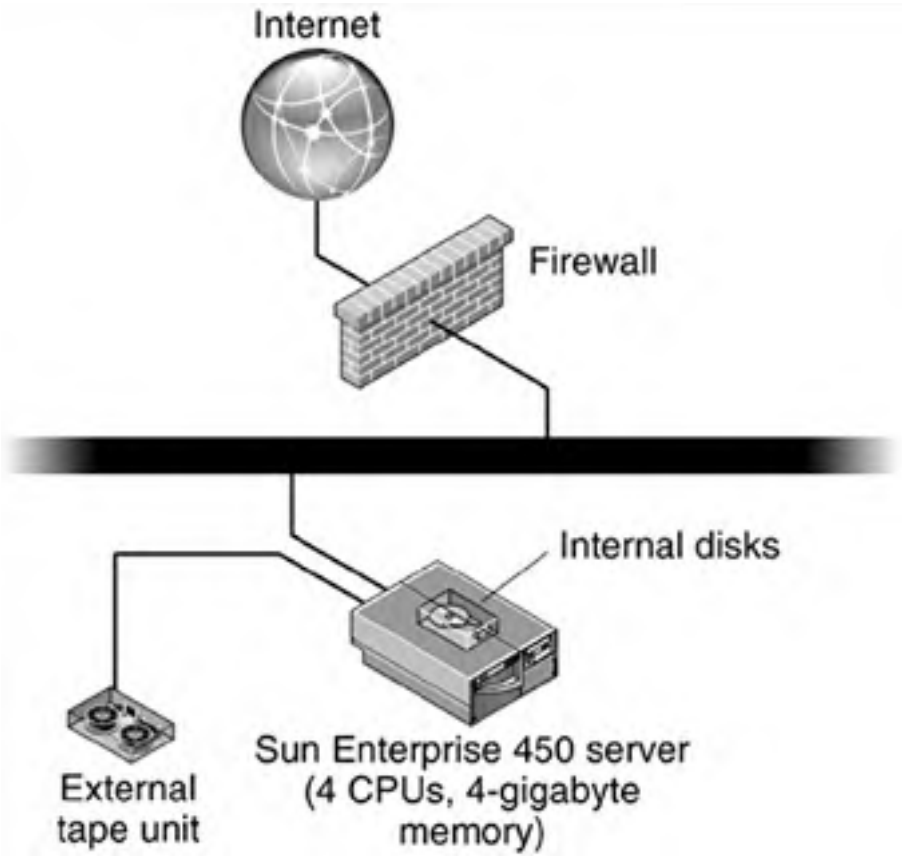
Acme University wanted to replace their existing sendmail system running on a small Sun system. It had been in place for approximately five years. They wanted a more secure system with room for growth while maintaining the same level of administration effort. The sendmail system leveraged files (for example, /etc/passwd) for user information. There was no system redundancy other than basic protection (for example, RAID 0+1) for disks. The customer would like to have most users (students) use web mail while reserving IMAP for faculty and staff. Ultimately they would like to eliminate POP if possible. They are satisfied with their current backup method of direct-attached tape backup.

Acme University has:

- 5,500 students
- 1,000 faculty, staff, and other employees

A single Sun Enterprise 450 server with four CPUs, four gigabytes of memory and 12 internal disk drives configured for RAID 0+1 was purchased along with a small Netra™ server (single CPU, dual network interfaces) for an SMTP firewall product (Interscan) for which the customer previously purchased a licence. Professional services from a local reseller assisted the customer in the setup of the hardware and the initial installation of the Messaging Server. A single DLT 7000 was directly attached to the Sun Enterprise 450 server for backup. No special software was to be used for backups. User information and mail was migrated en masse during the summer break and semester. Figure A-1 shows the Acme University architecture configuration.

Figure A-1. Acme University Architecture Diagram



Timeline

The purchase and implementation of the new messaging system took approximately four months from initial contact with Sun to the first production login. The initial architecture and purchase was done in four weeks and the equipment was on site approximately two weeks after that. Installation work began in week six. The basic messaging system installation was done in about two weeks The remaining two months was used to migrate existing users and mail and develop scripts to automate the provisioning of the user information.

Lessons Learned

The following lessons were learned in this case study:

- Do not short storage.

  One of the original assumptions made by the customer was that 30 gigabytes of space was sufficient for their environment. Unfortunately they did not take into consideration issues such as spindle count and file system requirements. Unlike many servers, messaging servers tend to need quick transactional storage as well as bulk storage. In this case, the specific issue was that no separate volume or spindle set was allocated to their message queues. This introduced some performance issues. Once a dedicated volume was configured for their MTA queues, the performance issue abated. Luckily the performance issue was noticed early on and did not turn into a major issue.

- Keep it simple is a good idea.

  In one of the initial discussions, the customer expressed an interest to keep things to a minimalist architecture because administrative staff was scarce and they did not want to add an administrative burden. By keeping the configuration simple with normal hardware availability efforts such as dual power supplies, RAID-0+1 protected storage, and a solid backup device, the customer received several benefits:

  - Much faster installation and configuration
  - Easier administration and management
  - Only one server to deal with when issues arose, like the performance issue
  - Lowest possible cost
  - Availability without extraordinary measures or complications
  - Training is important

  Initially the customer was reluctant to send their only administrator through training, even though they had paid for the class. Some of this apprehension was due to their limited staff and already busy workload, plus the additional costs of travel. Initially some of the up-front training was done by using web-based courses. However, it became very clear when examining the Sun Service Support call log about three months after installation that it was time to take the class, because some of the questions and issues could have been easily avoided. While there is no substitute for hands-on experience, getting the basics of installation, configuration, and operation is critical.

- Installation assistance makes things smoother.

  The local Sun Reseller provided experienced help to assist the customer in the installation and deployment of the new Sun ONE Messaging Server. This was critical in transferring knowledge and getting the system up and running initially.

Baker Tech

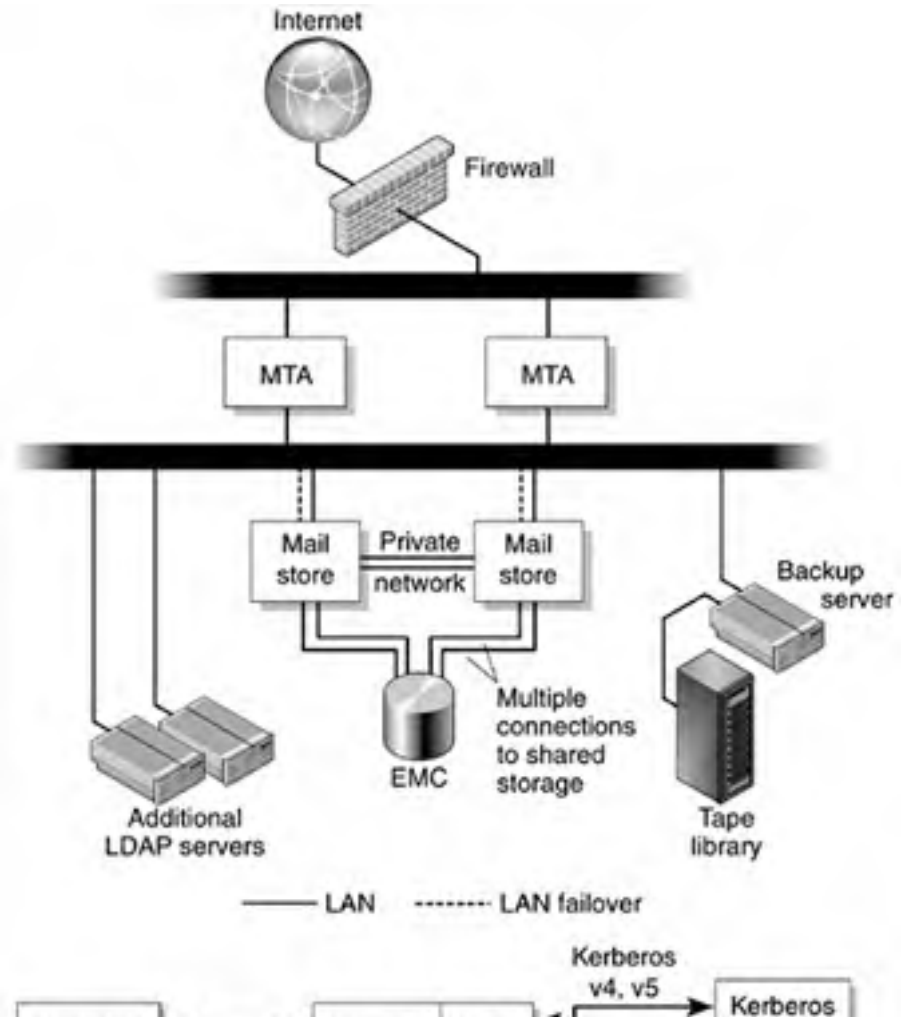
This large university had numerous mail servers across the campus and wanted to consolidate their infrastructure. They had some experience with directory technology (LDAP), but no single campus-wide directory yet. A single authentication system was being developed around Kerberos. They had no web mail or it varied between the mail systems on campus as to whether it was offered or not. They would like a central mail system with web mail that had failover and used directory technology for user information, but can use their Kerberos servers for authentication. Good Sun and Solaris expertise existed in the IT department as well as throughout the campus, but they had little or no experience with clustering technology. It was necessary to support the customer's existing EMC Symmetric storage system. The customer would use existing SNMP tools to monitor the messaging system.

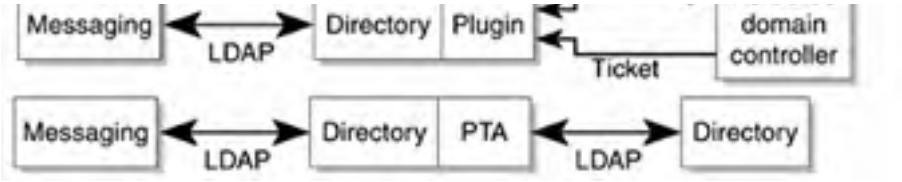
Baker Tech has:

- 40,000 students
- 10,000 faculty, staff, and other employees

A pair of Sun Enterprise 4500 servers with eight CPUs and eight gigabytes of memory was configured as the main mailstores and a pair of Sun Enterprise 280R servers was used for MTA and virus scanning. The architecture was designed to leverage about 1.2 terabytes of the customer's existing EMC storage subsystem and utilize Sun's Sun Cluster 3.0 software for high availability (clustering or failover). Unfortunately the customer still did not have a centralized enterprise Directory, but there were pockets of directory on campus. Additional Netra servers were added to one of their existing directory installations (islands) to support the messaging server's LDAP workload. An open source plug in to the Sun ONE Directory was used to provide Kerberos authentication out the back end of the directory. Figure A-2 shows the Baker Tech architecture configuration.

Figure A-2. Baker Tech Architecture Diagram





User information was already partially available, so the messaging server objects needed to be added to the directory and applied to the users.

They decided to add all new accounts to this system beginning with the next semester after going live, while allowing all other users the option to migrate. This policy would be revisited each year. Backups were integrated with their existing data center backup infrastructure using Legato Backup and a tape library. They elected to do the majority of the implementation themselves due to their experience level with Sun and Solaris, even though they had no experience with the messaging product. This implementation method was not recommended by Sun.

Timeline

The overall project took eight months from start to finish while the initial plan called for an aggressive three-month window. Several factors that contributed to the project delays are outlined the "Lessons Learned" section. The initial purchase from the initial contact to placement of the order took approximately eight weeks even though the customer's internal project plan was designed around a two week purchase cycle. The main delay was due to issues within the purchasing department and the requirements of their procedures and processes. Equipment was delivered to the customer in three weeks once the purchasing issues were resolved.

The initial equipment installation and Solaris set up took approximately a week since the customer had significant Solaris and Sun experience. Then, the installation of the Sun Cluster 3.0 software was started. However, something that should have taken approximately two weeks took almost six weeks due to EMC Symmetric storage unit integration issues. Incorrect adapter cards for the Sun system and incorrect drivers were recommended by EMC and purchased from the customer. After completely swapping out all 10 interface cards and installing the absolute latest driver from EMC for the cards, the EMC storage was able to be attached and failed over without issues. That means that just to get the basic hardware, operating system, and cluster software working took 18 weeks.

Once these initial obstacles and delays were overcome, the actual implementation of the messaging software took approximately two weeks. Load testing, backup restoration testing, and additional testing of the failover process took another three months. This process was started in May and targeted January of the following year, but this schedule was not met and production was delayed until Spring Break of the following year.

Lessons Learned

The following lessons were learned in his case study:

- SNMP has a failure issue.

  During the failover testing with the messaging product, once the failover was working, an issue existed during failover condition where both messaging instances were operating on the same host and SNMP visibility went away. This was not an major issue for the customer as this is a failover condition. Failure of the SNMP monitoring would further enforce the fact that the systems required attention. This may or may not be the case for all customers.

- Instrumenting and monitoring is key.

  During the initial testing of failover and load testing, no monitoring was enabled and many statistics were not being collected. Decisions regarding tuning specific parameters later on was difficult due to lack of data. This meant that some load tests had to be rerun once monitoring was enabled.

- Allow additional time for third-party storage.

  Due to the difficulty and issues encountered, additional time when dealing with third-party hardware or software involved should be added to the project schedule. This can vary widely based upon the product and relationships involved.

- For complex installations Sun Professional Services can make a difference.

  During the installation issues, using Sun Professional Services was brought up again and recommended to the customer. Some of the issues the customer experienced had already been encountered and addressed using Sun Professional Services. Many of the issues that caused significant delays would have been addressed quickly and would not have caused project time slippage.

Community City College

The customer was large community college system with 18 campuses distributed throughout the state. Each campus had at least one mail system and various levels of directory infrastructure, if any at all. The Chancellor's office decided that recent funding cutbacks required consolidation of IT services, including directory and messaging. There were pockets of UNIX administration experience with even less directory expertise and no clustering or failover experience. The use of the existing third-party enterprise storage and backup system (library and software) was required.

The initial thoughts were that 90 percent of the users would be using IMAP while the other 10 percent would use web mail.

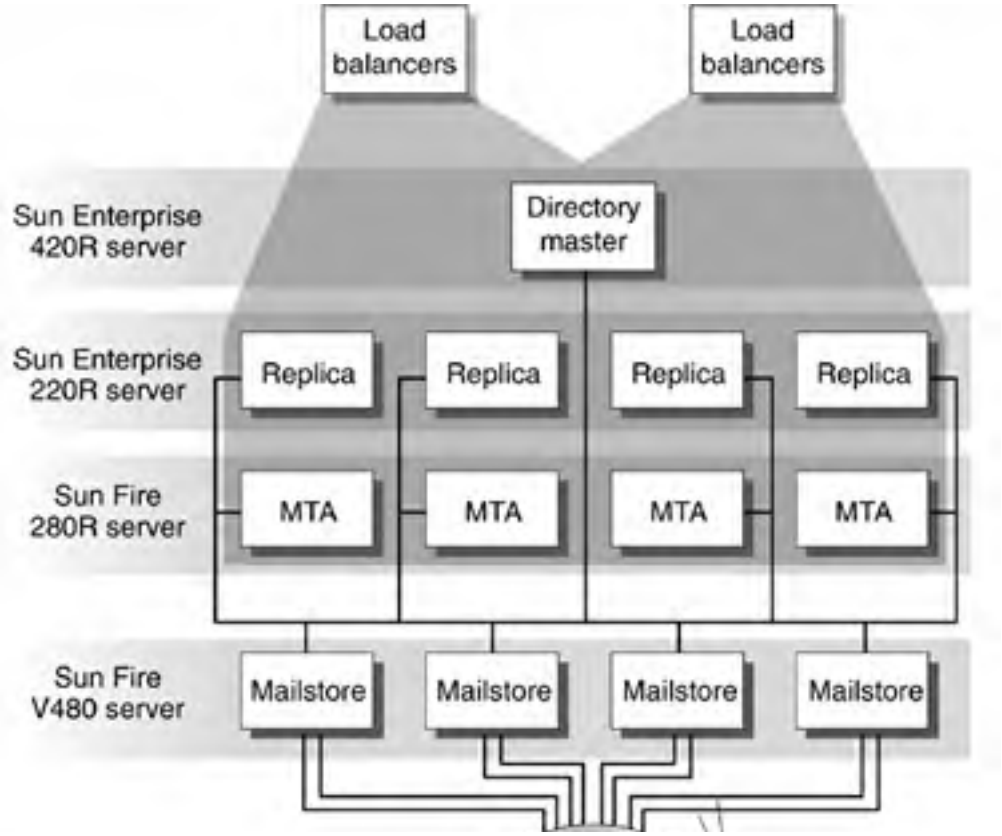
Community City College has:

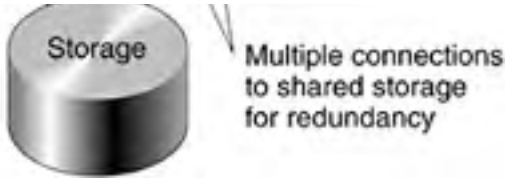
- 120,000 students
- 20,000 faculty, staff, and other employees

Prior to any specific solution, the decision was made to locate the new messaging system at one of the larger, more advanced campuses that had messaging and directory experience. Due to lack of high availability experience, the decision going into the architecture phase was to *not* use failover technology. Availability would be achieved through the use of multiple servers at each level of the architecture. Due to the large number of users, stress and load testing was critical. Migration would be initially only for faculty and staff, plus any new accounts created after the go-live date. No existing students would be migrated during the initial year. Establishing provisioning from the existing centralized student information and HR systems was required. The customer had existing license for antivirus but wanted to implement antispam at a later date.

The proposed configuration consisted of a directory master server using a Sun Enterprise 420R server, four back-end mail servers (mailstores) using Sun Fire V480 servers, each with four CPUs and eight gigabytes of memory. The MTA layer consisted of four Sun Fire 280R servers, each with two CPUs and four gigabytes of memory. To establish the directory environment, a combination of existing systems (old servers) and new was used. The master directory servers were Sun Enterprise 420R with two CPUs and four gigabytes of main memory, while the replicas were Sun Enterprise 220R servers with two CPUs and two gigabytes of memory. Load balancing among the servers was accomplished using the existing Cisco LoadDirectors. Figure A-3 shows the Community City College architecture configuration.

Figure A-3. Community City College Architecture Diagram





Timeline

This project took about one year from start to finish with the purchase of the software and hardware taking approximately two months from date of initial contact with Sun to the delivery of the hardware on site. The initial architecture, sizing, and training were done in parallel once the purchase order was given to Sun and took approximately three to four months. Load testing, instrumenting using Orca and BigBrother, verification of sizing, and practice migration of faculty staff took an additional three months during which scripts to migrate users and integrate the directory provision with the campus student information system were done.

Lessons Learned

The following lessons were learned in this case study:

- PAB size

  The initial architecture and planning called for web mail to be a minor factor, but most of the students used web mail almost exclusively. The actual web mail workload on the main messaging stores was not an issue because IMAP and web mail are close in terms of workload. The main issue was Personal Address Book (PAB) entries. Nine percent of the initial 40,000 accounts used web mail. Each account had an average of 15 entries in the PAB. This situation resulted in over 600,000 directory entries in the PAB portion of the directory. After the initial year of production operation, the decision was made to separate the PAB portion of the Directory onto separate LDAP servers so that they could be tuned and managed separately.

- Recovery

  Unfortunately recovery procedures were not practiced and were necessary due to a storage subsystem failure within the first three months of operation. However, since no one at the customer site had practiced recovery or was aware of specific steps which that could minimize recovery time, recovery took eight times longer than would have otherwise been necessary. Sun Professional Services was called in after this incident to provide a workshop and guidance on developing recovery procedures tailored to the customer's specific environment, so as to reduce the recovery time.

- Appropriate partition sizing

  Initial plans called for a minimum of two partitions to handle mailboxes on each of the message stores, based upon initial planning of 20,000 mailboxes per partition. Due to the backup and recovery issues explained previously, it was determined that even though the Messaging Server is quite capable of managing tens of thousands of mailboxes per partition, it is not necessarily an issue of function of the number of mailboxes, but rather a function of amount of storage per partition in gigabytes. It was decided to repartition the server into a more manageable and recoverable size equivalent to what a single backup tape or image would hold—in this case, roughly 200 gigabytes.

- Load testing

  Load testing in this large environment was critical to tuning of various parameters in each layer of the messaging architecture, and particularly valuable in testing the configuration of the Cisco LoadDirectors. It was discovered that specific settings were not quite correct and needed to be fixed. If the entire pathway had not been load tested, production issues would have occurred if one of the LoadDirectors had failed.

- Periodic maintenance

  During the initial year of operation, little attention was paid to the systems unless dictated by the monitoring tools or help desk ticket system. Upon examination of the directory and messaging system during the PAB migration, it was clear that specific tables in the directory had grown larger than originally anticipated, which necessitated tuning of parameters. Examination of the messaging statistics also indicated some tuning was necessary. In reality, the system had not been periodically maintained other than basic patches or issues surrounding outages. Quarterly examination of data and statistics, table sizes, and so forth could have avoided some slowdowns and outages. This is especially important in large environments where growth is quick.

Appendix B. Majordomo Integration

This appendix describes the procedure for integration working with a single test list. It is tedious, but it does work, and has all of the functionality of majordomo with sendmail. These instructions are for a single domain, but with some minor tweaks they should also work fine in a multidomain environment.

Assumptions:

1. Your messaging server is already installed and functioning correctly.
2. You have gcc installed, or you can compile the wrapper on a machine where it is installed.

Preparing for Integration

To prepare for integration:

1. **Make sure your mailsrv user is assigned to a $HOME directory and that it has write permissions to** *server-root*/**msg-**_hostname_.
2. **Create** /etc/passwd, /etc/shadow**, and** /etc/group **entries for the majordomo user.**

   Example password entries:

   ```
   iplanet:x:1002:101:iPlanet Servers:/opt/iplanet:/bin/ksh
   ldapsrv:x:1003:101:Directory Server User:/opt/iplanet:/bin/ksh
   mailsrv:x:1004:101:Messaging Server User:/opt/iplanet/msg-maxima:/bin/ksh
   icsuser:x:1005:101:Calendar Server User:/opt/iplanet/SUNWics5:/bin/ksh
   listsrv:x:1006:101:Mailing List Manager:/opt/iplanet:/usr/bin/csh
   majordom:x:91:91:Mailing List Manager:/opt/majordom:/usr/bin/bash
   ```

   Shadow entry:

   ```
   majordom:*LK*:::::::
   ```

   Group entries:

   ```
   majordom::91:mailsrv
   iplanet::101:iplanet,ldapsrv,mailsrv,icsuser,listsrv,majordom
   ```

3. **Create the $HOME directory for majordomo.**

   We used /opt/majordom with 775 permissions. We will probably tighten this up to 755 or 751 later.

   a. **Extract the majordomo tarball in a work directory.**

      This is where you will edit the Makefile to fit the environment you created for majordomo.

      [View full width]

      ```
      Makefile <snippet of interest>
      #-------------  Configure these items ----------------#
      #

      # Put the location of your Perl binary here:
      PERL = /usr/bin/perl

      # What do you call your C compiler?
      CC = gcc

      # Where do you want Majordomo to be installed?  This CANNOT be the
      # current directory (where you unpacked the distribution)W_HOME = /opt/majordom.

      # Where do you want man pages to be installed?
      MAN = /usr/local/man

      # You need to have or create a user and group which majordomo will run as.
      # Enter the numeric UID and GID (not their names!) here:
      W_USER =  91
      W_GROUP = 91

      # These set the permissions for all installed files and executables
      # (except the wrapper), respectively.  Some sites may wish to make these more lenient, or
      more restrictive.

      FILE_MODE = 644
      EXEC_MODE = 755
      HOME_MODE = 775

      # If your system is POSIX (e.g. Sun Solaris, SGI Irix 5 and 6, Dec Ultrix MIPS, BSDI or
      other 4.4-based BSD, Linux) use the following four lines.  Do not change these values!
      ```

```
WRAPPER_OWNER = root
WRAPPER_GROUP = $(W_GROUP)
WRAPPER_MODE = 4755
POSIX = -DPOSIX_UID=$(W_USER) -DPOSIX_GID=$(W_GROUP)
# Otherwise, if your system is NOT POSIX (e.g. SunOS 4.x, SGI Irix 4,
# HP DomainOS) then comment out the above four lines and uncomment
# the following four lines.

# WRAPPER_OWNER = $(W_USER)
# WRAPPER_GROUP = $(W_GROUP)
# WRAPPER_MODE = 6755
# POSIX =

# Define this if the majordomo programs should *also* be run in the same
# group as your MTA, usually sendmail.  This is rarely needed, but some
# MTAs require certain group memberships before allowing the message sender to be set
  arbitrarily.

# MAIL_GID =    numeric_gid_of_MTA

# This is the environment that (along with LOGNAME and USER inherited from the
# parent process, and without the leading "W_" in the variable names) gets
# passed to processes run by "wrapper"
W_SHELL = /usr/bin/bash
W_PATH = /bin:/usr/bin:/usr/local/bin
W_MAJORDOMO_CF = $(W_HOME)/majordomo.cf

# A directory for temp files..
TMPDIR = /var/tmp

# -----YOU SHOULDN'T HAVE TO CHANGE ANYTHING BELOW THIS LINE.-----

Now you can:
make wrapper
make install-wrapper
make install

# Once the wrapper and all the perl scripts are installed in /opt/majordom, there are some
  edits required to get Y2K compliance and squash a couple small boogs.  I will address them
  in a different format where possible.

***archive2.pl

155c155,159
&open_archive($FH, $year % 100, $MoY{$moy}, $dom);
---
 if ($year =~ /\d{4}/) {
 &open_archive($FH, $year -1900, $MoY{$moy}, $dom);
 } else {
 &open_archive($FH, $year % 100, $MoY{$moy}, $dom);
 }

***digest
176c176
foreach (@files) {
---
foreach (sort @files) {

***majordomo.pl
59c59,60
s/\n\s+/ /g;
---
 s/\015//g;        # strip DOS <CR>/^M from end of lines
 s/\n\s+/ /g;      # unfold wrapped headers

# Note, the ^M is a single character created by typing ctrl-v then ctrl-m.

***resend
591c591

---
s/\015//g;      # strip DOS <CR>/^M from end of lines

***majordomo.cf
9c9
$whereami = "example.com";
---
```

```
$whereami = "sonny.org";
25c25
$homedir = "/usr/test/majordomo";
---
$homedir = "/opt/majordom";
27a28
$datadir = "$homedir/data";
30c31
$listdir = "$homedir/lists";
---
$listdir = "$datadir/lists";
38c39
$digest_work_dir = "/usr/local/mail/digest";
---
$digest_work_dir = "$datadir/digests";
42c43
$log = "$homedir/Log";
---
$log = "$datadir/Log";
101a103
$config'default_unsubscribe_policy = "open+confirm";
137,138c139,140
$filedir = "$listdir";
$filedir_suffix = ".archive";
---
$filedir = "$datadir/archives";
$filedir_suffix = "";
159c161
$majordomo_request = 0;
---
$majordomo_request = 1;
167c169
max_which_hits = 0;
---
$max_which_hits = 1;
193c195
$TMPDIR = $ENV{'TMPDIR'} || "/var/tmp";
---
$TMPDIR = $ENV{'TMPDIR'} || "$datadir/tmp";
```

b. **You must create some subdirectories for majordomo to use in its** $HOME **directory to match the entries in the** majordomo.cf **file.**

   While you are at it, you can consider creating a link to majordomo.cf in /etc. This is good preventive medicine.

   As root, execute these commands, in order:

```
su majordom
cd /opt/majordom
mkdir -m 775 data
cd data
mkdir -m 775 archives digests lists tmp
cd archives
mkdir -m 775 test test-digest
cd ../digests
mkdir -m 775 test-digest
cd ../data/lists
touch test test-digest
exit
cd /etc

ln -s /opt/majordom/majordomo.cf majordomo.cf
cd /opt/iplanet/msg-maxima/imta/programs
su mailsrv
ln -s /opt/majordom/wrapper wrapper
exit
```

   The test* subdirectories are the beginnings of a test mailing list setup and configuration. Majordomo still does not work with the messaging server until you create *methods* for *program delivery*, the proper users in LDAP, and appropriate entries in the imta/config/aliases file. Start with the program methods.

c. **If you are still** root**, change directories to the** *server-root*/msg-*hostname* **directory (in our case** /opt/iplanet/msg-maxima**).**

   The imsimta program utility adds the methods you need to LDAP. This one is for the majordomo administrative user:

```
/imsimta program -a -m mjwrapper -p wrapper -g "majordomo" -e postmaster
```

d. **Create a set of these for each list you create:**

[View full width]

```
/imsimta program -a -m testr -p wrapper -g "resend -l test test-outgoing" -e user`
/imsimta program -a -m testa -p wrapper -g "archive2.pl -f /opt/majordom/data/archives
 /test/test -a M" -e postmaster`
/imsimta program -a -m testd -p wrapper -g "digest -r -C -l test-digest
 test-digest-outgoing" -e postmaster`
/imsimta program -a -m testq -p wrapper -g "majordomo -l test" -e postmaster`
```

The last entry could also be written as follows if the majordomo.cf used $majordomo_request = 0:

```
/imsimta program -a -m testq -p wrapper -g "request-send test" -e postmaster
```

Make sure to refresh the MTA after making additions and changes like these, but since you are also going to modify the aliases and add user entries to LDAP, you can hold off on the refresh for a bit.

An ./imsimta program -l should now produce output like the following:

```
================================================
Method_name        : mjwrapper
Program_name       : /opt/iplanet/msg-maxima/imta/programs/wrapper

Argument_list      : majordomo
Execute Permission : User
================================================
Method_name        : testr
Program_name       : /opt/iplanet/msg-maxima/imta/programs/wrapper
Argument_list      : resend -l test test-outgoing
Execute Permission : Postmaster
================================================
Method_name        : testa
Program_name       : /opt/iplanet/msg-maxima/imta/programs/wrapper
Argument_list      : archive2.pl -f /opt/majrdomo/data/archives/test/test -a -M
Execute Permission : Postmaster
================================================
Method_name        : testd
Program_name       : /opt/iplanet/msg-maxima/imta/programs/wrapper
Argument_list      : digest -r -C -l test-digest test-digest-outgoing
Execute Permission : Postmaster
================================================
Method_name        : testq
Program_name       : /opt/iplanet/msg-maxima/imta/programs/wrapper
Argument_list      : majordomo -l test
Execute Permission : Postmaster
================================================
```

We were not able to get -l test passed to the mjwrapper method correctly, or we might have been able to save one piece of work here too. We do not have a percent variable to use for that. Otherwise, everywhere in the argument_list that the word "test" exists could be an argument passed from the mailprogramdeliveryinfo attribute.

The additional entries you will need in msg-maxima/imta/config/aliases are:

```
owner-majordomo@sonny.org: dliston@ims-ms-daemon
owner-test@sonny.org: dliston@ims-ms-daemon
owner-test-digest@sonny.org: dliston@ims-ms-daemon
test-outgoing@sonny.org: </opt/majordom/data/lists/test
test-digest-outgoing@sonny.org: </opt/majordom/data/lists/test-digest
```

These are the LDAP user entries you will need to pull it all together. None of the LDAP entries need mailListCreate nsda capability, but the attribute might be handy as part of an ACL if majordomo ever becomes LDAP aware.

```
dn: uid=majordom,ou=people,o=sonny.org,o=isp
```

objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
objectclass: inetUser
objectclass: ipUser
objectclass: nsManagedPerson
objectclass: userPresenceProfile
objectclass: inetMailUser
objectclass: inetLocalMailRecipient
mailuserstatus: active
inetuserstatus: active
datasource: NDA 4.5 Delegated Administrator
nsdacapability: mailListCreate
userpassword: {crypt}4/Y1B.C4RmLnU
uid: majordom
givenname: Majordomo
sn: List Manager
cn: Majordomo List Manager
preferredlanguage: en
maildeliveryoption: program
mailprogramdeliveryinfo: mjwrapper
mailhost: maxima.liston.nu
mail: majordomo@sonny.org
mailalternateaddress: majordom@sonny.org

dn: uid=test,ou=people,o=sonny.org,o=isp
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
objectclass: inetUser
objectclass: ipUser
objectclass: nsManagedPerson
objectclass: userPresenceProfile
objectclass: inetMailUser
objectclass: inetLocalMailRecipient
mailuserstatus: active
inetuserstatus: active
datasource: NDA 4.5 Delegated Administrator
nsdacapability: mailListCreate
userpassword: {crypt}RI6GKwuXEifxA
uid: test
givenname: test
sn: resend

cn: test resend
preferredlanguage: en
maildeliveryoption: program
mailprogramdeliveryinfo: testr
mailhost: maxima.liston.nu
mail: test@sonny.org

dn: uid=test-archive,ou=people,o=sonny.org,o=isp
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
objectclass: inetUser
objectclass: ipUser
objectclass: nsManagedPerson
objectclass: userPresenceProfile
objectclass: inetMailUser
objectclass: inetLocalMailRecipient
mailuserstatus: active
inetuserstatus: active
datasource: NDA 4.5 Delegated Administrator
nsdacapability: mailListCreate
userpassword: {crypt}tcCW8XBsV.AB.
uid: test-archive
givenname: test
sn: archive
cn: test archive
preferredlanguage: en
maildeliveryoption: program
mailprogramdeliveryinfo: testa
mailhost: maxima.liston.nu
mail: test-archiver@sonny.org

```
dn: uid=test-digest,ou=people,o=sonny.org,o=isp
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
objectclass: inetUser
objectclass: ipUser
objectclass: nsManagedPerson
objectclass: userPresenceProfile
objectclass: inetMailUser
objectclass: inetLocalMailRecipient
mailuserstatus: active
inetuserstatus: active
datasource: NDA 4.5 Delegated Administrator
nsdacapability: mailListCreate

userpassword: {crypt}qvSQMcsoYwR5Q
uid: test-digest
givenname: test
sn: digest
cn: test digest
preferredlanguage: en
maildeliveryoption: program
mailprogramdeliveryinfo: testd
mailhost: maxima.liston.nu
mail: test-digest@sonny.org

dn: uid=test-request,ou=people,o=sonny.org,o=isp
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
objectclass: inetUser
objectclass: ipUser
objectclass: nsManagedPerson
objectclass: userPresenceProfile
objectclass: inetMailUser
objectclass: inetLocalMailRecipient
mailuserstatus: active
inetuserstatus: active
datasource: NDA 4.5 Delegated Administrator
nsdacapability: mailListCreate
userpassword: {crypt}RIMZpTZBydwqw
uid: test-request
givenname: test
sn: request
cn: test request
preferredlanguage: en
maildeliveryoption: program
mailprogramdeliveryinfo: testq
mailhost: maxima.liston.nu
mail: test-request@sonny.org
```

No end user should ever write directly to the *-outgoing aliases, or to the *-test or *-archive addresses. Subscriptions and removals are handled at the majordomo addresses, but to activate the archive or digest for the list, just add their address(es) as members of the "test" mailing list (/opt/majordom/data/lists/test) or as normal email commands to majordomo.

4. **Refresh the MTA, and perhaps even run** stop-msg **and** start-msg **for good measure.**

Glossary

access control information

A single item of information from an access control list within an LDAP directory.

access control list

A set of data associated with a directory that defines the permissions that users and groups have for accessing it.

ACI

See **[access control information]**

ACL

See **[access control list]**

API

applications programming interface.

APOP

See **[Authenticated Post Office Protocol]**

applications service provider

An application service provider is a company that offers individuals or enterprises access over the Internet to applications and related services that would otherwise have to be located in their own personal or enterprise computers.

ASP

See **[applications service provider]**

Authenticated Post Office Protocol

Similar to the Post Office Protocol, but instead of using a plaintext password for authentication, it uses an encoding of the password together with a challenge string.

CLI

See **[command-line interface]**

cn

LDAP alias for common name.

command-line interface

Text driven interface as opposed to a GUI; can easily be used to script or automate repetitive processes.

comment character

A character that, when placed at the beginning of a line, turns the line into a nonexecutable comment.

CSV

comma separated variable-length file.

DC Tree

Domain Component tree. A directory information tree that mirrors the DNS network syntax. An example of a distinguished name in a DC Tree is: cn=billbob,dc=bridge,dc=net,o=internet.

DHCP

Dynamic Host Configuration Protocol.

DMZ

demilitarized zone.

DNLC

directory name lookup cache.

DNS

See **[Domain Name Service]**

domain name

The unique name that identifies an Internet website. Domain names have two or more parts, separated by periods (dots).

Domain Name Service

A distributed name resolution software that allows computers to locate other computers on a network or the Internet by domain name. The system associates standard IP addresses with host names (such as www.siroe.com). Machines normally get this information from a DNS server. DNS servers provide a distributed, replicated, data query service for translating hostnames.

DOS

denial of service.

DSN

delivery status notification.

elm

Originally an acronym to refer to ELectronic Mail, but it is also a program used to read mail on terminals using a text interface (that is, not a GUI).

EOL

end of life.

ESMTP

See **[Extended Simple Mail Transfer Protocol]**

Extended Simple Mail Transfer Protocol

An Internet message transport protocol. ESMTP adds optional commands to the SMTP command set for enhanced functionality, including the ability for ESMTP servers to discover which commands are implemented by the remote site.

FQN

fully qualified host name.

FTP

File Transfer Protocol.

GUI

graphical user interface.

HRS

human resource system.

HTTP

See **[HyperText Transfer Protocol]**

HTTPS

Hypertext Transfer Protocol, Secure.

HyperText Transfer Protocol

A standard protocol that allows the transfer of hypertext documents over the Web. The iPlanet Messaging Server provides an HTTP service to support web-based email.

See also **[Messenger Express]**

IDA

iPlanet delegated administrator.

IDC

internet data center.

IETF

Internet Engineering Task Force.

IM

instant messaging.

IMAP

See **[Internet Message Access Protocol Version 4]**

IMAP4

See **[Internet Message Access Protocol Version 4]**

IMP

input message processing.

Internet Message Access Protocol Version 4

A standard protocol that allows users to be disconnected from the main messaging system and still be able to process their mail. The IMAP specification allows for Administrative control for these disconnected users and for the synchronization of the users' message store once they reconnect to the messaging system.

Internet Protocol

The basic network-layer protocol on which the Internet and intranets are based.

Internet Service Provider

A company that provides Internet services to its customers including email, electronic calendaring, access to the world wide web, and web hosting.

IP

See **[Internet Protocol]**

ISP

See **[Internet Service Provider]**

JASS

Jumpstart Architecture and Security Scripts.

LDAP

See **[Lightweight Directory Access Protocol]**

LDIF

See **[Lightweight Data Interchange Format]**

Lightweight Data Interchange Format

The format used to represent Directory Server entries in text form.

Lightweight Directory Access Protocol

Directory service protocol designed to run over TCP/IP and across multiple platforms. A simplification of the X.500 Directory Access Protocol (DAP) that allows a single point of management for storage, retrieval, and distribution of information, including user profiles, mail lists, and configuration data across iPlanet servers. The iPlanet Directory Server uses the LDAP protocol.

LMTP

See **[Local Mail Transfer Protocol]**

Local Mail Transfer Protocol

A derivative of the SMTP and E/SMTP protocols that is nearly identical. LMTP is designed to provide a status reply per message recipient versus SMTP's single reply code per message transaction.

Mail eXchanger record

A mail eXchanger record is an entry in your DNS table that controls where email is sent for a particular or given domain name.

MEM

messenger express multiplexer.

message-handling system

A group of connected MTAs, their user agents, and message stores.

Message Transfer Agent

A specialized program for routing and delivering messages. MTAs work together to transfer messages and deliver them to the intended recipient. The MTA determines whether a message is delivered to the local message store or routed to another MTA for remote delivery.

messaging multiplexer proxy

A specialized messaging server that acts as a single point of connection to multiple messaging servers.

Messaging Server administrator

The administrator whose privileges include installation and administration of an iPlanet Messaging Server instance.

Messenger Express

A mail client that enables users to access their mailboxes through a browser-based (HTTP) interface. Messages, folders, and other mailbox information are displayed in HTML in a browser window.

See also **[web mail]**

MHS

See also **[message-handling system]**
See also **[Simple Network Management Protocol]**

MIB

management information base.

MIME

See **[Multipurpose Internet Mail Extension]**

MMP

See **[messaging multiplexer proxy]**

MTA

See **[Message Transfer Agent]**

MTA configuration file

The file (imta.cnf) that contains all channel definitions for the Messaging Server plus the rewrite rules that determine how addresses are rewritten for routing.

MTBF

mean time between failures.

MTTR

mean time to repair (or recover).

Multipurpose Internet Mail Extension

A protocol you can use to include multimedia in email messages by appending the multimedia file in the message. This protocol that allows for the transmission of data in many forms, such as audio, binary, or video.

See also **[SMIME]**

MX

See **[Mail eXchanger record]**

NDA

Netscape delegated administrator.

NFS

Network File Server or Network File System.

NIS

Network Information Service.

NMS

Netscape Messaging Server.

PAB

personal address book.

password authentication

Verifies that the user's password is valid.

PDA

personal digital assistant.

PGP

pretty good protection.

PIN

personal identification number.

Pine

Program for Internet News and Email.

See also **[elm]**

Plaintext

Refers to a method for transmitting data. The definition depends on the context. For example, with SSL plaintext passwords are encrypted and are therefore not sent as cleartext. With SASL, plaintext passwords are hashed, and only a hash of the password is sent as text.

See also **[Secure Sockets Layer]**
See also **[Simple Authentication and Security Layer]**

plaintext authentication
See **[password authentication]**

POP
See **[Post Office Protocol Version 3]**

POP3
See **[Post Office Protocol Version 3]**

Post Office Protocol Version 3

A protocol that provides a standard delivery method and that does not require the message transfer agent to have access to the user's mail folders. Not requiring access is an advantage in a networked environment.

PS

PostScript.

QoS

Quality of Service.

RDNS

reverse DNS.

SASL
See **[Simple Authentication and Security Layer]**

SDLC

Systems development life cycle.

SDN

See **[Software Delivery Network]**

Secure Sockets Layer

The Secure Sockets Layer is a commonly used protocol for managing the security of a message transmission on the Internet. SSL has recently been succeeded by Transport Layer Security (TLS) which is based on SSL. SSL uses a program layer located between the HTTP and TCP layers.

Short Messaging Service

A service for sending messages of up to 160 characters (224 characters if using a 5-bit mode) to mobile phones and other devices that use Global System for Mobile communication. Due to the length restriction, it is advantageous to strip off attachments and certain header information from normal email when being delivered to an SMS device.

Simple Authentication and Security Layer

A means for controlling the mechanisms by which POP, IMAP or SMTP clients identify themselves to the server. iPlanet Messaging Server support for SMTP SASL use complies with RFC 2554 (ESMTP AUTH). SASL is defined in RFC 2222.

Simple Mail Transfer Protocol

The email protocol most commonly used by the Internet and the protocol supported by the iPlanet Messaging Server. Defined in RFC 821, with associated message format descriptions in RFC 822.

Simple Network Management Protocol

The protocol governing network management and the monitoring of network devices and their functions. It is not necessarily limited to TCP/IP networks. SNMP is described formally in the Internet Engineering Task Force (IETF) 1157 and in a number of other related RFCs.

SIMS

Sun Internet Mail Server.

SIP

Simple Internet Protocol.

SIS

student information system.

SMIME

Secure Multipurpose Internet Mail Extension.

SMS

See **[Short Messaging Service]**

SMTP

See **[Simple Mail Transfer Protocol]**

SNMP

See **[Simple Network Management Protocol]**

Software Delivery Network

Software Delivery Network, sometimes referred to as Service Delivery Network, is a term used by Sun to describe and define an infrastructure designed to provide a foundation for scalable network-based services, such as the Sun ONE Messaging Server and Sun ONE Directory Server, while meeting demands for reliability and performance.

SPI

stateful packet inspection.

SPN

Service provider networks.

SSL

See **[Secure Sockets Layer]**

SSO

single sign on.

TCO

total cost of ownership.

TCP

See **[Transmission Control Protocol]**

TCP/IP

See **[Transmission Control Protocol/Internet Protocol]**

TLS

See **[Transport Layer security]**

Transmission Control Protocol

The basic transport protocol in the Internet protocol suite that provides reliable, connection-oriented stream service between two hosts—the Transport Layer Protocol, Internet Protocol, and the Network Layer Protocol.

Transmission Control Protocol/Internet Protocol

The name given to the collection of network protocols used by the Internet protocol suite. The name refers to the two primary network protocols of the suite—the Internet Protocol and the Network Layer Protocol.

Transport Layer security

The standardized form of SSL.

See also **[Secure Sockets Layer]**

UBE

See **[unsolicited bulk email]**

UCE

unsolicited commercial email .

See also **[unsolicited bulk email]**

UDDI

universal description, discovery, and integration.

unsolicited bulk email

Unrequested and unwanted email, sent from bulk distributors, usually for commercial purposes.

user agent

The client component, such as Netscape Communicator, that allows users to create, send, and receive mail messages.

VPN

virtual private network.

WAN

wide area network.

web mail

A generic term for browser-based email services. A browser-based client—known as a *thin* client because more processing is done on the server—accesses mail that is always stored on a server.

See also **[Messenger Express]**

Bibliography

Abitz, Paul and Liu, Cricket, *DNS and BIND*, *4th Ed*ition, April 2001, O'Reilly.

Author unknown, "Sun ONE Messaging Server Version 5.2 - A Technical Whitepaper," Sun Microsystems.

Bialaski, Tom, "Understanding Solaris 9 Operating Environment Directory Services," Sun Blueprints, December 2002.

Bialaski, Tom, "Running Multiple Solaris Operating Environment Naming Services on a Client," Sun BluePrints, May 2001.

Bialaski, Tom, "Automating LDAP Client Installations," July 2001, Sun BluePrints.

Carter, Gerald, *LDAP System Administration and Managing IMAP*, March 2003, O'Reilly.

Deeths, David, and Howard, John S., *Configuring Boot Disks*, December 2001, Prentice Hall.

Elling, Richard, *Operating Environment: Solaris 8 Installation and Boot Disk Layout*, March 2000, Prentice Hall.

Johnson, Kevin, *Internet Email Protocols: A Developer's Guide*, 2000, Addison-Wesley Publishing Co.

Liu, Cricket, *DNS & BIND Cookbook*, October 2002, O'Reilly.

Lopez, Steve, "Solaris Operating Environment LDAP Capacity Planning and Performance Tuning," May 2002, Sun BluePrints

Twomey, John, "iPlanet Messaging Server Migration from UNIX® Sendmail," July 2002, Sun Microsystems.

Venditti, Nicola, "Writing an Authentication Plug-in for a Sun ONE Directory Server," March 2003, Sun BluePrints.

Weber, Stefan, "Securing LDAP Through TLS/SSL--A Cookbook," June 2002, Sun BluePrints.

Winsor, Janice, *Solaris System Administrator's Guide*, *4th Edition*, May 2003, Prentice Hall.

Wood, David, *Programming Internet EMail*, September 2000, O'Reilly.

[SYMBOL] [A] [B] [C] [D] [E] [F] [H] [I] [J] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

/etc/system tuning
tuning:/etc/system

[Team LiB]

access control
  instruction
  lists
ACI
additional attributes
administration ports
administration web interface
alias
aliases file
alternate address
antispam 2nd 3rd
applications programming interfaces
applications service provider
architecting, high availability differences
architecture
  categories
  high availability
  messaging
  secure
  secure with failover
  single layer
  typical
architecture:high availability
  high availability:architectures
authentication cache size
  DOS
  tuning
authentication cache TTL, tuning
  tuning:authentication cache TTL

b1
Big Brother
BMC Patrol

[Team LiB]

[SYMBOL] [A] [B] [C] [D] [E] [F] [H] [I] [J] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

calendar, web-based
  web-based calendar
checklists, periodic maintenance
checks:
  weekly checks
checks:annual
  annual checks
checks:daily
  daily checks
checks:monthly
  monthly checks
checks:quarterly
  quarterly checks
clients, popular
comma separated variable-length file
common name
configuration
  current settings
  Message Transfer Agent
  MTA directory
  MTA files
connectivity, network
console, administration
conversion channel
  append disclaimer
conversion channel:message processing
  message processing, conversion channel
conversion utility, PS to Acrobat
CONVERSIONS file
CSV file

daemons, multiple
data feeds
database temporary directory, tuning
delegated administrator
  creating and administrating aliases
  GUI installation
  server installing
delegated administrator, installing
delivery status notifications, tuning
  tuning:delivery status notification
demilitarized zone 2nd
denial of service
denial of service, prevention tuning
DHCP
direct LDAP
  manipulation
direct LDAP:lookups, tuning
  tuning:direct LDAP lookups
direct lookup, LDAP
directory, high availability
  high availability:directory
dirsync
disclaimer, adding a
disk layout
dispatcher, tuning
  tuning:dispatcher
domain name
Domain Name Service
Dynamic Host Configuration Protocol 2nd

[SYMBOL] [A] [B] [C] [D] [**E**] [F] [H] [I] [J] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

electronic messaging
    messaging:electronic
elm
ELM
elm
email system, overall design
enterprise web server, installing
Extended Simple Mail Transfer Protocol

[SYMBOL] [A] [B] [C] [D] [E] [F] [H] [I] [J] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

failover software
fully qualified host name

[Team LiB]

high availability
- best practices and caveats
- conclusions
- configuration
- installation procedure and notes
- other architectures

host name, fully qualified
hosts, critical

[Team LiB]

identifier and rights pairs, shared folders
  shared folders:identifier and rights pairs
IMAP 2nd
ims-ms channel, tuning
  tuning:ims-ms channel
ims_master processes, tuning
  tuning:ims_master processes
imta_tailor file, tuning
  tuning:imta_tailor file
installation
  script
  simple
  values
installation:software
  software:installation and configuration
instant messaging 2nd 3rd 4th
Internet data center
Internet Engineering Task Force 2nd
Internet Service Provider 2nd
issues
  regulatory

Java
JavaScript
job controller, tuning
    tuning:job controller
job_limit:tuning
    tuning:job_limit
JumpStart
JumpStart Architecture and Security Scripts

[Team LiB]

layers, security
  security:layers
LDAP: hosts, tuning
  tuning:LDAP hosts
LDAP:timeout, tuning
  tuning:LDAP timeout
Lightweight Data Interchange Format
Lightweight Directory Access Protocol 2nd
load balancing, network
Local Mail Transfer Protocol
log file location
login screen, customizing
logos, changing and adding

mail eXchanger record
mail gateway
mailstore
mailstore, high availability
  high availability:mailstore
management information bases
MAPPINGS file
mappings, changing
master directory server
  installing
  preparing for messaging
MAX_CLIENT_THREADS, tuning
  tuning:MAX_CLIENT_THREADS
MAX_INTERNAL_BLOCKS, tuning
  tuning:MAX_INTERNAL_BLOCKS
maximum physical I/O size, tuning
  tuning:maximum physical I/O size
message
  average size
message dequeue
  dequeue message
message transfer agent
Message Transfer Agent, configuration
message URL http
  //bb4.com/
  //devel-home.kde.org/~kmail/index.html 2nd
  //docs.sun.com
  //docs.sun.com/db/doc/806-4078
  //docs.sun.com/db/prod/sunone
  //docs.sun.com/source/816-5606-10/password.htm#1085603
  //docs.sun.com/source/816-6009-10/channel.htm#43150
  //docs.sun.com/source/816-6009-10/mtacncpt.htm#22760
  //docs.sun.com/source/816-6009-10/snmp.htm#23526
  //docs.sun.com/source/816-6009-10/trblesho.htm#13833
  //docs.sun.com/source/816-6010-10/index.html
  //docs.sun.com/source/816-6014-10
  //docs.sun.com/source/816-6014-10/ha.htm#11284
  //docs.sun.com/source/816-6020-10/ms_cmds.htm#15794
  //docs.sun.com/source/816-6092-10/index.html
  //docs.sun.com/source/816-6829-10/index.html
  //email.about.com/cs/openpgpsoftware/
  //hostutopia.com/support/s058.html
  //ims.balius.com/ 2nd
  //sunsolve.sun.com
  //sunsolve.sun.com/pub-cgi/secBulletin.pl?mode=latest
  //sunsolve.sun.com/pub-cgi/show.pl?target=patches/patch-license&nav=pub-patches
  //www.bc.edu/bc_org/tvp/email/helpers.shtml
  //www.bmc.com/
  //www.cert.org/
  //www.deadcat.net/
  //www.halcyoninc.com/downloads/home.html
  //www.haltabuse.org/pgp/index.shtml
  //www.ietf.org/rfc/rfc2086.txt?number=2086
  //www.interguru.com/mailconv.htm 2nd

//www.ipswitch.com/Products/WhatsUp/index.html
//www.isi.edu/in-notes/iana/assignments/media-types/media-types
//www.orcaware.com/orca/
//www.oreilly.com/catalog/dns4/
//www.oreilly.com/catalog/dnsbindckbk/
//www.oreilly.com/catalog/ldapsa/ 2nd
//www.pgpi.org/doc/overview/
//www.setoolkit.com/
//www.si.edu/resource/tours/comphist/ma1.html
//www.sun.com/blueprints
//www.sun.com/newsletters/
//www.sun.com/products/architectures-platforms/refarch/specs.html#g1_5.1
//www.sun.com/service/contacting
//www.sun.com/service/sunps/architect/delivery/
//www.sun.com/service/sunps/systemsandnetworkmanagement/bmcpatrol/
//www.sun.com/smi/Press/2003-04/sunflash.20030414.1.html
//www.sun.com/solutions/blueprints/browsesubject.html#jumpstart
//www.sun.com/solutions/blueprints/browsesubject.html#nds
//www.sun.com/solutions/blueprints/browsesubject.html#security 2nd
//www.ximian.org
//wwws.sun.com 2nd
//wwws.sun.com/software/products/provisioning_server/
//wwws.sun.com/software/security/jass/
//wwws.sun.com/software/solaris/sunmanagementcenter/index.html
//wwws.sun.com/software/solaris/webstartflash/
//wwws.sun.com/software/whitepapers/wp-solarisinst/solaris_installation_deployment.pdf
message URL ttp
//www.newplanetsoftware.com/arrow/
messages
virus scanning
messages, number of
messaging
devices
high availability deployment
high availability, differences in planning
implementations
in a box 2nd
managing and preventive maintenance
protocols
security
strategy 2nd
system testing
system verification
unified
messaging multiplexer proxy
messaging server
current configuration
installing
messaging services
beyond the basics
messaging services, overview
messaging system
basic parts
messaging, web
messaging:high availability deployment
high availability:messaging deployment
messenger express, customizing
customizing, messenger express
migration

aliases and system-wide mailing lists
basic steps
export and import
messages and folders
password importance
personal address books, lists, and bookmarks
sendmail
sendmail mailbox content
sendmail mailing lists
sendmail personal address books
sendmail user information
specialized software
user information
utilities, other
utility
MIME messages, parts
monitoring
SNMP
MTA 2nd
basics
history
possibilities

naming services
ncsize, tuning
  tuning:ncsize
Netscape Messaging Server 2nd
network connectivity, issues
Network Information Service
notices, tuning
  tuning:notices 2nd
number of processes, tuning and limitation

options tab
  adding and removing options
  adding options
  removing options
Orca
over-quota limits, configuring
overview
  messaging services

[Team LiB]

partitioning
passwords, options for handling
personal address book
personal digital assistants 2nd
PGP/MIME
Pine 2nd
port numbers
portal
Post Office Protocol
postmaster
  user account, creating
practices, good computing
pretty good protection (encryption)
process settings
production and a non-production environment, differences
production environment
production versus non-production
project Orion
protocol status
provisioning
  administration console
  authoritative sources
  data feeds
  delegated administrator for messaging
  issues
  Lightweight Directory Access Protocol
  methods
  sample script
  test user generation script
  user ID
  web
provisioning:command-line interface
  command-line interface:provisioning
provisioning:script
  sample provisioning script
proxy servers
  benefits

[SYMBOL] [A] [B] [C] [D] [E] [F] [H] [I] [J] [L] [M] [N] [O] [P] [**Q**] [R] [S] [T] [U] [V] [W]

Quality of Service

[SYMBOL] [A] [B] [C] [D] [E] [F] [H] [I] [J] [L] [M] [N] [O] [P] [Q] [**R**] [S] [T] [U] [V] [W]

regulatory issues
return errors, customizing
reverse database, tuning
    tuning:reverse database
reverse DNS

Secure Multipurpose Internet Mail Extensions
Secure Socket Layer
security
  antivirus and antispam
  digital signing
  directory
  enabling SSL
  message contents
  message store
  messaging server software points
  messaging software protocols
  MTA
  network layer
  non-standard ports
  PGP signing
  reverse DNS lookup
  search limits
  SMTP
  Solaris OE
  system
security:SMTP
  SMTP:security
sendmail, disabling
servers, proxy
service provider networks
services, directory
shared folders
  description
  limitations
  permission
shared folders:configuration
  configuration:shared folders
shared folders:direct deliver
  direct deliver:shared folders
shared folders:IMAP client
  IMAP:client, shared folders
shared folders:Mulberry
  Mulberry, shared folders
shared folders:Netscape Messenger
  Netscape Messenger, shared folders
shared folders:outlook express
  outlook express, shared folders
Short Messaging Service
Simple Authentication and Security Layer
Simple Internet Protocol 2nd
Simple Mail Transfer Protocol
simple messaging installation with MTA
  benefits
  drawbacks
Simple Network Management Protocol
SIMS 2nd
single layer architecture
  benefits
  drawbacks

single sign on 2nd
enabling
SMTP
relays
software
download location
software delivery network
concept
Solaris OE
basic installation
spam
standards
open
supported for shared folders
stateful packet inspection
store database cache size, tuning
tuning:store database cache size
Sun Internet Mail Server
Sun Management Center 2nd
system
startup
status
system security points
systems development life cycle

TCP/IP, tuning
tuning:TCP/IP 2nd
tcp_local_option files, tuning
tuning
tcp_local_option files
test accounts, creating
threaddepth, tuning
tuning:threaddepth
tools, alternative
total cost of ownership
Transmission Control Protocol/Internet Protocol
tuning:MMP
MMP, tuning
tuning:MTA
MTA:tuning
tuning:option.dat
option.dat, tuning
tuning:postmaster mail
postmaster:mail, tuning
tuning:Solaris OE
Solaris OE:tuning
typical architecture, benefits

unified messaging
unique user ID
UNIX user account and group, creating
unsolicited bulk email
user and group bind, tuning
  tuning:user and group bind
user folder, direct deliver
  direct deliver:user folder
user ID
  data file sample
  email address
user population turnover
user store

[Team LiB]

VERITAS file system, tuning
  tuning:VERITAS file system
VERITAS Volume Manager, tuning
  tuning, VERITAS Volume Manager
virtual private network 2nd
virus scanning
virus scanning messages

warning Email, configuring
web mail permissions
web mail spool directory, tuning
  tuning:web mail spool directory
web service
welcome email, setting initial

[Team LiB]

Brought to You by