Administrator's Guide

iPlanet Messaging Server

Release 5.1

May 2001

Copyright © 2001 Sun Microsystems, Inc. Some preexisting portions Copyright © 2000 Netscape Communications Corporation. All rights reserved.

Sun, Sun Microsystems, the Sun logo, iPlanet, and the iPlanet logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. Netscape and the Netscape N logo are registered trademarks of Netscape Communications Corporation in the U.S. and other countries. Other Netscape logos, product names, and service names are also trademarks of Netscape Communications Corporation, which may be registered in other countries.

Federal Acquisitions: Commercial Software—Government Users Subject to Standard License Terms and Conditions

The product described in this document is distributed under licenses restricting its use, copying, distribution, and decompilation. No part of the product or this document may be reproduced in any form by any means without prior written authorization of the Sun-Netscape Alliance and its licensors, if any.

THIS DOCUMENTATION IS PROVIDED “AS IS” AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright © 2001 Sun Microsystems, Inc. Pour certaines parties préexistantes, Copyright © 2000 Netscape Communication Corp. Tous droits réservés.

Sun, Sun Microsystems, et the Sun logo, iPlanet, et the iPlanet logo sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et d’autre pays. Netscape et the Netscape N logo sont des marques déposées de Netscape Communications Corporation aux Etats-Unis et d’autre pays. Les autres logos, les noms de produit, et les noms de service de Netscape sont des marques déposées de Netscape Communications Corporation dans certains autres pays.

Le produit décrit dans ce document est distribué selon des conditions de licence qui en restreignent l'utilisation, la copie, la distribution et la décompilation. Aucune partie de ce produit ni de ce document ne peut être reproduite sous quelque forme ou par quelque moyen que ce soit sans l’autorisation écrite préalable de l’Alliance Sun-Netscape et, le cas échéant, de ses bailleurs de licence.

CETTE DOCUMENTATION EST FOURNIE “EN L'ÉTAT”, ET TOUTES CONDITIONS EXPRESSES OU IMPLICITES, TOUTES REPRÉSENTATIONS ET TOUTES GARANTIES, Y COMPRIS TOUTE GARANTIE IMPLICITE D'APTITUDE À LA VENTE, OU À UN BUT PARTICULIER OU DE NON CONTREFAÇON SONT EXCLUES, EXCEPTÉ DANS LA MESURE OÙ DE TELLES EXCLUSIONS SERAIENT CONTRAIRES À LA LOI.

Contents

About This Guide **13**
Who Should Read This Book 13
What You Need to Know 14
How This Book is Organized 14
Document Conventions 16
Monospaced Font 16
Bold Monospaced Font 16
Italicized Font 17
Square or Straight Brackets 17
Command Line Prompts 17
Where to Find Related Information 18
Where to Find This Book Online 18

Chapter 1 Introduction **19**
Support for Standard Protocols 20
Support for Hosted Domains 20
Support for User Provisioning 20
Support for Unified Messaging 21
Support for Webmail 21
Powerful Security and Access Control 21
Convenient User Interfaces 22
Post-Installation Directory and File Organization 23

Chapter 2 Configuring General Messaging Capabilities **27**
Viewing Basic Server Information 28
Starting and Stopping Services 28
Starting and Stopping Services in an HA Environment 28
Starting and Stopping Services in a non-HA Environment 29
Configuring a Greeting Message 32
Configuring Languages for Auto-Reply Messages 32

Choosing a User-Preferred Language 33
Configuring a Server Site Language 34
Enabling Single Sign-On (SSO) 35
Messenger Express SSO Configuration Parameters 35
Messenger Express and Delegated Administrator for Messaging 37
Step 1a. Create a Proxy User Account 38
Step 1b. Create an ACI for Proxy Authentication 38
Step 2a. Add the Proxy User Credentials to the resource.properties File 39
Step 2b. Add the Single Sign-On Cookie Information 39
Step 2c. Add the Participating Servers Verification URL 40
Step 3. Restart the Enterprise Server 40
Customizing Directory Lookups 40
Encryption Settings 43

Chapter 3 Managing Mail Users and Mailing Lists 45
Introduction 45
Managing Mail Users 47
Accessing Mail Users 47
Creating a New User 47
Accessing an Existing User 48
Specifying User Email Addresses 48
Configuring Delivery Options 50
Specifying POP/IMAP Delivery 51
Specifying Program Delivery 51
Specifying UNIX Delivery 52
Specifying Forwarding Addresses 52
Configuring Auto-Reply Settings 53
Configuring Authorized Services 54
Managing Mailing Lists 55
Accessing Mailing Lists 55
Creating a New Group 55
Accessing an Existing Group 56
Specifying Mailing List Settings 57
Specifying List Members 59
Defining Dynamic Membership Criteria 60
Adding Mailing-List Members 61
Defining Message-Posting Restrictions 62
Defining Moderators 63

Chapter 4 Configuring POP, IMAP, and HTTP Services 65
General Configuration 66
Enabling and Disabling Services 66

Specifying Port Numbers 66
Ports for Encrypted Communications 67
IMAP Over SSL 67
HTTP Over SSL 67
Service Banner 67
Login Requirements 68
Password-Based Login 68
Certificate-Based Login 69
Performance Parameters 69
Number of Processes 70
Number of Connections per Process 70
Number of Threads per Process 71
Dropping Idle Connections 72
Logging Out HTTP Clients 72
Client Access Controls 72
Configuring POP Services 73
Configuring IMAP Services 74
Configuring HTTP Services 76

Chapter 5 Messaging Multiplexor 81
About Messaging Multiplexor 81
Multiplexor Benefits 82
How Multiplexor Works 84
Encryption (SSL) Option 85
Certificate-Based Client Authentication 85
User Pre-Authentication 86
Virtual Domains 86
Multiple Multiplexor Instances 87
Configuring Multiplexor 89
Starting Multiplexor 90
UNIX Systems 90
Windows NT Systems 90
A Sample Topology 91
IMAP Configuration Example 92
POP Configuration Example 94

Chapter 6 About MTA Services and Configuration 95
The Message Transfer Agent (MTA) 96
Channels 96
Master and Slave Programs 97
Channel Message Queues 99
Rewrite Rules 99

The Job Controller 100
The Dispatcher 101
Creation and Expiration of Server Processes 101
Controlling the Dispatcher 102
The MTA Configuration File 103
Other MTA Configuration Files 105
Autoreply Option File 106
Alias File 106
TCP/IP Channel Option Files 106
Conversion File 107
Dirsync Option File 107
Dispatcher Configuration File 107
Mapping File 108
Option File 109
Tailor File 109
Job Controller File 110
Examples of Use 111
Aliases 113
The Alias Database 114
The Alias File 114
Including Other Files in the Alias File 115
Command Line Utilities 115
The MTA Directory Cache 116
Synchronization Configuration Parameters 117
SMTP Security and Access Control 119
Log Files 119

Chapter 7 Configuring Rewrite Rules 121
Rewrite Rule Structure 122
Rewrite Rule Patterns and Tags 124
A Rule to Match Percent Hacks 126
A Rule to Match Bang-Style (UUCP) Addresses 126
A Rule to Match Any Address 127
Tagged Rewrite Rule Sets 127
Rewrite Rule Templates 127
Ordinary Rewriting Templates: A%B@C or A@B 128
Repeated Rewrites Template, A%B 128
Specified Route Rewriting Templates, A@B@C@D or A@B@C 129
Case Sensitivity in Rewrite Rule Templates 129
How the MTA Applies Rewrite Rules to an Address 130
Step 1. Extract the First Host or Domain Specification 131
Step 2. Scan the Rewrite Rules 133
Step 3. Rewrite Address According to Template 134

Step 4. Finish the Rewrite Process ... 134
Rewrite Rule Failure ... 135
Syntax Checks After Rewrite ... 135
Handling Domain Literals ... 135
Template Substitutions and Rewrite Rule Control Sequences ... 136
Username and Subaddress Substitution, $U, $0U, $1U ... 139
Host/Domain and IP Literal Substitutions, $D, $H, $nD, $nH, $L ... 139
Literal Character Substitutions, $$, $%, $@ ... 140
LDAP Query URL Substitutions, $]...[... 140
General Database Substitutions, $(...) ... 141
Apply Specified Mapping, ${...} ... 142
Customer-supplied Routine Substitutions, $[...] ... 142
Single Field Substitutions, $&, $!, $*, $# ... 143
Unique String Substitutions ... 144
Source-Channel-Specific Rewrite Rules ($M, $N) ... 144
Destination-Channel-Specific Rewrite Rules ($C, $Q) ... 145
Direction-and-Location-Specific Rewrite Rules ($B, $E, $F, $R) ... 146
Host-Location-Specific Rewrites ($A, $P, $S, $X) ... 146
Changing the Current Tag Value, $T ... 147
Controlling Error Messages Associated with Rewriting ($?) ... 148
Handling Large Numbers of Rewrite Rules ... 149
Testing Rewrite Rules ... 149
Rewrite Rules Example ... 150

Chapter 8 Configuring Channel Definitions ... 153
Channel Structure ... 154
Predefined Channels ... 156
Configuring SMTP Channels ... 157
SMTP Command and Protocol Support ... 158
Channel Protocol Selection and Line Terminators ... 160
EHLO Command Support ... 160
ETRN Command Support ... 161
VRFY Command Support ... 162
DNS Domain Verification ... 163
Character Set Labeling and Eight-Bit Data ... 163
Protocol Streaming ... 164
TCP/IP Connection and DNS Lookup Support ... 165
TCP/IP Port Number and Interface Address ... 168
Caching for Channel Connection Information ... 169
DNS Lookups ... 169
IDENT Lookups ... 170

TCP/IP MX Record Support 171
Nameserver Lookups 172
Last Resort Host 172
Alternate Channels for Incoming Mail 172
Target Host Choice 173
SMTP Authentication and SASL 174
Transport Layer Security 174
Channel Operation Type 175
Configuring Message Processing and Delivery 175
Delivery of Messages 178
Processing Pools for Channel Execution Jobs 179
Service Job Limits 179
Message Priority Based on Size 181
SMTP Channel Threads 181
Expansion of Multiple Addresses 182
Undeliverable Message Notification Times 183
Configuring Messages Sent to the Postmaster 184
Configuring Channel Options 185
Configuring Channel Defaults 186
Configuring Logging for Channels 187
Configuring Debugging for Channels 187
Setting Up Program Delivery 187
Using the Hold Channel 189
Using the Conversion Channel 189
Selecting Traffic for Conversion Processing 190
Configuration of the Conversion Channel 190
Conversion Control 190
Understanding Conversions 191
Character Set Conversion and Message Reformatting Mapping 192
Character Set Conversion 193
Message Reformatting 194
Service Conversions 198

Chapter 9 Mail Filtering and Access Control 201
PART 1. MAPPING TABLES 201
Controlling Access with Mapping Tables 202
SEND_ACCESS and ORIG_SEND_ACCESS Tables 203
MAIL_ACCESS and ORIG_MAIL_ACCESS Mapping Tables 205
FROM_ACCESS Mapping Table 207
PORT_ACCESS Mapping Table 209
Limiting Specified IP Address Connections to the MTA 211
When Access Controls Are Applied 212
Testing Access Control Mappings 213

Adding SMTP Relaying 214
Allowing SMTP Relaying for External Sites 216
Configuring SMTP Relay Blocking 217
Differentiate Between Internal and External Mail 217
Differentiate Authenticated Users' Mail 219
Prevent Mail Relay 220
Allowing localhost Submissions to the SMTP Port 221
Using DNS Lookups Including RBL Checking for SMTP Relay Blocking 222
Handling Large Numbers of Access Entries 224
Mapping Table Flags 227
PART 2. MAILBOX FILTERS 228
Introduction 228
Creating Per-User Filters 229
Creating Channel-Level Filters 232
Creating MTA-Wide Filters 234
Routing Discarded Messages out The FILTER_DISCARD Channel 235
Debugging User Filters 235

Chapter 10 Managing the Message Store 237
Overview 237
Message Store Directory Layout 239
How the Store Erases Message 242
Specifying Administrator Access to the Store 242
Adding an Administrator 243
Modifying an Administrator Entry 244
Deleting an Administrator Entry 244
About Message Store Quotas 244
User Quotas 245
Domain Quotas and Family Group Quotas 246
Exceptions for Telephony Application Servers 246
Configuring Message Store Quotas 246
Specifying a Default User Quota 247
Enabling Quota Enforcement and Notification 248
Enabling Quota Enforcement 248
Enabling Quota Notification 249
Defining a Quota Warning Message 249
Specifying a Quota Threshold 250
Setting a Grace Period 250
Specifying Aging Policies 251
Configuring Message Store Partitions 254
Adding a Partition 254
Moving Mailboxes to a Different Disk Partition 255
Performing Maintenance and Recovery Procedures 257

Managing Mailboxes 257
The mboxutil Utility 257
The hashdir Utility 260
The readership Utility 260
Monitoring Quota Limits 260
Monitoring Disk Space 261
Using the stored Utility 262
Repairing Mailboxes and the Mailboxes Database 263
Rebuilding Mailboxes 265
Checking and Repairing Mailboxes 266
Removing Orphaned Accounts 266
reconstruct Performance 267
Moving a User's Account 268
Backing Up and Restoring the Message Store 271
Creating a Backup Policy 272
Peak Business Loads 272
Full and Incremental Backups 273
Parallel or Serial Backups 273
Creating Backup Groups 273
Messaging Server Backup and Restore Utilities 274
The imsbackup Utility 275
The imsrestore Utility 275
Considerations for Partial Restore 275
Using Legato Networker 277
Backing Up Data Using Legato Networker 277
Restoring Data Using Legato Networker 279

Chapter 11 Configuring Security and Access Control 281
About Server Security 282
About HTTP Security 283
Configuring Authentication Mechanisms 284
Configuring Access to Plaintext Passwords 285
Transitioning Users 285
User Password Login 286
IMAP, POP, and HTTP Password Login 287
SMTP Password Login 287
Configuring Encryption and Certificate-Based Authentication 288
Obtaining Certificates 290
Managing Internal and External Modules 290
Requesting a Server Certificate 291
Installing the Certificate 291
Installing Certificates of Trusted CAs 292
Managing Certificates and Trusted CAs 293

Creating a Password File ... 293
Enabling SSL and Selecting Ciphers ... 294
About Ciphers ... 294
Setting Up Certificate-Based Login ... 296
Configuring Administrator Access to Messaging Server ... 298
Hierarchy of Delegated Administration ... 298
Providing Access to the Server as a Whole ... 299
Restricting Access to Specific Tasks ... 299
Configuring Client Access to POP, IMAP, and HTTP Services ... 301
How Client Access Filters Work ... 301
Filter Syntax ... 302
Wildcard Names ... 304
Wildcard Patterns ... 304
EXCEPT Operator ... 305
Server-Host Specification ... 305
Client User-Name Specification ... 306
Filter Examples ... 306
Mostly Denying ... 307
Mostly Allowing ... 307
Allowing Only Identified Users ... 307
Denying Access to Spoofed Domains ... 308
Controlling Access to Virtual Domains ... 308
Denying an Individual User ... 309
Creating Access Filters for Services ... 309
Creating Access Filters for HTTP Proxy Authentication ... 310
Configuring Client Access to SMTP Services ... 311

Chapter 12 Logging and Log Analysis ... 313
PART 1: Introduction ... 313
Logged Services ... 314
Analyzing Logs with Third-Party Tools ... 314
PART 2: Service Logs (Message Store and Administration Server) ... 315
Log Characteristics ... 315
Logging Levels ... 315
Categories of Logged Events ... 317
Filename Conventions for Message Store and Administration Logs ... 317
Log-File Directories ... 318
Log File Format ... 319
Defining and Setting Logging Options ... 320
Flexible Logging Architecture ... 320
Planning the Options You Want ... 321
Setting Logging Options ... 322
Searching and Viewing Logs ... 324

Search Parameters 325
Specifying a Search and Viewing Results 326
PART 3: Service Logs (MTA) 326
Enabling MTA Logging 327
Specifying Additional MTA Logging Options 328
MTA Log Entry Format 329
Managing the MTA Log Files 332
Examples of MTA Message Logging 332
SNMP Implementation 349
SNMP Operation in the Messaging Server 350
Configuring SNMP Support for the iPlanet Messaging Server on Solaris 8 351
Monitoring from an SNMP Client 352
Co-existence with Other iPlanet Products on Unix Platforms 353
SNMP Information from the Messaging Server 353
applTable 354
applTable Usage 355
assocTable 355
assocTable Usage 356
mtaTable 356
mtaTable Usage 357
mtaGroupTable 357
mtaGroupTable Usage 359
mtaGroupAssociationTable 359
mtaGroupErrorTable 360
mtaGroupErrorTable Usage 361

Glossary 363

Index 395

About This Guide

This manual explains how to administer and configure iPlanet Messaging Server. iPlanet Messaging Server provides a powerful and flexible cross-platform solution to the email needs of enterprises and messaging hosts of all sizes using open Internet standards.

Topics covered in this chapter include:

- Who Should Read This Book
- What You Need to Know
- How This Book is Organized
- Document Conventions
- Where to Find Related Information
- Where to Find This Book Online

Who Should Read This Book

You should read this book if you are responsible for administering and configuring iPlanet Messaging Server at your site.

What You Need to Know

This book assumes that you are responsible for configuring and administering the Messaging Server software and that you have a general understanding of the following:

- The Internet and the World Wide Web
- iPlanet Administration Server
- iPlanet Directory Server and LDAP
- Netscape Console

How This Book is Organized

This book contains the following chapters and appendix:

- About This Guide (this chapter)
- Chapter 1, “Introduction”

  This chapter provides a high-level overview of iPlanet Messaging Server.
- Chapter 2, “Configuring General Messaging Capabilities”

  This chapter describes the general Messaging Server tasks—such as starting and stopping services and configuring directory access.
- Chapter 3, “Managing Mail Users and Mailing Lists”

  This chapter describes how to use the Console interface to create and manage your users’ mail accounts and mailing lists.
- Chapter 4, “Configuring POP, IMAP, and HTTP Services”

  This chapter describes how to configure your server to support one or more of these services by using the iPlanet Console or by using command-line utilities.
- Chapter 5, “Messaging Multiplexor”

  This chapter provides concepts about iPlanet Messaging Multiplexor, a specialized messaging server that acts as a single point of connection to multiple messaging servers.

- Chapter 6, “About MTA Services and Configuration”

  This chapter provides concepts about configuring MTA services for your server.

- Chapter 7, “Configuring Rewrite Rules”

  This chapter describes how to configure rewrite rules (for address rewriting) in the MTA configuration file, `imta.cnf`.

- Chapter 8, “Configuring Channel Definitions”

  This chapter describes how to configure channel definitions in the MTA configuration file, `imta.cnf`.

- Chapter 9, “Mail Filtering and Access Control”

  This chapter describes how to control access to mail services and how to filter mail using mapping tables and server-side rules (SSR).

- Chapter 10, “Managing the Message Store”

  This chapter describes the message store directory layout, how to configure message store partitions, set up quotas, set up aging policies, and so on.

- Chapter 11, “Configuring Security and Access Control”

  This chapter describes the security and access control features available with iPlanet Messaging Server.

- Chapter 12, “Logging and Log Analysis”

  This appendix describes how to view and configure service logs for the MTA and for the message store and message access services.

- Appendix A, “SNMP Support”

  This chapter describes how to enable SNMP support for the Messaging Server. It also gives an overview of the type of information provided by SNMP.

- Glossary

  The glossary provides definitions for the terms and naming conventions used throughout the iPlanet Messaging Server documentation.

Document Conventions

Monospaced Font

`Monospaced font` is used for any text that appears on the computer screen or text that you should type. It is also used for filenames, distinguished names, functions, and examples.

Bold Monospaced Font

`Bold monospaced font` is used to represent text within a code example that you should type. For example, you might see something like this:

```
./setup
Sun-Netscape Alliance
iPlanet Server Products Installation/Uninstallation
---------------------------------------------------------------------

Welcome to the iPlanet Server Products installation program. This
program will install iPlanet Server Products and the iPlanet Console
on your computer.

It is recommended that you have "root" privilege to install the
software.

Tips for using the installation program:

- Press "Enter" to choose the default and go to the next screen
- Type "Control-B" to go back to the previous screen
- Type "Control-C" to cancel the installation program
- You can enter multiple items using commas to separate them.
For example: 1, 2, 3

Would you like to continue with installation? [Yes]:
```

In this example, **`./setup`** is what you would type from the command line and the rest is what would appear as a result.

Italicized Font

Italicized font is used to represent text that you enter using information that is unique to your installation (for example, variables). It is used for server paths and names and account IDs.

For example, throughout this document you will see path references of the form:

server-root`/msg-`*serverID*`/...`

In these situations, *server-root* represents the directory path in which you install the server, and `msg-`*serverID* represents the server instance you use when you install it. For example, if you install your server in the directory `/usr/iplanet/server5` and use the server instance `tango`, the actual path is:

```
/usr/iplanet/server5/msg-tango/
```

Square or Straight Brackets

Square (or straight) brackets `[]` are used to enclose optional parameters. For example, in this document you will see the usage for the `setup` command described as follows:

```
./setup [options] [argument]
```

It is possible to run the `setup` command by itself as follows to start the Messaging Server installation:

```
./setup
```

However, the presence of `[options]` and `[arguments]` indicate that there are additional optional parameters that may be added to the `setup` command. For example, you could use `setup` command with the `-k` option to keep the installation cache:

```
./setup -k
```

Command Line Prompts

Command line prompts (for example, `%` for a C-Shell, or `$` for a Korn or Bourne shell) are not displayed in the examples. Depending on which operating system environment you are using, you will see a variety of different command line prompts. However, you should enter the command as it appears in the document unless specifically noted otherwise.

Where to Find Related Information

In addition to this guide, iPlanet Messaging Server comes with supplementary information for administrators as well as documentation for end users and developers. Use the following URL to see all the Messaging Server documentation:

`http://docs.iplanet.com/docs/manuals/messaging.html`

Listed below are some of the additional documents that are available:

- iPlanet Messaging Server Installation Guide
- iPlanet Messaging Server Reference Manual
- iPlanet Messaging Server Provisioning Guide
- iPlanet Messaging Server Schema Reference Manual
- iPlanet Delegated Administrator for Messaging Installation and Administration Guide

Where to Find This Book Online

You can find the *iPlanet Messaging Server 5.1 Administrator's Guide* online in PDF and HTML formats. To find this book, use this URL:

`http://docs.iplanet.com/docs/manuals/messaging.html`

Chapter 1

Introduction

iPlanet Messaging Server is a powerful, standards-based Internet messaging server designed for high-capacity, reliable handling of the messaging needs of both enterprises and service providers. The server consists of several modular, independently configurable components that provide support for several standards-based email protocols.

Messaging Server uses a centralized LDAP database for storing information about users, groups, and domains. Some information about server configuration is stored in the LDAP database; some is stored in a set of configuration files.

The Messaging Server product suite provides tools to support user provisioning and server configuration.

This chapter contains the following sections:

- Support for Standard Protocols
- Support for Hosted Domains
- Support for User Provisioning
- Support for Unified Messaging
- Support for Webmail
- Powerful Security and Access Control
- Convenient User Interfaces
- Post-Installation Directory and File Organization

Support for Standard Protocols

- Internet-standard Simple Mail Transfer Protocol (SMTP) service to handle both internal and Internet mail messages.
- Internet Mail Access Protocol (IMAP4) service for mailbox retrieval, supporting thousands of simultaneous users.
- Post Office Protocol (POP3) service for mailbox retrieval, providing complete support for the most widely used Internet mailbox protocol.
- Specialized Hypertext Transfer Protocol (HTTP) service for web-based email; HTTP clients, such as iPlanet Messenger Express, send mail to a specialized HTTP service which transfers requests to the MTA.

Support for Hosted Domains

Messaging Server provides full support for hosted domains—email domains that are outsourced by an ISP. That is, the ISP provides email domain hosting for an organization by operating and maintaining the email services for that organization. A hosted domain shares the same Messaging Server host with other hosted domains. In earlier LDAP-based email systems, a domain was supported by one or more email server hosts. With Messaging Server, many domains can be hosted on a single server. For each hosted domain, there is an LDAP entry that points to the user and group container for the domain.

Support for User Provisioning

Messaging Server uses a centralized LDAP database for storing information about users, groups, and domains. The iPlanet Delegated Administrator for Messaging product provides a Console graphical user interface and a set of command-line utilities for managing the users, groups, and domains within an organization.

For more information about managing users, groups, and domains, see the following documents:

- *Messaging Server Provisioning Guide* - describes how to create domain, user, group, or administrator entries using LDAP.
- *Messaging Server Reference Manual* - describes the Delegated Administrator command line utilities for managing users, groups, and domains.

In addition, the Delegated Administrator Console interface provides online help for its users.

Support for Unified Messaging

iPlanet Messaging Server provides the basis for a complete unified messaging solution: the concept of using a single message store for email, voicemail, fax, and other forms of communication.

Support for Webmail

iPlanet Messaging Server includes Messenger Express, a web-enabled electronic mail program that lets end users access their mailboxes using a browser running on an Internet-connected computer system using HTTP. Messenger Express clients send mail to a specialized web server that is part of iPlanet Messaging Server. The HTTP service then sends the message to the local MTA or to a remote MTA for routing or delivery.

Powerful Security and Access Control

iPlanet Messaging Server provides the following security and access control features:

- Support for password login (to POP, IMAP, HTTP, or SMTP) and certificate-based login
- Support for standard security protocols: Transport Layer Security (TLS), Secure Sockets Layer (SSL) and Simple Authentication and Security Layer (SASL)
- Delegated administration through access-control instructions (ACIs)
- Client access filters to POP, IMAP, and HTTP
- Client access filters to the MTA
- Filtering of unsolicited bulk email using mapping tables and server-side rules

Convenient User Interfaces

Messaging Server consists of several modular, independently configurable components that provide support for email transport and access protocols.

To configure the Message Transfer Agent (MTA), Messaging Server provides a complete set of configuration files stored locally on the server and a set of command-line utilities. To configure the message store and message access services, Messaging Server provides a Console graphical user interface and a complete set of command-line utilities.

For information about how to configure the MTA and configure access to the MTA, see the following chapters in this manual:

- Chapter 6, “About MTA Services and Configuration”
- Chapter 7, “Configuring Rewrite Rules”
- Chapter 8, “Configuring Channel Definitions”
- Chapter 9, “Mail Filtering and Access Control”
- Chapter 11, “Configuring Security and Access Control”

See also the *iPlanet Messaging Server Reference Manual*.

For information about how to configure the message store and access to the store, see the following chapters in this manual:

- Chapter 4, “Configuring POP, IMAP, and HTTP Services”
- Chapter 10, “Managing the Message Store”
- Chapter 11, “Configuring Security and Access Control”

See also the *iPlanet Messaging Server Reference Manual*.

In addition, you’ll want to review the following chapters in this manual:

- Chapter 2, “Configuring General Messaging Capabilities,” describes general Messaging Server tasks—such as starting and stopping services and configuring directory access.
- Chapter 5, “Messaging Multiplexor,” describes the iPlanet Messaging Multiplexor (MMP)—a specialized messaging server that acts as a single point of connection to multiple messaging servers.

Post-Installation Directory and File Organization

After you install iPlanet Messaging Server, its directories and files are arranged in the organization depicted in Table 1-1. The table is not exhaustive; it shows only those directories and files of most interest for typical server administration tasks.

Table 1-1 Post-Installation Directories and Files

Directory	Default Location and Description
server root directory (*serverRoot*)	`/usr/iplanet/server5/` (default location)
	The directory into which all servers of a given server group (that is, all servers managed by a given Administration Server) are installed. This may include other iPlanet servers in addition to Messaging Server.
	This directory also contains the binary executables for starting and stopping the administration server (`start-admin`, `stop-admin`) and for starting the Console (`startconsole`).
Manuals directory `manual`	*serverRoot*`/manual` (required location)
	Contains the documentation installed with the servers.
	`manual/en/admin/` contains Administration Server documentation.
	`manual/en/msg/` contains Messaging Server documentation
	`manual/en/slapd/` contains Directory Server documentation.
installation directory (*installDirectory*)	*serverRoot*`/bin/msg/` (required location)
	This directory contains some of the binary executables of the installed Messaging Server.

Table 1-1 Post-Installation Directories and Files

Directory	Default Location and Description
instance directory (*instanceDirectory*)	*serverRoot* / msg-*instanceName* / (required location)
	instanceName is the name of this instance of Messaging Server, as specified at installation. (Default = host name of server machine)
	This directory contains the configuration files that define a given instance of Messaging Server. Multiple instances of Messaging Server, all using the same binary files, may exist on a given host machine.
	This directory also contains some of the binary executables of the installed Messaging Server, such as `configutil`, start-`msg`, `stop-msg`, and so on.
configuration directory `config`	*instanceDirectory* / config / (required location)
	Contains general configuration files such as `local.conf`, `msg.conf`, `sslpassword.conf`.
	The values in the file `msg.conf` are set at installation time. Messaging Server uses this file for information it needs at startup such as the LDAP host name and port number.
MTA directory `imta`	*instanceDirectory* / imta / (required location)
	Contains several directories related to MTA configuration: `bin`, `config`, `db`, `dl`, `programs`, `queue`, `tmp`.
MTA configuration directory `config`	*instanceDirectory* / imta / config / (required location)
	Contains the MTA configuration files, such as `imta.cnf`, `dispatcher.cnf`, `job_controller.cnf`, `aliases`, `imta_tailor`, and so on.

Table 1-1 Post-Installation Directories and Files

Directory	Default Location and Description
MTA queue directory `queue`	*instanceDirectory* / imta / queue / (required location)
	Contains the message queue subdirectories. There is one subdirectory for each channel queue; for example: `ims-ms`, `tcp_intranet`, `tcp_local`, `autoreply`.
MTA program directory `programs`	*instanceDirectory*`/imta/programs/` (required location)
	Contains the site-supplied executable programs—if any—for processing user mail.
MTA databases directory `db`	*instanceDirectory*`/imta/db/` (required location)
	Contains the databases used by the MTA: `aliasesdb.db`, `domaindb.db`, `profiledb.db`, `reversedb.db`, `ssrdb.db`.
Message store directory `store`	*instanceDirectory* / store (required location)
	Contains the directories related to message store processing: `mboxlist`, `partition`, `user`.
	For more information, see “Message Store Directory Layout,” on page 239.

Chapter 2

Configuring General Messaging Capabilities

This chapter describes the general Messaging Server tasks—such as starting and stopping services and configuring directory access—that you can perform by using Netscape Console (hereafter called Console) or by using command-line utilities. Tasks specific to individual Messaging Server services—such as POP, IMAP, HTTP, and SMTP—are described in subsequent chapters. This chapter contains the following sections:

- Viewing Basic Server Information
- Starting and Stopping Services
- Configuring a Greeting Message
- Configuring Languages for Auto-Reply Messages
- Enabling Single Sign-On (SSO)
- Customizing Directory Lookups
- Encryption Settings

NOTE End-user account information and domain-specific information is managed primarily through the Delegated Administrator for Messaging interface. For more information, see the *Delegated Administrator for Messaging Installation and Administration Guide* and the online help that comes with Delegated Administrator.

Viewing Basic Server Information

You can review some of the basic information about an installed Messaging Server by viewing its Information form in Console.

To display the Information form:

1. In Console, open the Messaging Server whose information you want to view.
2. Select the server's icon in the left pane.
3. Click the Configuration tab in the left pane.
4. Click the Information tab in the right pane, if it is not already frontmost.

   The Information form appears. It displays the server name, server root directory, installation directory, and instance directory.

Starting and Stopping Services

Services are started and stopped differently depending on whether they are installed in an HA environment or note.

Starting and Stopping Services in an HA Environment

While the Messaging Server is running under HA control, you cannot use the normal Messaging Server start, restart, and stop commands to control individual Messaging Server services. Doing so will cause the HA control to think that one or more services have unexpectedly stopped at which point it will either attempt to restart all of Messaging Server or fail it over to another cluster node.

The appropriate start, stop and restart commands are shown in the tables below. Note that there are no Sun Cluster commands to start, restart, or stop a single Messaging Server service (for example, SMTP). Sun Cluster's finest granularity is that of an individual resource. Since Messaging Server is known to Sun Cluster as a resource, `scswitch` commands affect all Messaging Server services as a whole.

Table 2-1 Start, stop, restart in a Sun Cluster 3.0 environment

Action	Individual Resource	Entire Resource Group
Start	`scswitch -e -j <`*resource*`>`	`sscswitch -Z -g <`*resource-group*`>`

Table 2-1 Start, stop, restart in a Sun Cluster 3.0 environment

Action	Individual Resource	Entire Resource Group
Restart	`scswitch -n -j <resource>` `scswitch -e -j <resource>`	`scswitch -R -g <resource-group>`
Stop	`scswitch -n -j <resource>`	`scswitch -F -g <resource-group>`

Table 2-2 Start, stop, restart in a Sun Cluster 2.2 environment

Action	Individual Data Service	All Registered Data Services
Start	`hareg -y <data_service>`	`hareg -Y`
Restart	`hareg -n <data_service>` `hareg -y <data_service>`	`hareg -N` `hareg -Y`
Stop	`hareg -n <data_service`	`hareg -N`

Table 2-3 Start, stop, restart in Veritas 1.1 environment

Action	Individual Resource	Entire Resource Group
Start	`hares -online <resource> -sys <system>`	`hagrp -online <group> -sys <system>`
Restart	`hares -offline <resource> -sys <system>` `hares -online <resource> -sys <system>`	`hagrp -offline <group> -sys <system>` `hagrp -online <group> -sys <system>`
Stop	`hares -online <resource> -sys <system>`	`hagrp -online <group> -sys <system>`

Starting and Stopping Services in a non-HA Environment

You can start and stop services from Console or from the command line.

You only need to run the services that your server actually uses. For example, if you are temporarily using a particular instance of Messaging Server solely as a message transfer agent (MTA), you can turn on the MTA alone. Or, if maintenance, repair, or security needs require shutting down the server, you may be able to turn off just the affected service. (If you never intend to run a particular service, you should disable it instead of just turning it off.)

NOTE You must first enable the POP, IMAP, and HTTP services before starting or stopping them. For more information, see "Enabling and Disabling Services" on page 66.

Important: If a server process crashes, other processes will hang as they wait for locks held by the server process that crashed. Therefore, if any server process crashes, you should stop all processes, then restart all processes. This includes the POP, IMAP, HTTP, and MTA processes, as well as the `stored` (message store) process, and any utilities that modify the message store, such as `mboxutil`, `deliver`, `reconstruct`, `readership`, or `upgrade`.

Console. Console provides a form that allows you to start and stop individual services and view status information about each service.

For each service—IMAP, POP, SMTP, and HTTP—the form displays the service's current state (on or off). If the service is running, the form shows the time at which the service was last started up, and it can also display other status information.

To start up, shut down, or view the status of any messaging services:

1. From Console, open the Messaging Server whose services you want to start or stop.
2. Get to the Services General Configuration form in either of these two ways:
   a. Click the Tasks tab, then click "Start/Stop Services".
   b. Click the Configuration tab and select the Services folder in the left pane. Then click the General tab in the right pane.
3. The Services General Configuration form appears.

   The left column of the Process Control field lists the services supported by the server; the right column gives the basic status of each of the services (ON or OFF, plus—if it is ON—the time it was last started).

4. To view status information about a service that is currently on, select the service in the Process Control field.

   The Service Status field displays status information about the service.

   For POP, IMAP, and HTTP the field shows the last connection time, the total number of connections, the current number of connections, the number of failed connections since the service last started, and the number of failed logins since the service last started.

   The information in this field helps you to understand the load on the server and the reliability of its service, and it can help spotlight attacks against the server's security.

5. To turn a service on, select it in the Process Control field and click Start.

6. To turn a service off, select it in the Process Control field and click Stop.

7. To turn all enabled services on or off simultaneously, click the Start All or Stop All button.

Command Line. You can use the `start-msg` and `stop-msg` commands to start or stop any of the messaging services (`pop`, `imap`, `http`, `smtp`, `store`), as shown in the following example:

server-root`/msg-`*instance*`/start-msg imap`
server-root`/msg-`*instance*`/stop-msg pop`
server-root`/msg-`*instance*`/stop-msg smtp`

NOTE The `start-msg smtp` and `stop-msg smtp` commands start and stop all of the MTA services—not just the SMTP server. If you want more granular control when starting or stopping the MTA services, use the `imsimta start` and `imsimta stop` commands. For more information, see the *Messaging Server Reference Manual*.

Configuring a Greeting Message

Messaging Server allows you to create a greeting message to be sent to each new user.

Console. To create a new-user greeting by using Console:

1. In Console, open the Messaging Server whose new-user greeting you want to configure.
2. Click the Configuration tab. If the server's icon in the left pane is not already highlighted, select it.
3. Click the Miscellaneous tab in the right pane.
4. Create a new-user greeting or make changes, as needed.

   You must format the greeting as an email message, with a header (containing at least a subject line), then a blank line, then the message body.

   When you create a message, specify its language with the drop-down list above the message field. You can create several messages in several languages, if desired. The server attempts to send the correct language version of the message to the new user based on the information described in "Configuring Languages for Auto-Reply Messages."
5. Click Save.

Command Line. To create a new-user greeting by using the command line:

`configutil -o gen.newuserforms -v` *value*

Configuring Languages for Auto-Reply Messages

This section describes how, for notices and messages sent by the server, the server selects the language-specific version to send. It also describes how users specify a preferred language and how you can specify a default server-site language.

Users can create messages for the server to send automatically under certain specified conditions. For example, an "I am on vacation" message as an automatic reply to all incoming mail. When users create messages of this kind, they can specify that the message is written in a particular language. This allows users to create different, language-specific versions of messages that the server is to send.

Users can also specify a preferred language that indicates in which language they wish to receive automatic reply messages—if that language version is available.

The server selects the language-specific version of a message to send according to the following rules:

1. If the user to whom the message is being sent has chosen a preferred language (see “Choosing a User-Preferred Language” on page 33) and a language-specific version of that message exists, the server sends that version of the message. For example, if the user has chosen Japanese, and there is a Japanese version of the message, the Japanese version is sent.
2. If the user has not chosen a preferred language, or has chosen a preferred language but there is no version of the message in that language, the version that matches the default server-site language (see “Configuring a Server Site Language” on page 34) is sent. For example, if the default site language is Spanish and the user has chosen French but there is no French version of the message, the Spanish version is sent.
3. If there is no version of the message that matches either the user’s preferred language or the default site language, but there is an English-language version, the English version is sent. For example, if the default site language is Spanish and the user has chosen German but there are only French and English versions of the message, the English version is sent.
4. If there is only one version of the message, regardless of language preference or site language, that is the version that is sent.

NOTE Although Delegated Administrator top-level administrators can set a default language for a hosted domain, Messaging Server does not use this setting when determining which language version of a message to send.

Choosing a User-Preferred Language

Users can choose a preferred language by using the Delegated Administrator for Messaging interface. Some mail clients also allow users to specify a preferred language. If the preferred language is set using Delegated Administrator, the information is stored in Directory Server.

When the server sends messages to users outside of the server’s administrative domain it does not know what their preferred language is unless it is responding to an incoming message with a preferred language specified in the incoming message’s header. The header fields (`Preferred-Language` or `X-Accept-Language`) are set according to attributes specified in the user’s mail client.

If there are multiple settings for the preferred language—for example, if a user has a preferred language attribute stored in the Directory Server and also has a preferred language specified in their mail client—the server chooses the preferred language in the following order:

1. The `Preferred-Language` header field of the original message
2. The `X-Accept-Language` header field of the original message
3. The preferred language attribute of the sender (if found in the LDAP directory)

Configuring a Server Site Language

You can specify a default site language for your server as follows. The site language will be used to send language-specific versions of messages if no user preferred language is set.

Console. To specify a site language from Console:

1. Open the Messaging Server you want to configure.
2. Click the Configuration tab.
3. In the right pane, click the Miscellaneous tab.
4. From the site language drop-down list, choose the language you wish to use.
5. Click Save.

Command Line. You can also specify a site language at the command line as follows:

`configutil -o gen.sitelanguage -v` *value*

where *value* is one of the local supported languages:

```
af    Afrikaans
ca    Catalan
da    Danish
de    German
en    English
es    Spanish
fi    Finnish
fr    French
ga    Irish
gl    Galician
is    Icelandic
it    Italian
```

```
ja    Japanese
nl    Dutch
no    Norwegian
pt    Portuguese
sv    Swedish
```

Enabling Single Sign-On (SSO)

Single sign-on allows an end user to authenticate once to use multiple applications. For example, a user can log on to Messenger Express then use Delegated Administrator for Messaging without authenticating again.

To enable single sign-on between applications, you must configure each application. This section describes how to enable single sign-on between Messenger Express and Delegated Administrator. See “Messenger Express and Delegated Administrator for Messaging,” on page 37.

Messenger Express SSO Configuration Parameters

You can modify the single sign-on configuration parameters for Messenger Express, shown in Table 2-4, by using the `configutil` command. For more information about `configutil`, see the *Messaging Server Reference Manual*.

Table 2-4 Messenger Express Single Sign-On Parameters

Parameter	Description
`local.webmail.sso.enable`	Enables or disables all single sign-on functionality, including accepting and verifying SSO cookies presented by the client when the login page is fetched, returning an SSO cookie to the client on successful login and responding to requests from other SSO partners to verify its own cookies. If set to any non-zero value, the server performs all SSO functions. If set to zero, the server does not perform any of these SSO functions. The default value is zero.

Table 2-4 Messenger Express Single Sign-On Parameters

Parameter	Description
`local.webmail.sso.prefix`	The string value of this parameter is used as the prefix value when formatting SSO cookies set by the HTTP server. Only SSO cookies with this prefix will be recognized by the server; all other SSO cookies will be ignored. A null value for this parameter effectively disables all SSO functionality on the server. The default value is null.
`local.webmail.sso.id`	The string value of this parameter is used as the application ID value when formatting SSO cookies set by the HTTP server. The default value is null.
`local.webmail.sso.cookiedomain`	The string value of this parameter is used to set the cookie domain value of all SSO cookies set by the HTTP server. The default value is null.
`local.webmail.sso.singlesignoff`	The integer value of this parameter, if set to any non-zero value, clears all SSO cookies on the client with prefix values matching the value configured in local.webmail.sso.prefix when the client logs out. If set to zero, Messenger Express will clear its own SSO cookie when the client logs out. The default value is zero.
`local.sso.`*appid*`.verifyurl`	Sets the verify URL values for peer SSO hosts. *appid* is the application ID of a peer SSO host whose SSO cookies are to be honored. For example, the *appid* for Delegated Administrator is `nda45.` There should be one parameter defined for each trusted peer SSO host. The standard form of the verify URL is: `http://`*nda-host:port*`/VerifySSO?`

So to enable single sign-on for Messenger Express, you would set the configuration parameters as follows (your default domain is eng.siroe.com)

```
configutil -o local.sso.appid.verifyurl -v “http://nda-host:port/verifySSO?”
configutil -o local.webmail.sso.enable -v 1
configutil -o local.webmail.sso.prefix -v ssogrp1
configutil -o local.webmail.sso.id -v msg50
configutil -o local.webmail.sso.cookiedomain -v “.siroe.com”
configutil -o local.webmail.sso.singlesignoff -v 1
```

Messenger Express and Delegated Administrator for Messaging

To enable single sign-on between Messenger Express and Delegated Administrator, you must perform additional steps as follows:

1. Configure Directory Server
   a. Create a proxy user account entry in the Directory Server
   b. Create an ACI (Access Control Instructions) for proxy authentication
2. Configure Delegated Administrator
   a. Add the proxy user credentials
   b. Add the single sign-on cookie information
   c. Add the participating servers verification URL
3. Restart the Enterprise Server

To configure Directory Server, you will use the `ldapmodify` utility. For more information about this utility, see your Directory Server documentation.

To configure Delegated Administrator, you will modify the following configuration files:

DA-server-root`/nda/classes/netscape/nda/servlet/resource.properties`

Enterprise-Server-Root`/https-`*instancename*`/config/servlets.properties`

Enterprise-Server-Root`/https-`*instancename*`/config/contexts.properties`

Step 1a. Create a Proxy User Account

The proxy user account allows users to bind to the Directory Server for proxy authentication. You must create this account (using the ldapmodify utility) in a base suffix other than the Delegated Administrator base suffix (osiroot). For example, the following is an example of a proxy user account entry (we will assume that osiroot is at o=isp):

```
dn: uid=proxy, ou=people, o=siroe.com, o=mailqa
objectclass: top
objectclass: person
objectclass: organizationalperson
objectclass: inetorgperson
uid: proxy
givenname: Proxy
sn: Auth
cn: Proxy Auth
userpassword: proxypassword
```

Step 1b. Create an ACI for Proxy Authentication

Next, using the `ldapmodify` utility, create an ACI for the suffixes you created at install time:

- `osiroot` - The suffix you entered to store the user data
- `dcroot` - The suffix you entered to store the domain information
- `osiroot` - The suffix you entered to store the configuration information (the default is `osiroot`)

For example, the following is an example of an ACI entry:

```
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=siroe.com, o=mailqa";)
```

Step 2a. Add the Proxy User Credentials to the resource.properties File

To configure Delegated Administrator for proxy authentication, uncomment and modify the following entries in the Delegated Administrator `resource-properties` file:

`LDAPDatabaseInterface-ldapauthdn=`*Proxy-Auth-DN*

`LDAPDatabaseInterface-ldapauthpw=`*Proxy-Auth-Password*

For example:

```
LDAPDatabaseInterface-ldapauthdn=uid=proxy, ou=people,o=siroe.com,
o=mailqa
LDAPDatabaseInterface-ldapauthpw=proxypassword
```

Step 2b. Add the Single Sign-On Cookie Information

To add the single sign-on cookie information, define a context identifier for Delegated Administrator and specify a cookie name for the context, as follows:

- To define a context identifier, edit the Enterprise Server `servlets.properties` file and uncomment all lines containing the text `servlet.xxxxx.context=ims50`.
- To specify a cookie name for the context in the Delegated Administrator configuration, add the following entry to the Delegated Administrator `resource.properties` file:

  ```
  NDAAuth-singleSignOnId=ssogrp1-
  NDAAuth-applicationId=nda45
  ```

- To specify a cookie name for the context in the Enterprise Server configuration, add the following entry to the Enterprise Server `contexts.properties` file:

  ```
  context.ims50.sessionCookie=ssogrp1-nda45
  ```

Step 2c. Add the Participating Servers Verification URL

To verify a single sign-on cookie it received, Delegated Administrator must know who to contact. You must provide a verification URL for all known participating servers.

For purposes of the following example, assume Messenger Express is installed and its application ID is `msg50`. Edit the Delegated Administrator `resource.properties` file and add an entry such as:

```
verificationurl-ssogrp1-msg50=http://<webmail_hostname>:port/
VerifySSO?
```

```
verificationurl-ssogrp1-nda45=http://<nda_hostname>:port/
VerifySSO?
```

Step 3. Restart the Enterprise Server

After you've made the configuration changes described in steps 1a through 2c, you must restart the Enterprise Server for the changes to take effect.

Customizing Directory Lookups

iPlanet Messaging Server cannot function without an LDAP-based directory system such as the iPlanet Directory Server. Messaging Server and Console require directory access for three purposes:

- When you first install a Messaging Server, you enter configuration settings for the server. These settings are stored in a central *configuration directory*. Part of the installation process includes configuring the connection to that directory.
- When you create or update account information for mail users or mail groups, the information is stored in a directory called the *user directory*. Your server group's Administration Server is configured at installation so that when you access Users and Groups, Console connects by default to the user directory that defines your *administrative topology*—the set of iPlanet servers that all share the same configuration directory and user directory.
- When routing messages and delivering mail to mailboxes, Messaging Server looks up information about the sender or recipients in the user directory. By default, Messaging Server looks in the same user directory that its Administration Server has been configured to use.

You can modify each of these directory-configuration settings in the following ways:

- The Administration Server interface of Console lets you change the connection settings for the configuration directory. (For more information, see the Administration Server chapter of *Managing Servers with Netscape Console*.)
- The Users and Groups interface of Console lets you temporarily connect to a different user directory from the default when making changes to user and group information. (For more information, see the Users and Groups chapter of *Managing Servers with Netscape Console*.)
- The Messaging Server interface of Console lets you configure your Messaging Server to connect to a different user directory from the default defined by the Administration Server. This is the configuration task discussed in this section.

Reconfiguring your Messaging Server to connect to a different user directory for user and group lookups is strictly optional. In most cases, the user directory that defines your server's administrative domain is the one used by all servers in the domain.

NOTE If you specify a custom user directory for your Messaging Server lookups, you must also specify that same directory whenever you access the Users and Groups interface of Console to make changes to the directory's user or group information. For more information, see Chapter 3, "Managing Mail Users and Mailing Lists."

Console. To modify the Messaging Server LDAP user-lookup settings by using Console:

1. From Console, open the Messaging Server whose LDAP connection you want to customize.
2. Click the Configuration tab.
3. Select the Services folder in the left pane.
4. Select the LDAP tab in the right pane. The LDAP form appears.

   The LDAP form displays the configuration settings for both the configuration directory and the user directory. The configuration-directory settings, however, are read-only in this form. See the Administration Server chapter of *Managing Servers with Netscape Console* if you need to change them.
5. To change the user-directory connection settings, click the box labeled "Use messaging server specific directory settings".

6. Update the LDAP configuration by entering or modifying any of the following information (for explanations of directory concepts, including definitions of terms such as *distinguished name*, see the *Directory Server Administrator's Guide*):

   Host name: The name of the host machine on which the directory containing your installation's user information resides. This is typically not the same as the Messaging Server host, although for very small installations it might be.

   Port number: The port number on the directory host that Messaging Server must use to access the directory for user lookup. This number is defined by the directory administrator, and may not necessarily be the default port number (389).

   Base DN: The search base—the distinguished name of a directory entry that represents the starting point for user lookups. To speed the lookup process, the search base should be as close as possible in the directory tree to the information being sought. If your installation's directory tree has a "people" or "users" branch, that is a reasonable starting point.

   Bind DN: The distinguished name that your Messaging Server uses to represent itself when it connects to the directory server for lookups. The bind DN must be the distinguished name of an entry in the user directory itself that has been given search privileges to the user portion of the directory. If the directory allows anonymous search access, you can leave this entry blank.

7. To change the password used, in conjunction with the Bind DN, to authenticate this Messaging Server to the LDAP directory for user lookups, click the Change Bind password button. A Password-Entry window opens, into which you can enter the updated password.

   Your own security policies should determine what password you use in this situation. Initially, the password is set to no password. The password is not used if you have specified anonymous access by leaving the Bind DN field blank.

   This step updates the password stored in server configuration, but does not change the password in the LDAP server. This account is also used for PAB lookups by default. The following two steps need to be performed after the password has been changed.

8. Modify the password for the user specified in the configuration attribute `local.ugldapbinddn`. This user account exists in the directory server specied in configuration attribute `local.ugldaphost`.

9. If the same account is used for PAB access, specified in the attributes `local.service.pab.ldapbinddn` and `local.service.pab.ldaphost`, then the password stored in `local.service.pab.ldappasswd` must also be updated.

To return to using the default user directory, uncheck the “Use messaging server specific directory settings” box.

Command Line. You can also set values for the user-directory connection settings at the command line as follows. Be sure to also set the LDAP and PAB password as described in the steps 8 and 9 above.

To specify whether to use messaging server specific directory settings:

```
configutil -o local.ugldapuselocal -v [ yes | no ]
```

To specify the LDAP host name for user lookup:

`configutil -o local.ugldaphost -v` *name*

To specify the LDAP port number for user lookup:

`configutil -o local.ugldapport -v` *number*

To specify the LDAP base DN for user lookup:

`configutil -o local.ugldapbasedn -v` *basedn*

To specify the LDAP bind DN for user lookup:

`configutil -o local.ugldapbinddn -v` *binddn*

Encryption Settings

You can use Console to enable Secure Sockets Layer (SSL) encryption and authentication for Messaging Server and to select the specific encryption ciphers that the server will support across all of its services.

Although this task is a general configuration task, it is described in the section “Enabling SSL” in Chapter 11, “Configuring Security and Access Control” which also contains background information on all security and access-control topics for Messaging Server.

Chapter 3

Managing Mail Users and Mailing Lists

This chapter describes how to use the Console interface to create and manage your users' mail accounts and mailing lists.

This chapter has the following sections:

- Introduction
- Managing Mail Users
- Managing Mailing Lists

Introduction

An LDAP user directory can contain a wide range of information about an organization's employees, members, clients, or other types of individuals that in one way or another "belong" to the organization. These individuals constitute the *users* of the organization.

In the LDAP directory, the information about users is structured for efficient searching, with each user entry identified by a set of attributes. Directory attributes associated with a user can include the user's name and other identification, division membership, job classification, physical location, name of manager, names of direct reports, access permission to various parts of the organization, and preferences of various kinds.

In an organization with electronic messaging services, many if not all users hold mail accounts. For iPlanet Messaging Server, mail-account information is not stored locally on the server; it is part of the LDAP user directory. The information for each mail account is stored as mail attributes attached to a user's entry in the directory.

To retrieve or modify information for a specific user's mail account, you must access that user's mail attributes in the directory. To do this, you can use the iPlanet Console interface (described in this chapter), the iPlanet Delegated Administrator for Messaging interface, or you can directly modify LDAP by using LDAP tools.

This chapter explains how you can use Console to create and manage your users' mail accounts and mailing lists. Although this functionality is provided in Console, iPlanet recommends you use iPlanet Delegated Administrator for Messaging (or LDAP tools) for managing users, groups, and domains.

Delegated Administrator for Messaging provides full support for managing users, groups, family groups, and hosted domains. With Delegated Administrator, you can delegate user and group administration, and set up administrators per hosted domain. Delegated Administrator provides a GUI interface for administrators to manage users and groups and for end users to manage their own mail accounts. Administrators can also use the Delegated Administrator command-line utilities for managing users and groups. For more information about using Delegated Administrator, see the *Delegated Administrator Installation and Administration Guide* and the Delegated Administrator online help. For more information about using LDAP tools to manage users, groups, and domains, see the *Messaging Server Provisioning Guide*.

If you choose to use Console, for any user in your user directory, you can perform the following tasks:

- Access the user's mail account
- Specify mail addressing information for the account
- Define the delivery method(s) and attributes for the account
- Specify forwarding addresses and attributes for the account
- Specify auto-reply procedures for the account

For any group in your user directory, you can perform the following tasks:

- Access the group's mailing list
- Specify mail addressing information for the mailing list
- Specify *email-only* members for the mailing list
- Define restrictions for posting messages to the mailing list
- Define and enable message-rejection actions for the mailing list

Subsequent sections in this chapter give detailed discussions of these administrative tasks. Before you can perform them, however, you must first enable the mail-administration interface, as described in the next section.

Managing Mail Users

Accessing Mail Users

This section describes how to open the mail administration interface for your users. Messaging Server mail accounts are stored as attributes of user entries in your enterprise's central LDAP user directory. Therefore, to manage mail accounts, you modify user entries in that directory.

Creating a New User

To create a new mail account, you create a new user in the directory. You must also install a mail account for that user; if you do not install the mail account, the mail-administration portion of Console is not available for that user. (The full process of creating a user and specifying other kinds of user information is described in more detail in Chapter 4, "User and Group Administration," of *Managing Servers with Netscape Console*.)

To create a new mail user:

1. In the Console main window, click the Users and Groups tab.
2. From the drop-down list, choose New User and click Create.
3. Select an organizational unit for the user and click OK. The Create User window opens.
4. Enter information about the user as described in Chapter 4, "User and Group Administration," of *Managing Servers with Netscape Console*.
5. Leave the Create User window open and click the Account tab. A list of installed products for the new user's account appears in the right pane.
6. Click the Mail Account Install box. The Mail tab becomes visible in the Create User window.
7. Click the Mail tab in the Create User window, then click the tab you want in the right pane.

8. Enter your changes, then click OK at the bottom of the Create User window.

NOTE Make sure you complete all setup procedures in the relevant tabs before clicking OK.

Accessing an Existing User

To modify an existing mail account or to add mail capabilities to an existing user, you access the appropriate user in the user directory and then add or modify that user's mail-account attributes.

To access mail information for an existing user:

1. In the Console main window, click the Users and Groups tab.
2. In the Users and Groups main window, Click Search or Advanced Search.
3. Enter your search criteria (such as the user's last name) in the Search window, and perform the search of the user directory.
4. Return to the Users and Groups main window, select a user from the search results and click Edit.
5. If the Mail tab is not visible in the Edit Entry window, do this:
   a. Click the Account tab. A list of installed accounts appears in the right pane.
   b. Check the Mail Account box. The Mail tab displays in the Edit Entry window.
6. Click the Mail tab in the Edit Entry window, then click the tab you want in the right pane.
7. Enter your changes, then click OK at the bottom of the Edit Entry window.

Specifying User Email Addresses

Before mail can be delivered successfully to a user, you must specify the mail addressing information for that user. This consists of the Messaging Server host name, the user's primary address, and any alternate addresses. The host name and primary address information is mandatory; alternate address information is optional.

To specify a user's mail addressing information:

1. In Console, access the Create User or Edit Entry window, as described in “Accessing Mail Users” on page 47.
2. Click the Mail tab.
3. Click the Settings tab, if it is not already active.
4. (Required) Enter the Messaging Server host name.

   This is the machine hosting the Messaging Server that will process this user’s mail. This must be the fully-qualified domain name (FQDN) known to the Messaging Server on that machine.
5. (Required) Enter the user’s primary email address.

   This is the publicized address to which this user’s mail is sent. There can be only one primary address for a user, which must be a valid, correctly formatted SMTP address conforming to RFC 821 specifications.

   If you want to implement host name hiding (the host name in the user’s address is not shown in the outgoing mail header), do not specify the host name in the Primary email address field. Instead, enter an alternate address that includes the host name as described in the next step.
6. (Optional) Add an address to the Alternate Address list.

   An alternate address is essentially an alias for the user’s primary address. You can use this feature to:

   - Ensure proper delivery of frequently misspelled addresses (such as “Smith” as an alias for “Smythe”).
   - Enable host name hiding in outgoing mail headers. To do so, supply an alternate address that includes the host name and do not include the host name in the user’s Primary email address. For example, enter `jsmith@siroe.com` as a Primary email address and then enter `jsmith@sesta.com` as an Alternate address. When this user sends mail, the outgoing header will show `jsmith@siroe.com`, but all mail sent to that address (including replies) are actually routed to `jsmith@sesta.com` (assuming that `sesta.com` is a valid host name).

   You can specify any number of alternate addresses for a particular user, as long as each address is unique. Messages that arrive for any of these aliases are directed to the primary address.

   To add an alternate address:

   a. Click the Add button beneath the Alternate Addresses field.

b. In the Alternate Addresses window, enter an alternate address. (You can add as many alternate addresses as you like, but you can enter only one address each time you open this window.)

c. Click OK to add the alternate address and close the Alternate Addresses window. (To enter another alternate address, click Add again to re-open the Alternate Addresses window.)

7. Click OK at the bottom of the Edit Entry window if you have finished making changes to this user's mail information. Otherwise, click other tabs to continue making changes.

Configuring Delivery Options

Messaging Server supports three principal mail-delivery options that you can enable and configure, in any combination, for each user. You can provide regular POP/IMAP delivery, program delivery, and UNIX delivery (for clients of a UNIX Messaging Server host).

iPlanet Delegated Administrator for Messaging also provides an end-user HTML interface through which users can themselves enable and configure these options. The Console interface and the Delegated Administrator interface both manipulate the same directory attributes; when opened, each shows the current settings, whether they were set by the administrator or by the user.

To configure delivery options for a user:

1. In Console, access the Create User or Edit Entry window, as described in "Accessing Mail Users" on page 47.

2. Click the Mail tab.

3. Click the Delivery tab.

4. Select the delivery method or methods you want to enable for this user:

- To specify POP/IMAP delivery, follow the instructions in "Specifying POP/IMAP Delivery" on page 51.
- To specify program delivery, follow the instructions in "Specifying Program Delivery" on page 51.
- To specify UNIX delivery, follow the instructions in "Specifying UNIX Delivery" on page 52.

5. Click OK at the bottom of the Edit Entry window if you have finished making changes to this user's mail information. Otherwise, click other tabs to continue making changes.

Specifying POP/IMAP Delivery

Specifying this option enables mail delivery to the user's regular POP3 or IMAP4 mailboxes. To enable POP/IMAP delivery for this user:

1. Click the Delivery tab.
2. Check the POP/IMAP box, and click the Properties button to open the POP/IMAP Delivery window.
3. (Optional) Enter the nickname (not the path name or absolute physical path) of the message-store partition to which the user's messages will be delivered and stored for processing. If you leave this field blank, the current primary partition is used. For more information, see "Managing the Message Store" on page 237.
4. (Optional) Enter the storage limit, or disk quota, to be allotted to the user. The quota can be the default specified (see "Configuring Message Store Quotas" on page 246), unlimited (no maximum storage limit), or you can specify a limit (in KB or MB).
5. (Optional) Enter the message number limit to be allotted to the user. The limit can be the default specified (see "Configuring Message Store Quotas" on page 246), unlimited (no maximum storage limit), or you can specify a limit (in numbers).

Specifying Program Delivery

Specifying this option provides a mechanism for forwarding messages to an external application for processing before delivery to the user.

NOTE This section describes only how to make the program delivery option available to an individual user. Before you can make it available to a user, you must first enable the program delivery module as a whole, which requires performing several other administrative tasks. For details, see "Configuring Channel Definitions" on page 153.

To enable program delivery for this user:

1. Click the Delivery tab.
2. Check the Program delivery box, and click the Properties button to open the Program Delivery window.
3. Enter the external application command(s) to be used for processing this user's mail.
4. Click OK.

Specifying UNIX Delivery

Specifying this option selects UNIX delivery for this user. The UNIX delivery feature allows messages to be delivered to the user's designated UNIX mailbox. UNIX delivery is available only to users whose Messaging Server runs on a UNIX host machine.

To enable UNIX delivery for this user:

1. Click the Delivery tab.
2. Check the UNIX delivery box.

NOTE To provide UNIX delivery to Messaging Server users, you must also perform normal UNIX mail administrative tasks

Specifying Forwarding Addresses

The mail-forwarding feature of Messaging Server enables a user's mail to be forwarded to another address instead of or in addition to the primary address for that user.

iPlanet Delegated Administrator for Messaging provides an end-user HTML interface through which users can themselves specify forwarding addresses. The Console interface and the Delegated Administrator interface both manipulate the same directory attributes; when opened, each shows the current settings, whether they were set by the administrator or by the user.

To specify forwarding-address information for a user:

1. In Console, access the Create User or Edit Entry window, as described in "Accessing Mail Users" on page 47.
2. Click the Mail tab.

3. Click the Forwarding tab.

   The Forwarding Address field shows the current set of forwarding addresses, if any, for the user.

4. To add a forwarding address, Click Add.

5. In the Forwarding Address window, enter a forwarding address.

6. Click OK to add the address to the Forwarding address field in the Mail Forwarding tab and close the Forwarding Address window.

7. Click OK at the bottom of the Edit Entry window if you have finished making changes to this user's mail information. Otherwise, click other tabs to continue making changes.

NOTE Do not set up forwarding address for two users on the same Messaging Server to point to each other if both user accounts have no other delivery type enabled. Doing so can cause mail delivery problems.

Configuring Auto-Reply Settings

The auto-reply feature of iPlanet Messaging Server lets you specify an automatic response to incoming mail for a user. You can specify three different auto-reply modes: echo mode, vacation mode, and auto-reply mode.

iPlanet Delegated Administrator for Messaging also provides an end-user HTML interface through which users can themselves enable and configure auto-reply settings. The Console interface and the Delegated Administrator interface both manipulate the same directory attributes; when opened, each shows the current settings, whether they were set by the administrator or by the user.

To enable an auto-reply service for a user:

1. In Console, access the Create User or Edit Entry window, as described in "Accessing Mail Users" on page 47.

2. Click the Mail tab.

3. Click the Auto-Reply tab.

4. Select one of the auto-reply modes:

   Off: Disables auto-reply for this user.

Echo: An automatic reply is sent for each received message. If you select this mode, you can enter a reply message in the Message field.

Vacation: The first message received by this user from a given sender generates an automatic response; subsequent messages from that sender do not generate a response until the automatic reply time-out is reached. When the time-out is reached, a new message is sent, once, until the next time-out is reached, and so on. If you select this mode, you use the Vacation start/end date options and enter a reply message in the Reply text field.

5. If you selected vacation mode, supply dates and times to determine when the auto-reply message should start and end:
   - Check the Vacation start/end date checkbox.
   - Click the Edit buttons for Start and End then use the calendar that displays to specify a date and time.
6. Specify an automatic reply time-out value in hours or days.
7. If you selected echo or vacation mode, type an auto-reply subject line, then type a reply message to be returned to the sender.

   You can type a reply message for internal senders and a reply message for external senders. If you type a reply only for internal senders, only senders within your domain will receive an automatic reply.

   You can create one message in each of several available languages that you select with the drop-down list located above the message text area.
8. Click OK at the bottom of the Edit Entry window if you have finished making changes to this user's mail information. Otherwise, click other tabs to continue making changes.

Configuring Authorized Services

To enable the mail services for which this user can access mail:

1. In Console, access the Create User or Edit Entry window, as described in "Accessing Mail Users" on page 47.
2. Click the Mail tab.
3. Click the Authorized Services tab.

   The Authorized Services window shows the services that apply to a particular domain.

4. You can Add, Edit, or Delete services by clicking the associated button. The "Modify rule for authorized services" window appears.
5. From the service drop-down list, choose the service you wish to create a rule for (IMAP, POP, SMTP, HTTP, All).
6. Specify Allow or Deny and specify the domain to which this rule applies.
7. Click OK to submit your changes.

Managing Mailing Lists

Accessing Mailing Lists

This section describes how to get to the administration interface for your mailing lists. Because Messaging Server mailing lists are stored as attributes of group entries in an LDAP user directory, managing mailing lists means accessing and modifying directory groups.

Creating a New Group

To create a new mailing list, you create a new group in the directory. You must also install a mail account for that group; if you do not install the mail account, the mail-administration portion of Console is not available for that group. (The full process of creating a directory group and specifying other kinds of group information is described in more detail in Chapter 4, "User and Group Administration," of *Managing Servers with Netscape Console*.)

To create a new mailing list:

1. In the Console main window, click the Users and Groups tab.
2. From the drop-down list, choose New Group and click Create.
3. Select an organizational unit for the group and click OK.
4. In the Create Group window, enter the information required to create the group entry as described in Chapter 4, "User and Group Administration," of *Managing Servers with Netscape Console*.

   Note that For mailing-list purposes only, you do *not* have to add members using the Users and Groups Members tab; you can instead add them using the Mail account Email-Only Members tab:

- Regular group members have full mailing-list privileges, but they also have any other privileges that their group membership indicates. You add regular members (either static or dynamic) through the Members tab.
- Mailing-list members have group privileges limited to those provided by the mailing-list component of the group (which may or may not be the only purpose for the group's existence). Mailing-list members are called *email-only members*, and you add them through the Mail tab.

5. Leave the Create Group window open and click the Account tab.

   A list of installed products for the group account appears in the right pane.

6. Click the Mail Account box.

   The Mail tab becomes visible in the Create Group window.

7. Click the Mail tab in the Create Group window, then click the appropriate tab in the right pane.

8. Enter your changes, then click OK at the bottom of the Create Group window.

   This action submits your entries and dismisses the Create Group window.

NOTE Clicking OK at the bottom of any mail administration window submits all of the current mail configuration information entered in all of the mail administration tabs. Make sure you complete all setup procedures in the relevant windows before clicking OK.

Accessing an Existing Group

To modify an existing mailing list, or to add mailing-list capabilities to an existing group, you access the appropriate group in the user directory and then add or modify its mail-account attributes.

To access mailing-list information for an existing group:

1. In the Console main window, click the Users and Groups tab.
2. In the Users and Groups main window, Click Search or Advanced Search.
3. Enter your search criteria (such as the group's name) in the Search window, and perform the search of the user directory.
4. Return to the Users and Groups main window, select a group from the search results and click Edit.
5. If the Mail tab is not visible in the Edit Entry window, do this:

- Click the Account tab. A list of installed accounts appears in the right pane.
- Check the Mail Account box. The Mail tab displays in the Edit Entry window.

6. In the Edit Entry window, click the Mail tab, then click the tab you want in the right pane.

   (These tabs are identical to those you access through the Create Group window.)

7. Enter your changes, then click OK at the bottom of the Edit Entry window to submit your modifications.

Specifying Mailing List Settings

Before mail can be delivered successfully to your mailing list, you must specify its mail-addressing information. This consists of the primary address for the group and any alternate addresses you want to accept as aliases to the primary address. You can also specify the owner(s) of the list, optional descriptive information, members, attributes, restrictions, and actions (email responses) of the mailing list.

To specify mailing-list information:

1. In Console, access the Create Group or Edit Entry window, as described in “Accessing Mailing Lists” on page 55.
2. Click the Mail tab.
3. Click the Settings tab, if it is not already the active tab.
4. (Required) Enter the mailing list’s primary email address.

   This is the publicized address to which this list’s mail will be delivered. There can be only one primary address for a list. It must be a correctly formatted SMTP address that conforms to RFC 821 specifications.

5. (Optional) Specify an alternate address for the mailing list.

   An alternate address is an alias for the group’s primary address. You can use this feature to:

   - Ensure proper delivery of a frequently misspelled address.
   - Enable host name hiding in outgoing mail headers. To do so, supply an alternate address that includes the host name and do not include the host name in the group’s Primary email address.

You can specify any number of alternate addresses for a group, as long as each address is unique. Messages that arrive for any of these aliases are directed to the primary address.

To add an alternate email address:

a. Click the Add button beneath the Alternative email addresses field.

b. In the Alternative Email Addresses window, enter an alternate address. (You can add as many alternate addresses as you like, but you can enter only one address each time you open this window.)

c. Click OK to add the alternate address and close the Alternative Email Addresses window. (To enter another alternate address, click Add again to re-open the Alternative Email Addresses window.)

6. (Optional) In the "Errors to" field, enter the email address of a person to whom errors in posting messages to the list should be sent.

7. (Optional) In the "Messaging Server hostname" field, enter the host name of the machine hosting this mailing list.

If the "Primary email address" field for this mailing list includes a host name, you can leave this field blank. If you implement host-name hiding by having no host name in the primary email address, specify the host name in this field.

Unlike a user mail account, if you do not specify a host name for a mailing list, any host that has access to the list's LDAP entry will be able to process the list (which, in most cases, is what you want). If you want to restrict processing of the list to one or more specific hosts, you should specify one or more host names. For example, you may want to force a large group to be processed on an under-utilized server to reduce stress on a server that is more heavily used.

Note that this window lets you enter only one host name at a time. To enter multiple host names, use the `ldapmodify` command line utility.

8. (Optional) Enter a mailing list owner.

A list owner has administrative privileges for adding or removing users, modifying configuration settings, or deleting the list.

To specify a new mailing list owner, click the Owners tab and then either:

- Click Add, then enter the distinguished name (DN) of a new mailing list owner (such as `uid=jsmith, ou=people, o=siroe.com`) in the Enter List Owner's DN window and click OK.
- Click Search to open the Search Users and Group window to locate an owner.

Note that selecting an owner from the Search Users and Group window automatically adds the correct syntax of the DN for you. For more details on the Search Users and Groups window, see Chapter 4, “User and Group Administration,” of *Managing Servers with Netscape Console*.

9. (Optional) Add descriptive information.

   To add text or a URL for information purposes (not for use by Messaging Server), click the Descriptions tab, then use one or both of the following options:

   - Enter a description of the purpose or nature of the mailing list.
   - Enter a URL to an HTML page providing additional information about the mailing list. This is for informational purposes only; the URL is not used by Messaging Server.

10. Click OK at the bottom of the Edit Entry window if you have finished making changes to this mailing list. Otherwise, click other tabs to continue making changes.

Specifying List Members

To add email-only members to your mailing list, use one or both of the following methods:

- Explicitly add each member to the mailing list.
- Define dynamic criteria to be applied to the user directory as a filter for determining group membership.

The mailing-list members described here are called *email-only members* in the Users and Groups interface of Console because they have group privileges limited to those provided by the mailing-list component of the group. “Regular” group members, which you add using a different part of the interface (described in *Managing Servers with Netscape Console*), might have additional privileges or responsibilities beyond those of mailing-list members. For more information on groups, see Chapter 4, “User and Group Administration,” of *Managing Servers with Netscape Console*.

Defining Dynamic Membership Criteria

Dynamic criteria consist of LDAP search URLs that are used as filters in searching the user directory for determining membership. This mechanism is dynamic in that, when a message arrives for the group, the individuals that receive it are determined by a directory search rather than by consulting a static list of names. You can thus create and maintain very large or complex groups without having to track each member explicitly.

LDAP search filters must be formatted in LDAP URL syntax. For more detailed information on constructing LDAP filters, see Chapter 4, "User and Group Administration," of *Managing Servers with Netscape Console*. See also the iPlanet Directory Server documentation and RFC 1959.

An LDAP URL has the following syntax:

`ldap://hostname:port/base_dn?attributes?scope?filter`

where the options of the URL have the following meanings:

Table 3-1 LDAP URL Options

option	Description
hostname	Host name of the Directory Server (Defaults to the Directory server host name used by Messaging Server).
port	Port number for the LDAP server. If no port is specified, it defaults to the standard LDAP port used by Messaging Server.
base_dn	Distinguished name of an entry in the directory, to be used as the search base. This component is required.
attributes	The attributes to be returned. These attributes are supplied by Messaging Server.
scope	Scope of search: A scope of `base` retrieves information only on the search base (*base_dn*) itself. A scope of `one` retrieves information one level below the search base (the search-base level is not included). A scope of `sub` retrieves information on the search base and all entries below the search base.
filter	Search filter to apply to entries within the specified scope of the search. If no filter is specified, (`objectclass=*`) is used.

The following is an example of an LDAP search URL that filters for users who have Sunnyvale as their mail host:

```
ldap:///o=Siroe Corp,c=US??sub?(&(mailHost=sunnyvale.siroe.com)
  (objectClass=inetLocalMailRecipient))
```

The above URL filters for users who are members of the organization of Siroe (`o=Siroe`), in the United States (`c=US`), and have a mail host of Sunnyvale (`mailHost=sunnyvale`). The `objectClass` attribute defines the type of entry for which to search, in this case `inetLocalMailRecipient` (`objectClass=inetLocalMailRecipient`).

Note that when you create a search filter using Console, all group names are ignored; that is, only user names are included in the search results whereas group members are not. The purpose of this setting is to avoid duplicating users that are also group members in the search results. This setting can be overridden using the command line configuration utility (`configutil`), but it is not recommended.

As noted in the next section, Console provides a template window (the Construct LDAP Search URL window) that you can use as an aid in building a search URL.

Adding Mailing-List Members

To add (email-only) members to a mailing list:

1. In Console, access the Create Group or Edit Entry window, as described in “Accessing Mailing Lists” on page 55.
2. Click the Mail tab.
3. Click the Email-only Members tab.
   - (Optional) To specify an LDAP Search URL for determining membership, click the Add button beneath the “Dynamic criteria for email-only membership” field, then in the Add Dynamic Criterion window:
   - Enter an LDAP Search URL in the field or click the Construct button to open the Construct LDAP Search URL window, a template that aids construction of the search URL.
   - Click OK to add your entry to the “Dynamic criteria for email-only membership” field and dismiss the Add Dynamic Criterion window.

   For instructions on creating an LDAP Search URL, see “Defining Dynamic Membership Criteria” on page 60.

4. (Optional) To add an individual member to the mailing list, click the Add button beneath the "Members with email only membership" field, then in the Add Email-Only Member window:
   - Enter the primary address for the new member in the field. The primary address must be a correctly-formatted SMTP address that conforms to RFC 821 specifications. You should not enter an alternate address—especially if you specify restrictions for the group. You can add only one new member each time you open this window; the field cannot hold more than one address.
   - Click OK to add the user to the members list and dismiss the Add Email-Only Member window. To enter another address, click Add again to re-open the Add Email-Only Member window.
5. Click OK at the bottom of the Edit Entry window if you have finished making changes to this mailing list. Otherwise, click other tabs to continue making changes.

Defining Message-Posting Restrictions

You can impose various kinds of restrictions on messages sent to a mailing list. You can define the set of people allowed to post messages, you can require authentication of senders, you can restrict where posted messages can come from, and you can limit the size of a posted message. Messages that violate the restrictions are rejected.

NOTE Although these restrictions are useful for controlling several aspects of the incoming messages for a group, they are not intended to provide high-security access control.

To define message-posting restrictions for a group:

1. In Console, access the Create Group or Edit Entry window, as described in "Accessing Mailing Lists" on page 55.
2. Click the Mail tab.
3. Click the Restrictions tab.
4. (Optional) Define the allowed senders by choosing one of the following options:

- **Anyone:** No restrictions on senders. (This is the default.) Note that if you choose this option, you cannot select SMTP authentication described in the next step.
- **Anyone in the mailing list:** Only mailing-list members (including group members that are not email-only members) can post messages.
- **Anyone in the following list:** Only those users explicitly listed in the following field can post messages.

If you choose "Anyone in the following list", to add a sender click Add below the Allowed Senders field—or you can click Search to open the Search Users and Groups window. If you click Add, the Add Allowed Sender window opens. Enter the email address or distinguished name (DN) of the allowed sender into the field. Click OK to add the sender to the Allowed Senders field and dismiss the Add Allowed Sender window. Repeat this step for all other allowed senders you want to add.

For a description of the Search Users and Groups window, see *Managing Servers with Netscape Console*.

5. (Optional) Define the allowed sender domains to restrict where senders can post messages from:
   - Click the Add button beneath the Allowed sender domains field.
   - In the Add Allowed Sender Domain window, enter a domain name, then click OK to add the domain to the list.

   Note that a domain automatically includes any of its subdomains. For example, `siroe.com` includes `sales.siroe.com`.
6. (Optional) Define the maximum permitted message size.

   Enter the size (in bytes).
7. Click OK at the bottom of the Edit Entry window if you have finished making changes to this mailing list. Otherwise, click other tabs to continue making changes.

Defining Moderators

You can add one or more moderators for a mailing list.

When a moderator receives the forwarded message, that person decides how to process the message. (In the case of multiple moderators, processing of the message is determined by the action taken by the first moderator.) Processing might include approving the message and forwarding it back to the list (perhaps with a password) or deleting it.

To define moderators for a mailing list:

1. In Console, access the Create Group or Edit Entry window, as described in "Accessing Mailing Lists" on page 55.
2. Click the Mail tab.
3. Click the Moderators tab.
4. Click the Add button beneath the List moderators field.
5. In the Add Moderator window, enter a moderator's primary email address or distinguished name (DN) in the field. You can enter the address explicitly or you can click Search to use the Search Users and Groups window to locate an address. Note that you can add only one moderator each time you open the Add Moderator window.

   For a description of the Search Users and Groups window, see *Managing Servers with Netscape Console*.
6. Click OK to add the moderator to the List Moderators list and dismiss the Add Moderator window. (To enter another address, click Add again to re-open the Add Moderator window.)
7. Click OK at the bottom of the Edit Entry window if you have finished making changes to this mailing list. Otherwise, click other tabs to continue making changes.

Chapter 4

Configuring POP, IMAP, and HTTP Services

iPlanet Messaging Server supports the Post Office Protocol 3 (POP3), the Internet Mail Access Protocol 4 (IMAP4), and the HyperText Transfer Protocol (HTTP) for client access to mailboxes. IMAP and POP are both Internet-standard mailbox protocols. Messenger Express, a web-enabled electronic mail program, lets end users access their mailboxes using a browser running on an Internet-connected computer system using HTTP.

This chapter describes how to configure your server to support one or more of these services by using the iPlanet Console or by using command-line utilities. For information on configuring Simple Mail Transfer Protocol (SMTP) services, see Chapter 6, “About MTA Services and Configuration.”

This chapter contains the following sections:

- General Configuration
- Login Requirements
- Performance Parameters
- Client Access Controls
- Configuring POP Services
- Configuring IMAP Services
- Configuring HTTP Services

General Configuration

Configuring the general features of the Messaging Server POP, IMAP, and HTTP services includes enabling or disabling the services, assigning port numbers, and optionally modifying service banners sent to connecting clients. This section provides background information; for the steps you follow to make these settings, see “Configuring POP Services” on page 73, “Configuring IMAP Services” on page 74, and “Configuring HTTP Services” on page 76.

Enabling and Disabling Services

You can control whether any particular instance of Messaging Server makes its POP, IMAP, or HTTP service available for use. This is not the same as starting and stopping services (see “Starting and Stopping Services” on page 28); to function, POP, IMAP, or HTTP must be both enabled and started.

Enabling a service is a more “global” process than starting or stopping a service. For example, the Enable setting persists across system reboots, whereas you must restart a previously “stopped” service after a reboot.

There is no need to enable services that you do not plan to use. For example, if a Messaging Server instance is used only as a message transfer agent (MTA), you should disable POP, IMAP, and HTTP. If it is used only for POP services, you should disable IMAP and HTTP. If it used only for web-based email, you should disable both POP and IMAP.

Specifying Port Numbers

For each service, you can specify the port number that the server is to use for service connections:

- If you enable the POP service, you can specify the port number that the server is to use for POP connections. The default is 110.
- If you enable the IMAP service, you can specify the port number that the server is to use for IMAP connections. The default is 143.
- If you enable the HTTP service, you can specify the port number that the server is to use for HTTP connections. The default is 80.

You might need to specify a port number other than the default if you have, for example, two or more IMAP server instances on a single host machine, or if you are using the same host machine as both an IMAP server and a Messaging Multiplexor server. (For information about the Multiplexor, see Chapter 5, “Messaging Multiplexor.”)

Keep the following in mind when you specify a port:

- Port numbers can be any number from 1 to 65535.
- Make sure the port you choose isn’t already in use or reserved for another service.

Ports for Encrypted Communications

Messaging Server supports encrypted communications with IMAP and HTTP clients by using the Secure Sockets Layer (SSL) protocol. For general information on support for SSL in Messaging Server, see “Configuring Encryption and Certificate-Based Authentication” on page 288.

IMAP Over SSL

You can accept the default IMAP over SSL port number (993) or you can specify a separate port for IMAP over SSL.

Messaging Server provides the option of using separate ports for IMAP and IMAP over SSL because most current IMAP clients require separate ports for them. Same-port communication with both IMAP and IMAP over SSL is an emerging standard; as long as your Messaging Server has an installed SSL certificate (see “Obtaining Certificates” on page 290), it can support same-port IMAP over SSL.

HTTP Over SSL

You can accept the default HTTP over SSL port number (443) or you can specify a separate port for HTTP.

Service Banner

When a client first connects to the Messaging Server POP or IMAP port, the server sends an identifying text string to the client. This service banner (not normally displayed to the client’s user) identifies the server as iPlanet Messaging Server, and gives the server’s version number. The banner is most typically used for client debugging or problem-isolation purposes.

You can replace the default banner for the POP or IMAP service if you want a different message sent to connecting clients.

You can use iPlanet Console or the `configutil` utility to set service banners. For detailed syntax information about `configutil`, see the *Messaging Server Reference Manual*.

Login Requirements

You can control how users are permitted to log in to the POP, IMAP, or HTTP service to retrieve mail. You can allow password-based login (for all services), and certificate-based login (for IMAP or HTTP services). This section provides background information; for the steps you follow to make these settings, see "Configuring POP Services" on page 73, "Configuring IMAP Services" on page 74, or "Configuring HTTP Services" on page 76.

Password-Based Login

In typical messaging installations, users access their POP, IMAP, or HTTP mailboxes by entering a password into their mail client. The client sends the password to the server, which uses it to authenticate the user. If the user is authenticated, the server decides, based on access-control rules, whether or not to grant the user access to certain mailboxes stored on that server.

If you allow password login, users can access POP, IMAP, or HTTP by entering a password. (Password-based login is the only authentication method for POP services.) Passwords are stored in an LDAP directory. Directory policies determine what password policies, such as minimum length, are in effect.

If you disallow password login for IMAP or HTTP services, password-based authentication is not permitted. Users are then required to use certificate-based login, as described in the next section.

To increase the security of password transmission for IMAP and HTTP services, you can require that passwords be encrypted before they are sent to your server. You do this by selecting a minimum cipher-length requirement for login.

- If you choose 0, you do not require encryption. Passwords are sent in the clear or they are encrypted, depending on client policy.
- If you choose a nonzero value, the client must establish an SSL session with the server—using a cipher whose key length is at least the value you specify—thus encrypting any IMAP or HTTP user passwords the client sends.

If the client is configured to require encryption with key lengths greater than the maximum your server supports, or if your server is configured to require encryption with key lengths greater than what the client supports, password-based login cannot occur. For information on setting up your server to support various ciphers and key lengths, see “Enabling SSL and Selecting Ciphers” on page 294.

Certificate-Based Login

In addition to password-based authentication, iPlanet servers support the authentication of users through examination of their digital certificates. Instead of presenting a password, the client presents the user’s certificate when it establishes an SSL session with the server. If the certificate is validated, the user is considered authenticated.

For instructions on setting up Messaging Server to accept certificate-based user login to the IMAP or HTTP service, see “Setting Up Certificate-Based Login” on page 296.

You don’t need to uncheck the “Allow password login” box in the IMAP or HTTP System form to enable certificate-based login. If the box is checked (its default state), and if you have performed the tasks required to set up certificate-based login, both password-based and certificate-based login are supported. Then, if the client establishes an SSL session and supplies a certificate, certificate-based login is used. If the client does not use SSL or does not present a client certificate, it will send a password instead.

Performance Parameters

You can set some of the basic performance parameters for the POP, IMAP, and HTTP services of Messaging Server. Based on your hardware capacity and your user base, you can adjust these parameters for maximum efficiency of service. This section provides background information; for the steps you follow to make these settings, see “Configuring POP Services” on page 73, “Configuring IMAP Services” on page 74, or “Configuring HTTP Services” on page 76.

Number of Processes

Messaging Server can divide its work among several executing processes, which in some cases can increase efficiency. This capability is especially useful with multiprocessor server machines, in which adjusting the number of server processes can allow more efficient distribution of multiple tasks among the hardware processors.

There is a performance overhead, however, in allocating tasks among multiple processes and in switching from one process to another. The advantage of having multiple processes diminishes with each new one added. A simple rule of thumb for most configurations is to have one process per hardware processor on your server machine, up to a maximum of perhaps 4 processes. Your optimum configuration may be different; this rule of thumb is meant only as a starting point for your own analyses.

Note: On some platforms you might also want to increase the number of processes to get around certain per-process limits (such as the maximum number of file descriptors), specific to that platform, that may affect performance.

The default number of processes is 1 each for the POP, IMAP, or HTTP service.

Number of Connections per Process

The more simultaneous client connections your POP, IMAP, or HTTP service can maintain, the better it is for clients. If clients are denied service because no connections are available, they must then wait until another client disconnects.

On the other hand, each open connection consumes memory resources and makes demands on the I/O subsystem of your server machine, so there is a practical limit to the number of simultaneous sessions you can expect the server to support. (You might be able to increase that limit by increasing server memory or I/O capacity.)

IMAP, HTTP, and POP have different needs in this regard:

- IMAP connections are generally long-lived compared to POP and HTTP connections. When a user connects to IMAP to download messages, the connection is usually maintained until the user quits or the connection times out. In contrast, a POP or HTTP connection is usually closed as soon as the POP or HTTP request has been serviced.
- IMAP and HTTP connections are generally very efficient compared to POP connections. Each POP reconnection requires re-authentication of the user. In contrast, an IMAP connection requires only a single authentication because the connection remains open for the duration of the IMAP session (login to

logout). An HTTP connection is short, but the user need not reauthenticate for each connection because multiple connections are allowed for each HTTP session (login to logout). POP connections, therefore, involve much greater performance overhead than IMAP or HTTP connections. iPlanet Messaging Server, in particular, has been designed to require very low overhead by open but idle IMAP connections and by multiple HTTP connections.

NOTE For more information about HTTP session security, see "About HTTP Security" on page 283.

Thus, at a given moment for a given user demand, Messaging Server may be able to support many more open IMAP or HTTP connections than POP connections.

The default value for IMAP is 4000; the default value for HTTP is 6000 connections per process; the default value for POP is 600. These values represent roughly equivalent demands that can be handled by a typically configured server machine. Your optimum configuration may be different; these defaults are meant only as general guidelines.

Number of Threads per Process

Besides supporting multiple processes, Messaging Server further improves performance by subdividing its work among multiple threads. The server's use of threads greatly increases execution efficiency, because commands in progress are not holding up the execution of other commands. Threads are created and destroyed, as needed during execution, up to the maximum number you have set.

Having more simultaneously executing threads means that more client requests can be handled without delay, so that a greater number of clients can be serviced quickly. However, there is a performance overhead to dispatching among threads, so there is a practical limit to the number of threads the server can make use of.

For POP, IMAP, and HTTP, the default maximum value is 250 threads per process. The numbers are equal despite the fact that the default number of connections for IMAP and HTTP is greater than for POP. It is assumed that the more numerous IMAP and HTTP connections can be handled efficiently with the same maximum number of threads as the fewer, but busier, POP connections. Your optimum configuration may be different, but these defaults are high enough that it is unlikely you would ever need to increase them; the defaults should provide reasonable performance for most installations.

Dropping Idle Connections

To reclaim system resources used by connections from unresponsive clients, the IMAP4, POP3, and HTTP protocols permit the server to unilaterally drop connections that have been idle for a certain amount of time.

The respective protocol specifications require the server to keep an idle connection open for a minimum amount of time. The default times are 10 minutes for POP, 30 minutes for IMAP, 3 minutes for HTTP. You can increase the idle times beyond the default values, but you cannot make them less.

If a POP or IMAP connection is dropped, the user must reauthenticate to establish a new connection. In contrast, if an HTTP connection is dropped, the user need not reauthenticate because the HTTP session remains open. For more information about HTTP session security, see “About HTTP Security” on page 283.

Idle POP connections are usually caused by some problem (such as a crash or hang) that makes the client unresponsive. Idle IMAP connections, on the other hand, are a normal occurrence. To keep IMAP users from being disconnected unilaterally, IMAP clients typically send a command to the IMAP server at some regular interval that is less than 30 minutes.

Logging Out HTTP Clients

An HTTP session can persist across multiple connections. HTTP clients are not logged out when a connection is dropped. However, if an HTTP session remains idle for a specified time period, the server will automatically drop the HTTP session and the client is logged out (the default time period is 2 hours). When the session is dropped, the client’s session ID becomes invalid and the client must reauthenticate to establish another session. For more information about HTTP security and session ID’s, see “About HTTP Security” on page 283.

Client Access Controls

iPlanet Messaging Server includes access-control features that allow you to determine which clients can gain access to its POP, IMAP, or HTTP messaging services (and SMTP as well). You can create flexible access filters that allow or deny access to clients based on a variety of criteria.

Client access control is an important security feature of iPlanet Messaging Server. For information on creating client access-control filters and examples of their use, see “Configuring Client Access to POP, IMAP, and HTTP Services” on page 301 and “Configuring Client Access to SMTP Services” on page 311.

Configuring POP Services

You can perform basic configuration of the Messaging Server POP service by using the `configutil` command or by using iPlanet Console.

For more information, see also:

- “Enabling and Disabling Services” on page 66
- “Specifying Port Numbers” on page 66
- “Number of Connections per Process” on page 70
- “Dropping Idle Connections” on page 72
- “Number of Threads per Process” on page 71
- “Number of Processes” on page 70

Console. To configure the POP service using Console:

1. From iPlanet Console, open the Messaging Server you want to configure.
2. Click the Configuration tab and open the Services folder in the left pane.
3. Select POP.
4. Click the System tab in the right pane.
5. To enable the service, check the box labeled “Enable POP service at port” and assign a port number.
6. Specify connection settings as follows:
   - Set the maximum number of network connections per process. For more information, see “Number of Connections per Process” on page 70.
   - Set the maximum idle time for connections. For more information, see “Dropping Idle Connections” on page 72.
7. Specify process settings as follows:
   - Set the maximum number of threads per process. For more information, see “Number of Threads per Process” on page 71.

- Set the maximum number of processes. For more information, see “Number of Processes” on page 70.

8. If desired, in the POP service banner field, specify a service banner.
9. Click Save.

NOTE	For the POP service, password-based login is automatically enabled.

Command Line. You can set values for POP attributes at the command line as follows:

To enable or disable the POP service:

```
configutil -o service.pop.enable -v [ yes | no ]
```

To specify the port number:

```
configutil -o service.pop.port -v number
```

To set the maximum number of network connections per process:

```
configutil -o service.pop.maxsessions -v number
```

To set the maximum idle time for connections:

```
configutil -o service.pop.idletimeout -v number
```

To set the maximum number of threads per process:

```
configutil -o service.pop.maxthreads -v number
```

To set the maximum number of processes:

```
configutil -o service.pop.numprocesses -v number
```

To specify a protocol welcome banner:

```
configutil -o service.pop.banner -v banner
```

Configuring IMAP Services

You can perform basic configuration of the Messaging Server IMAP service by using the `configutil` command or by using iPlanet Console. For more information, see also:

- “Enabling and Disabling Services” on page 66

- “Specifying Port Numbers” on page 66
- “Password-Based Login” on page 68
- “Number of Connections per Process” on page 70
- “Dropping Idle Connections” on page 72
- “Number of Threads per Process” on page 71
- “Number of Processes” on page 70

Console. To configure the IMAP service from the Console:

1. From iPlanet Console, open the Messaging Server you want to configure.
2. Click the Configuration tab and open the Services folder in the left pane.
3. Select IMAP.
4. Click the System tab in the right pane.
5. To enable the service, check the box labeled “Enable IMAP service at port” and assign a port number.
6. If desired, enable password-based login.
7. Specify connection settings as follows:
    - Set the maximum number of network connections per process. For more information, see “Number of Connections per Process” on page 70.
    - Set the maximum idle time for connections. For more information, see “Dropping Idle Connections” on page 72.
8. Specify process settings as follows:
    - Set the maximum number of threads per process. For more information, see “Number of Threads per Process” on page 71.
    - Set the maximum number of processes. For more information, see “Number of Processes” on page 70.
9. If desired, in the IMAP service banner field, specify a service banner.
10. Click Save.

Command Line. You can set values for the IMAP attributes at the command line as follows:

To enable or disable the IMAP service:

```
configutil -o service.imap.enable -v [ yes | no ]
```

To specify the port number:

```
configutil -o service.imap.port -v number
```

To enable a separate port for IMAP over SSL:

```
configutil -o service.imap.enablesslport -v [ yes | no ]
```

To specify a port number for IMAP over SSL:

```
configutil -o service.imap.sslport -v number
```

To enable or disable password login to the IMAP service:

```
configutil -o service.http.plaintextmincipher -v value
```

where *value* is one of the following:

```
 -1 - Disables password login
  0 - Enables password login without encryption
 40 - Enables password login and specifies an encryption strength
128 - Enables password login and specifies an encryption strength
```

To set the maximum number of network connections per process:

```
configutil -o service.imap.maxsessions -v number
```

To set the maximum idle time for connections:

```
configutil -o service.imap.idletimeout -v number
```

To set the maximum number of threads per process:

```
configutil -o service.imap.maxthreads -v number
```

To set the maximum number of processes:

```
configutil -o service.imap.numprocesses -v number
```

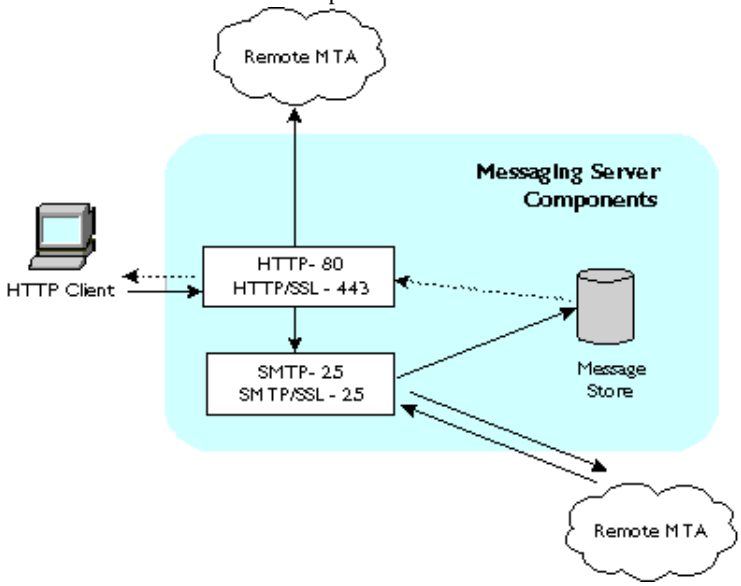
To specify a protocol welcome banner:

```
configutil -o service.imap.banner -v banner
```

Configuring HTTP Services

POP and IMAP clients send mail directly to the iPlanet Messaging Server MTA for routing or delivery. In contrast, HTTP clients send mail to a specialized web server that is part of iPlanet Messaging Server. The HTTP service then sends the message to the local MTA or to a remote MTA for routing or delivery, as shown in Figure 4-1.

Figure 4-1 HTTP Service Components



Many of the HTTP configuration parameters are similar to the parameters available for the POP and IMAP services. These include parameters for connection settings and process settings. For more information, see also:

- “Enabling and Disabling Services” on page 66
- “Specifying Port Numbers” on page 66
- “Password-Based Login” on page 68
- “Number of Connections per Process” on page 70
- “Dropping Idle Connections” on page 72
- “Logging Out HTTP Clients” on page 72
- “Number of Threads per Process” on page 71
- “Number of Processes” on page 70

Some parameters are specific to the HTTP service; these include parameters for message settings and MTA settings.

Message Settings. When an HTTP client constructs a message with attachments, the attachments are uploaded to the server and stored in a file. The HTTP service retrieves the attachments and constructs the message before sending the message to an MTA for routing or delivery. You can accept the default attachment spool directory or specify an alternate directory. You can also specify a maximum size allowed for attachments.

MTA Settings. By default, the HTTP service sends outgoing web mail to the local MTA for routing or delivery. You might want to configure the HTTP service to send mail to a remote MTA, for example, if your site is a hosting service and most recipients are not in the same domain as the local host machine. To send web mail to a remote MTA, you need to specify the remote host name and the SMTP port number for the remote host.

Console. To configure your HTTP service by using iPlanet Console:

1. From iPlanet Console, open the Messaging Server you want to configure.
2. Click the Configuration tab and open the Services folder in the left pane.
3. Select HTTP.
4. Click the System tab in the right pane.
5. To enable the service, check the box labeled “Enable HTTP service at port” and assign a port number.
6. If desired, enable password-based login.
7. Specify connection settings as follows:
   - Set the maximum number of network connections per process. For more information, see “Number of Connections per Process” on page 70.
   - Set the maximum idle time for connections. For more information, see “Dropping Idle Connections” on page 72.
   - Set the maximum idle time for client sessions. For more information, see “Logging Out HTTP Clients” on page 72.
8. Specify process settings as follows:
   - Set the maximum number of threads per process. For more information, see “Number of Threads per Process” on page 71.
   - Set the maximum number of processes. For more information, see “Number of Processes” on page 70.
9. Specify Message settings as follows:

- If desired, specify the attachment spool directory.
- If desired, specify the maximum outgoing mail size. Note that this includes all the attachments encoded in base64, and that base64 encoding requires an extra 33% more space. Thus a 5 megabyte limit in the console results in the maximum size of one message and attachements being about 3.75M.

For more information, see “Message Settings” on page 78.

10. Specify MTA settings as follows:
    - If desired, specify an alternate MTA host name.
    - If required, specify an alternate MTA port.

    For more information, see “MTA Settings” on page 78.
11. Click Save.

Command Line. You can set values for the HTTP attributes at the command line as follows:

To enable or disable the HTTP service:

```
configutil -o service.http.enable -v [ yes | no ]
```

To specify the port number:

```
configutil -o service.http.port -v number
```

To enable a separate port for HTTP over SSL:

```
configutil -o service.http.enablesslport -v [ yes | no ]
```

To specify a port number for HTTP over SSL:

```
configutil -o service.http.sslport -v number
```

To enable or disable password login:

```
configutil -o service.http.plaintextmincipher -v value
```

where *value* is one of the following:

```
 -1 - Disables password login
  0 - Enables password login without encryption
 40 - Enables password login and specifies an encryption strength
128 - Enables password login and specifies an encryption strength
```

To set the maximum number of network connections per process:

```
configutil -o service.http.maxsessions -v number
```

To set the maximum idle time for connections:

```
configutil -o service.http.idletimeout -v number
```

To set the maximum idle time for client sessions:

```
configutil -o service.http.sessiontimeout -v number
```

To set the maximum number of threads per process:

```
configutil -o service.http.maxthreads -v number
```

To set the maximum number of processes:

```
configutil -o service.http.numprocesses -v number
```

To specify the attachment spool directory for client outgoing mail:

```
configutil -o service.http.spooldir -v dirpath
```

To specify the maximum message size:

```
configutil -o service.http.maxmessagesize -v size
```

where *size* is a number in bytes. Note that this includes all the attachments encoded in base64, and that base64 encoding requires an extra 33% more space. Thus a 5 megabyte limit in the console results in the maximum size of one message and attachements being about 3.75M.

To specify an alternate MTA host name:

```
configutil -o service.http.smtphost -v hostname
```

To specify the port number for the alternate MTA host name:

```
configutil -o service.http.smtpport -v portnum
```

Chapter 5

Messaging Multiplexor

This chapter provides concepts about iPlanet Messaging Multiplexor. This chapter contains the following sections:

- About Messaging Multiplexor
- Configuring Multiplexor
- Starting Multiplexor
- A Sample Topology

For information about installing Multiplexor, see the *Messaging Server Installation Guide*. For details about Multiplexor configuration parameters, see the *Messaging Server Reference Manual*.

About Messaging Multiplexor

The iPlanet Messaging Multiplexor (MMP) is a specialized messaging server that acts as a single point of connection to multiple messaging servers. With the Multiplexor, large-scale messaging-service providers can distribute POP and IMAP user mailboxes across many machines to increase messaging capacity. All users connect to the single Multiplexor server, which redirects each connection to the appropriate messaging server.

If you provide electronic mail service to many users, you can install and configure the MMP so that an entire array of messaging servers will appear to your mail users to be a single host.

The Messaging Multiplexor is provided as part of iPlanet Messaging Server. You can install the MMP at the same time you install the Messaging Server or other iPlanet servers, or you can install the MMP separately at a later time.

The MMP supports:

- Both unencrypted and encrypted (SSL) communications with mail clients.
- Client certificate-based authentication, described in “Certificate-Based Client Authentication” on page 85.
- User pre-authentication, described in “User Pre-Authentication” on page 86.
- Virtual domains that listen on different IP addresses and automatically append domain names to user IDs, described in “Virtual Domains” on page 86.
- Multiple installations of the MMP on different machines (one installation per machine). See the *Messaging Server Installation Guide*.
- Multiple instances of Multiplexor on a server machine, described in “Multiple Multiplexor Instances” on page 87. Multiple instances can be used for alternate configurations such as SSL or the listen port that cannot be handled through virtual domains.
- Enhanced LDAP searching.

Multiplexor Benefits

Message stores on heavily used messaging servers can grow quite large. Spreading user mailboxes and user connections across multiple servers can therefore improve capacity and performance. In addition, it may be more cost-effective to use several small server machines than one large, high-capacity, multiprocessor machine.

If the size of your mail-server installation requires the use of multiple messaging servers, your organization can benefit in several ways from using the MMP. The indirect connection between users and their message stores, coupled with the ease of reconfiguration of user accounts among messaging servers allows for the following benefits:

- **Simplified User Management**

  Because all users connect to one server (or two, if you have separate Multiplexors for POP and IMAP), you can preconfigure email clients and distribute uniform login information to all users. This simplifies your administrative tasks and reduces the possibility of distributing erroneous login information.

  For especially high-load situations, you can run multiple Multiplexor servers and manage connections to them by DNS round robin or by using a load-balancing program, such as LocalDirector from Cisco Systems.

Because the MMP uses information stored in the LDAP directory to locate each user's Messaging Server, moving a user to a new server is simple for the system administrator and transparent to the user. The administrator can move a user's mailbox from one Messaging Server to another, and then update the user's entry in the LDAP directory. The user's mail address, mailbox access, and other client preferences need not change.

- **Improved Performance**

  If a message store grows prohibitively large for a single machine, you can balance the load by moving some of the message store to another machine.

  You can assign different classes of users to different machines. For example, you can choose to locate premium users on a larger and more powerful machine.

  The MMP performs some buffering so that slow client connections (through a modem, for example) do not slow down the Messaging Server.

- **Decreased Cost**

  Because you can efficiently manage multiple Messaging Servers with the MMP, you can decrease overall costs by purchasing several small server machines that together cost less than one very large machine.

- **Better Scalability**

  With the MMP, your configuration can expand easily. You can incrementally add machines as your performance or storage-capacity needs grow, without replacing your existing investment.

- **Minimum User Downtime**

  Using the MMP to spread a large user base over many small store machines isolates user downtime. When an individual server fails, only its users are affected.

- **Increased Security**

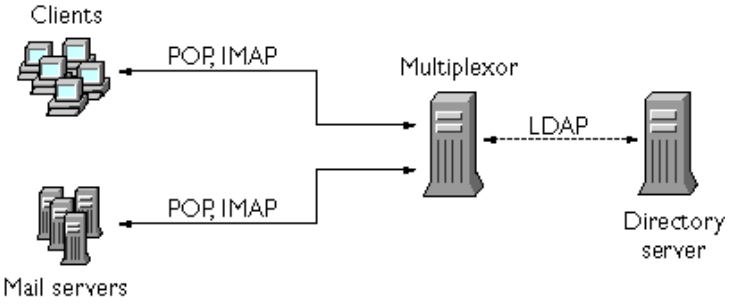
  You can use the server machine on which the MMP is installed as a firewall machine. By routing all client connections through this machine, you can restrict access to the internal message store machines by outside computers. The MMP supports both unencrypted and encrypted communications with clients.

How Multiplexor Works

The iPlanet MMP is a multithreaded server that facilitates distributing mail users across multiple server machines. the MMP handles incoming client connections destined for other server machines (the machines on which user mailboxes reside). Clients connect to the MMP itself, which then redirects the session to the server with the correct mailbox. This capability allows Internet service providers and other large installations to spread message stores across multiple machines (to increase capacity) while providing the appearance of a single mail host for users (to increase efficiency) and for external clients (to increase security).

Figure 5-1 shows how servers and clients relate to each other in an MMP installation.

Figure 5-1 Clients and Servers in an MMP Installation



All POP and IMAP clients work with the Messaging Multiplexor. The MMP accepts connections, performs LDAP directory lookups, and routes the connections appropriately. As is typical with other mail server installations, each user is assigned a specific address and mailbox on a specific Messaging Server. However, all connections are routed through the MMP.

In more detail, these are the steps involved in establishing a user connection:

1. A user's client connects to the MMP, which accepts preliminary authentication information (user name).
2. The MMP queries the Directory Server to determine which Messaging Server contains that user's mailbox.
3. The MMP connects to the proper Messaging Server, replays authentication, then acts as a pass-through pipe for the duration of the connection.

Encryption (SSL) Option

The iPlanet Messaging Multiplexor supports both unencrypted and encrypted (SSL) communications between the Messaging Server(s) and their mail clients.

In SSL mode, the MMP listens by default on port 993. When SSL is enabled, the MMP IMAP supports STARTTLS and the MMP can also be configured to listen on additional ports for SSL IMAP and POP connections.

To enable SSL encryption for your IMAP and POP services, edit the `ImapProxyAService.cfg` and `PopProxyAService.cfg` files, respectively. You must also edit the `default:ServiceList` option in the `AService.cfg` file to include the list of all IMAP and POP server ports regardless of whether or not they are secure.

By default, SSL is not enabled since the SSL configuration parameters are commented out. To enable SSL, you must un-comment and set the SSL parameters. For a list of the SSL parameters, see the *Messaging Server Reference Manual*.

Certificate-Based Client Authentication

The MMP can use `certmap` to match a client's certificate to the correct user in the Users / Groups Directory Server.

In order to use certificate-based client authentication, you must also enable SSL encryption as described in "Encryption (SSL) Option" on page 85.

You also have to configure a store administrator. You can use the mail administrator, but it is recommended that you create a unique user ID, such as `mmpstore` for this purpose so that you can set permissions as needed.

Note that the MMP does not support `certmap` plug-ins. Instead, the MMP accepts enhanced `DNComps` and `FilterComps` property value entries in the `certmap.conf` file. These enhanced format entries use the form:

mapname`:DNComps` *FROMATTR=TOATTR*
mapname`:FilterComps` *FROMATTR=TOATTR*

So that a *FROMATTR* value in a certificate's subjectDN can be used to form an LDAP query with the *TOATTR=value* element. For example, a certificate with a subjectDN of "`cn=Pilar Lorca, ou=pilar o=siroe.com`" could be mapped to an LDAP query of "`(uid=pilar)`" with the line:

mapname`:FilterComps ou=uid`

To enable certificate-based authentication for your IMAP service:

1. Decide on the user ID you intend to use as store administrator.

   While you can use the mail administrator for this purpose, it is recommended that you create a unique user ID for store administrator (for example, `mmpstore`).

2. Make sure that SSL encryption is (or will be) enabled as described in "Encryption (SSL) Option" on page 85.

3. Configure the MMP to use certificate-based client authentication by specifying the location of the `certmap.conf` file in your configuration files.

User Pre-Authentication

The MMP provides you with the option of pre-authenticating users by binding to the directory as the incoming user and logging the result.

NOTE Enabling user pre-authentication will reduce server performance

The log entries are in the format:

date time `(sid 0x%p) user` *name* `pre-authenticated - client IP` *address*

Where *date* is in the format `yyyymmdd`, *time* is in the format `hhmmss`, `sid` is the session object, the `user` *name* includes the virtual domain (if any), and the IP address is in dot-quad format.

Virtual Domains

Virtual domains listen on different IP addresses and automatically append domain names to user IDs. They can also be used to specify alternate configurations.

The MMP can map IP addresses to domain names for searching an LDAP directory and for logging in to the store server. When a connection is accepted from a client, if the server's IP address is in the virtual domain mapping file, the domain is appended to the user ID and used for the LDAP search and for subsequent replay of authentication. This capability is useful for hosting multiple domains with overlapping user ID name spaces.

To enable virtual domains, edit the `ImapProxyAService.cfg` and/or `PopProxyAService.cfg` file(s) in the instance directory to point to the virtual domain mapping file.

Each entry of a virtual domain file has the following syntax:

`vdmap` *name IPaddr*
name:parameter value

Where *name* is whatever name you choose to use, *IPaddr* is in dot-quad format, and *parameter* and *value* pairs configure the virtual domain. When set, virtual domain configuration parameter values override global configuration parameter values.

Listed below are the configuration parameters you can specify for a virtual domain:

`AuthCacheSize` and `AuthCacheSizeTTL`
`AuthService`
`BindDN` and `BindPass`
`CanonicalVirtual`
`CertMap`
`CRAMs`
`DomainDelim`
`HostedDomains`
`LdapCacheSize` and `LdapCacheTTL`
`LdapURL`
`MailHostAttrs`
`PreAuth`
`ReplayFormat`
`StoreAdmin` and `StoreAdminPass`
`SearchFormat`
`TCPAccess`
`VDomain`

For detailed descriptions of these configuration parameters, see the *Messaging Server Reference Manual*.

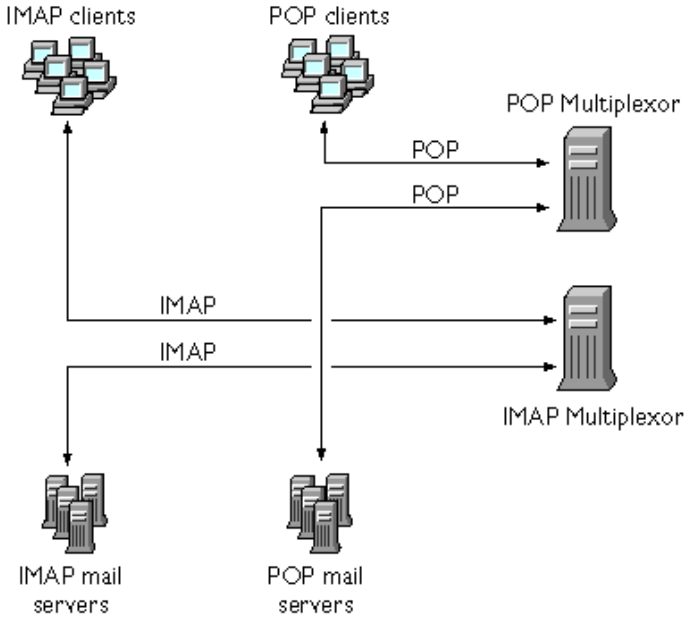
Multiple Multiplexor Instances

You can create multiple instances of the MMP, all of which must be on the same server. In other words, you can have multiple installations of the MMP on different servers, and on any given machine you can have multiple instances.

Using multiple instances of the MMP allows you to create alternate configurations, such as SSL or the listen port, that cannot be handled through virtual domains.

You can configure a single instance of the MMP to support both POP and IMAP protocols (as shown in Figure 5-1), or you can create separate MMP instances for each protocol, as shown in Figure 5-2. By splitting messaging services across different machines, you can tune the resources on each computer for maximum performance.

Figure 5-2 Separate MMP Instances for Each Protocol



For instructions on creating multiple instances of the MMP, see the *Messaging Server Installation Guide*.

Configuring Multiplexor

To configure the Multiplexor, you must manually edit the configuration parameters in the Multiplexor configuration files listed in Table 5-1.

Table 5-1 Messaging Multiplexor Configuration Files

File	Description
`PopProxyAService.cfg`	Configuration file specifying environment variables used for POP services.
`ImapProxyAService.cfg`	Configuration file specifying environment variables used for IMAP services.
`AService.cfg`	Configuration file specifying which services to start and a few options shared by both POP and IMAP services.

The Multiplexor configuration files are stored in the *server-root*`/mmp-`*hostname* directory, where *server-root* is the directory where you installed the Messaging Server and `mmp-`*hostname* is the subdirectory named after the MMP instance. For example, if you installed the MMP on a machine named `tarpit` and accepted the default installation location, the configuration files would be located in `/usr/iplanet/server5/mmp-tarpit`.

As an example, the `LogDir` and `LogLevel` parameters can be found in all three configuration files. In `ImapProxyAService.cfg`, they are used to specify logging parameters for IMAP-related events; similarly, these parameters in `PopProxyAService.cfg` are used to configure logging paraIf the stored -d option is unable to make the database consistent, you should perform the following steps in the order indicated:

Shut down all servers.

Remove all files in server-root / msg-instance / store / mboxlist.

Restart the server processes.

Run reconstruct -m to build a new mailboxes database from the contents of the spool area.meters for POP-related events. In `AService.cfg`, however, `LogDir` and `LogLevel` are used for logging MMP-wide failures, such as the failure to start a POP or IMAP service.

For a complete description of all MMP configuration parameters, see the *Messaging Server Reference Manual*.

Starting Multiplexor

UNIX Systems

To start an instance of the Messaging Multiplexor in a UNIX system, run the `AService.rc` script in the *server-root*`/mmp-`*hostname* directory as follows:

`./AService.rc [`*options*`]`

Optional parameters for the `AService.rc` script are described in Table 5-2.

Table 5-2 Optional Parameters for the `AService.rc` Script

Option	Description
`start`	Start the MMP (even if one is already running).
`stop`	Stop the most recently started MMP.
`restart`	Stop the most recently started MMP, then start an MMP.
`reload`	Causes an MMP that is already running to reload its configuration without disrupting any active connections.

Windows NT Systems

To start an instance of the Messaging Multiplexor in a Windows NT, go to Services in the Windows NT Control Panel and click on "Start." You can also click on "Stop" to stop the MMP. The service options are described below in Table 5-3. Windows

Table 5-3 Windows NT MMP Service Options

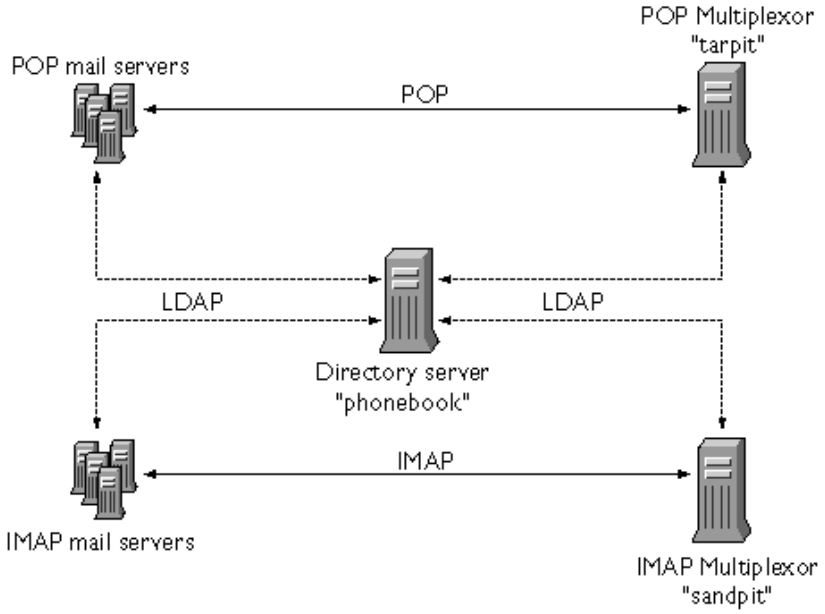
Option	Description
`start`	At the Control Panel, start the MMP (even if one is already running), or, at the command line run the command `AService.exe start`
`stop`	At the Control Panel, stop the most recently started MMP, or, at the command line run the command `AService.exe stop`
`restart`	To restart on Windows NT, stop the most recently started MMP and then start an MMP.
`reload`	To reload the MMP, go to the mmp-instance directory and at the command line run the command `AService.exe refresh`

A Sample Topology

The fictional Siroe Corporation has two Multiplexors on separate machines, each supporting several Messaging Servers. POP and IMAP user mailboxes are split across the Messaging Server machines, with each server dedicated exclusively to POP or exclusively to IMAP. (You can restrict client access to POP services alone by removing the IMAP-server binary; likewise, you can restrict client access to IMAP services alone by removing the POP-server binary.) Each Multiplexor also supports only POP or only IMAP. The LDAP directory service is on a separate, dedicated machine.

This topology is illustrated below in Figure 5-3.

Figure 5-3 Multiple MMPs Supporting Multiple Messaging Servers



IMAP Configuration Example

The IMAP Multiplexor in Figure 5-3 is installed on `sandpit`, a machine with two processors. This Multiplexor is listening to the standard port for IMAP connections (143). Multiplexor communicates with the LDAP server on the host phonebook for user mailbox information, and it routes the connection to the appropriate IMAP server. It overrides the IMAP capability string, provides a virtual domain file, and supports SSL communications.

This is its `ImapProxyAService.cfg` configuration file:

```
default:LdapUrl             ldap://phonebook/o=Siroe.com
default:LogDir              /usr/iplanet/server5/mmp-sandpit/log
default:LogLevel            5
default:BindDN              “cn=Directory Manager"
default:BindPass            secret
default:BacksidePort        143
default:Timeout             1800
default:Capability          "IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE
UIDPLUS CHILDREN LANGUAGE XSENDER X-NETSCAPE XSERVERINFO AUTH=PLAIN"
default:SearchFormat        (uid=%s)
default:SSLEnable           yes
default:SSLPorts            993
default:SSLSecmodFile       /usr/iplanet/server5/mmp-sandpit/secmod.db
default:SSLCertFile         /usr/iplanet/server5/mmp-sandpit/cert7.db
default:SSLKeyFile          /usr/iplanet/server5/mmp-sandpit/key3.db
default:SSLKeyPasswdFile    “”
default:SSLCipherSpecs      all
default:SSLCertNicknames    Siroe.com Server-Cert
default:SSLCacheDir         /usr/iplanet/server5/mmp-sandpit
default:SSLBacksidePort     993
default:VirtualDomainFile   /usr/iplanet/server5/mmp-sandpit/vdmap.cfg
default:VirtualDomainDelim  @
default:ServerDownAlert     "your IMAP server appears to be temporarily out of
service"
default:MailHostAttrs       mailHost
default:PreAuth             no
default:CRAMs               no
default:AuthCacheSize       10000
default:AuthCacheTTL        900
default:AuthService         no
default:AuthServiceTTL      0
default:BGMax               10000
default:BGPenalty           2
default:BGMaxBadness        60
default:BGDecay             900
default:BGLinear            no
default:BGExcluded          /usr/iplanet/server5/mmp-sandpit/bgexcl.cfg
default:ConnLimits          0.0.0.0|0.0.0.0:20
default:LdapCacheSize       10000
default:LdapCacheTTL        900
default:HostedDomains       yes
default:DefaultDomain       Siroe.com
```

POP Configuration Example

The POP Multiplexor example in Figure 5-3 is installed on `tarpit`, a machine with four processors. This Multiplexor is listening to the standard port for POP connections (110). Multiplexor communicates with the LDAP server on the host phonebook for user mailbox information, and it routes the connection to the appropriate POP server. It also provides a spoof message file.

This is its `PopProxyAService.cfg` configuration file:

```
default:LdapUrl             ldap://phonebook/o=Siroe.com
default:LogDir              /usr/iplanet/server5/mmp-tarpit/log
default:LogLevel            5
default:BindDN              “cn=Directory Manager"
default:BindPass            password
default:BacksidePort        110
default:Timeout             1800
default:Capability          "IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE
UIDPLUS CHILDREN LANGUAGE XSENDER X-NETSCAPE XSERVERINFO AUTH=PLAIN"
default:SearchFormat        (uid=%s)
default:SSLEnable           no
default:VirtualDomainFile   /usr/iplanet/server5/mmp-tarpit/vdmap.cfg
default:VirtualDomainDelim  @
default:MailHostAttrs       mailHost
default:PreAuth             no
default:CRAMs               no
default:AuthCacheSize       10000
default:AuthCacheTTL        900
default:AuthService         no
default:AuthServiceTTL      0
default:BGMax               10000
default:BGPenalty           2
default:BGMaxBadness        60
default:BGDecay             900
default:BGLinear            no
default:BGExcluded          /usr/iplanet/server5/mmp-tarpit/bgexcl.cfg
default:ConnLimits          0.0.0.0|0.0.0.0:20
default:LdapCacheSize       10000
default:LdapCacheTTL        900
default:HostedDomains       yes
default:DefaultDomain       Siroe.com
```

Chapter 6

About MTA Services and Configuration

This chapter provides concepts about configuring MTA services for your server. This chapter contains the following sections:

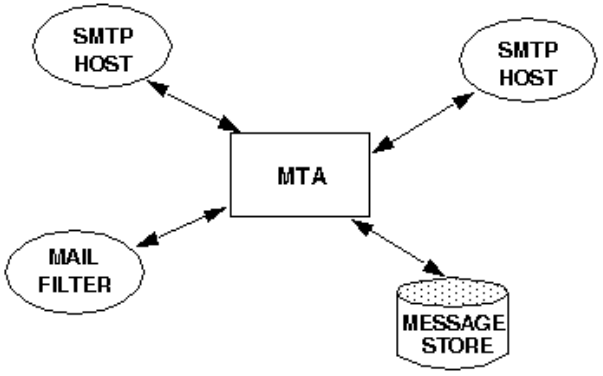
- The Message Transfer Agent (MTA)
- Channels
- Rewrite Rules
- The Job Controller
- The Dispatcher
- The MTA Configuration File
- Other MTA Configuration Files
- Aliases
- Command Line Utilities
- The MTA Directory Cache
- SMTP Security and Access Control
- Log Files

The Message Transfer Agent (MTA)

The Messaging Server Message Transfer Agent (MTA) is part of a store and forward messaging system based on Internet standards. The MTA determines how to route messages for ultimate delivery to the appropriate recipient. As shown in Figure 6-1, the MTA can:

- Route messages to another SMTP host
- Deliver messages to the local message store
- Deliver messages to a program for processing (such as mail filtering)

Figure 6-1 MTA Routing and Delivery



Channels

The channel is the fundamental MTA component that processes a message. A channel represents a connection with another computer system or group of systems. The actual hardware connection or software transport or both may vary widely from one channel to the next.

Channels perform the following functions:

- Transmit messages to remote systems, deleting them from their queue after they are sent.
- Accept messages from remote systems, placing them in the appropriate channel queues.

- Deliver messages to the local message store.
- Deliver messages to programs for special processing.

Messages are enqueued by channels on the way into the MTA and dequeued on the way out. Typically, a message enters via one channel and leaves by another. A channel might dequeue a message, process the message, or enqueue the message to another MTA channel.

You define channels in the primary MTA configuration file, `imta.cnf`. You can also set global options for channels in the MTA option file, `option.dat`, or set options for a specific channel in a channel option file.

For more information about the MTA configuration file, see “The MTA Configuration File,” on page 103. For more information on the option files, see “Option File,” on page 109, and “TCP/IP Channel Option Files,” on page 106. For complete details about configuring channels, see Chapter 8, “Configuring Channel Definitions.”

Master and Slave Programs

Generally (but not always), a channel consists of two programs: master and slave. A channel program that initiates a transfer to a remote system on its own is called a “master” program, while a program that accepts transfers initiated by a remote system is called a “slave” program.

For example, an SMTP channel has a master program that transmits messages and a slave program that receives messages. These are, respectively, the SMTP client and server.

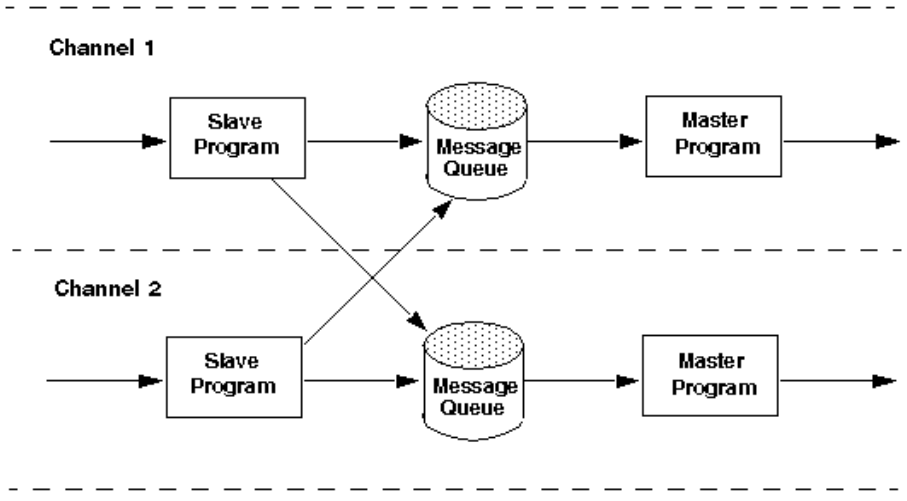
- The master channel program is typically responsible for outgoing connections where the MTA has initiated the operation. The master channel program:
  - Runs in response to a local request for processing.
  - Dequeues the message from the channel message queue.
  - If the destination format is not the same format as the queued message, performs conversion of addresses, headers, and content, as necessary.
  - Initiates network transport of the message.

- The slave channel program typically accepts incoming connections where the MTA is responding to an external request. The slave channel program:
  - Runs in response to an external event or upon local demand.
  - Enqueues a message to a channel. The target channel is determined by passing envelope addresses through a rewrite rule. (Rewrite rules are described in further detail later in this chapter.)

For example, Figure 6-2 shows two channel programs, Channel 1 and Channel 2. Assume the slave program in Channel 1 receives a message from a remote system. The slave program looks at the address, applies rewrite rules as necessary, then based on the rewritten address, enqueues the message to the appropriate channel message queue.

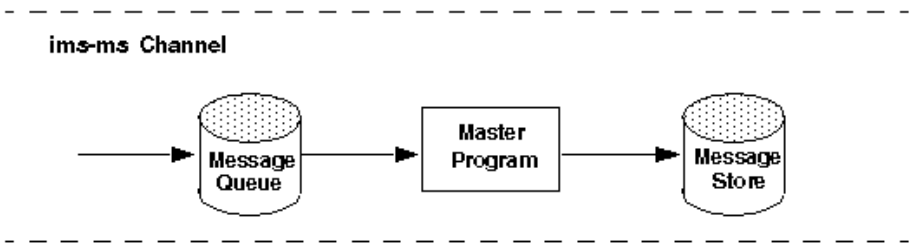
The slave program can enqueue the message to its own message queue or to a message queue belonging to another channel. The master program dequeues the message from the queue and initiates network transport of the message. Note that the master program can only dequeue messages from its own channel queue.

Figure 6-2 Master and Slave Programs



Although a typical channel has both a master and a slave program, it is possible for a channel to contain only a slave program *or* a master program. For example, the `ims-ms` channel supplied with Messaging Server contains only a master program because this channel is responsible only for dequeueing messages to the local message store, as shown in Figure 6-3.

Figure 6-3 ims-ms Channel



Channel Message Queues

All channels have an associated message queue. When a message enters the messaging system, a slave program determines to which message queue the message is enqueued. The enqueued messages are stored in message files in the channel queue directories. By default, these directories are stored at the following location: `/`*server-instance*`/imta/queue/`*channel*`/*`.

Rewrite Rules

Rewrite rules determine the following:

- How to rewrite addresses into their proper or desired format.
- To which channel the message should be enqueued after the address is rewritten.

Each rewrite rule consists of a pattern, which indicates the string to search for in the domain name of an address, and a template, which indicates how the address should be rewritten based on the pattern it matches. After the address is rewritten, the message is enqueued to the destination channel for delivery to the intended recipient.

For more information about configuring rewrite rules, see “The MTA Configuration File,” on page 103 and Chapter 7, “Configuring Rewrite Rules.”

The Job Controller

The Job Controller creates and manages channel jobs for delivering messages. These channel jobs run inside processing pools within the Job Controller.

Each time a message is enqueued to a channel, the Job Controller ensures that there is a job running to deliver the message. This might involve starting a new job process, adding a thread, or simply noting that a job is already running. If a job cannot be started because the job limit for the channel or pool has been reached, the Job Controller waits until another job has exited, then, when the job limit is no longer exceeded, starts another job.

At startup, the Job Controller reads a configuration file that specifies general parameters, return job scheduling, purge job scheduling, pool definitions, and channel processing information. This configuration information is specified in the file `job_controller.cnf` in the *server_root*`/msg-`*instance*`/imta/config/` directory.

For information about configuring the Job Controller, see “Job Controller File,” on page 110 and “Configuring Message Processing and Delivery,” on page 175.

To start the Job Controller, execute the command:

```
imsimta start job_controller
```

To shut down the Job Controller, execute the command:

```
imsimta stop job_controller
```

To restart the Job Controller, execute the command:

```
imsimta restart job_controller
```

Restarting the Job Controller has the effect of shutting down the currently running Job Controller, then immediately starting a new one.

The Dispatcher

The Dispatcher is a multithreaded connection dispatching agent that permits multiple multithreaded servers to share responsibility for a given service. When using the Dispatcher, it is possible to have several multithreaded SMTP server processes running concurrently. In addition, each server may have one or more active connections.

The Dispatcher acts as a central receiver for the TCP ports listed in its configuration. For each defined service, the Dispatcher may create one or more SMTP server processes to handle the connections after they are established.

In general, when the Dispatcher receives a connection for a defined TCP port, it checks its pool of available worker processes for the service on that port and chooses the best candidate for the new connection. If no suitable candidate is available and the configuration permits it, the Dispatcher may create a new worker process to handle this and subsequent connections. The Dispatcher may also create a new worker process in expectation of future incoming connections. There are several configuration options which may be used to tune the Dispatcher's control of its various services, and in particular, to control the number of worker processes and the number of connections each worker process handles.

Creation and Expiration of Server Processes

Automatic housekeeping facilities within the Dispatcher control the creation of new and expiration of old or idle server processes. The basic options that control the Dispatcher's behavior are `MIN_PROCS` and `MAX_PROCS`. `MIN_PROCS` provides a guaranteed level of service by having a number of server processes ready and waiting for incoming connections. `MAX_PROCS`, on the other hand, sets an upper limit on how many server processes may be concurrently active for the given service.

It is possible that a currently running server process might not be able to accept any connections because it is already handling the maximum number of connections of which it is capable, or because the process has been scheduled for termination. The Dispatcher may create additional processes to assist with future connections.

The `MIN_CONNS` and `MAX_CONNS` options provide a mechanism to help you distribute the connections among your server processes. `MIN_CONNS` specifies the number of connections that flags a server process as “busy enough” while `MAX_CONNS` specifies the “busiest” that a server process can be.

In general, the Dispatcher creates a new server process when the current number of server processes is less than `MIN_PROCS` or when all existing server processes are "busy enough" (the number of currently active connections each has is at least `MIN_CONNS`).

If a server process is killed unexpectedly, for example, by the UNIX system `kill` command, the Dispatcher still creates new server processes as new connections come in.

For information about configuring the Dispatcher, see "Dispatcher Configuration File," on page 107.

Controlling the Dispatcher

The Dispatcher is a single resident process that starts and shuts down server processes for various services, as needed.

To start the Dispatcher, execute the command:

```
imsimta start dispatcher
```

This command subsumes and makes obsolete any other `imsimta start` command that was used previously to start up a component of the MTA that the Dispatcher has been configured to manage. Specifically, you should no longer use `imsimta start smtp`. An attempt to execute any of the obsoleted commands causes the MTA to issue a warning.

To shut down the Dispatcher, execute the command:

```
imsimta stop dispatcher
```

What happens with the server processes when the Dispatcher is shut down depends upon the underlying TCP/IP package. If you modify your MTA configuration or options that apply to the Dispatcher, you must restart the Dispatcher so that the new configuration or options take effect.

To restart the Dispatcher, execute the command:

```
imsimta restart dispatcher
```

Restarting the Dispatcher has the effect of shutting down the currently running Dispatcher, then immediately starting a new one.

The MTA Configuration File

The primary MTA configuration file is named `imta.cnf`. By default, this file is found in the following location: *server-instance*`/imta/config/imta.cnf`. This file contains all channel definitions for your server as well as the rewrite rules that determine how addresses are rewritten for routing. The channel associated with a rewritten destination address becomes the destination channel.

This section provides a brief introduction to the MTA configuration file. For details about configuring the rewrite rules and channel definitions that make up the MTA configuration file, see Chapter 7, “Configuring Rewrite Rules,” and Chapter 8, “Configuring Channel Definitions.”

By modifying the MTA configuration file, you establish the channels in use at a site and establish which channels are responsible for which sorts of addresses via rewrite rules. The configuration file establishes the layout of the email system by specifying the transport methods available (channels) and the transport routes (rewrite rules) associating types of addresses with appropriate channels.

The following example of an `imta.cnf` configuration file shows how rewrite rules are used to route messages to the proper channel. No domain names are used to keep things as simple as possible. The rewrite rules appear in the upper half of the configuration file followed by the channel definitions in the lower half of the configuration file.

Figure 6-4 Simple MTA Configuration File

```
! test.cnf - An example configuration file. (1)
!
! This is only an example of a configuration file. It serves
! no useful purpose and should not be used in a real system.
!
a     $U@a-daemon (2)
b     $U@b-daemon
c     $U%c@b-daemon
d     $U%d@a-daemon
              (3)
l             (4)
local-host

a_channel defragment charset7 usascii (5)
a-daemon

b_channel noreverse notices 1 2 3
b-daemon
```

The key items (labeled with boldface numbers, enclosed in parentheses) in the preceding configuration file are explained in the following list:

1. Exclamation points (!) are used to include comment lines. The exclamation point must appear in the first column. An exclamation point appearing anywhere else is interpreted as a *literal* exclamation point.
2. The rewrite rules appear in the first half of the configuration file. No blank lines can appear among the lines of rewrite rules. Lines with comments (beginning with an exclamation point in the first column) are permitted.
3. The first blank line to appear in the file signifies the end of the rewrite rules section and the start of the channel blocks.
4. On UNIX, the first channel block to appear is always the `l` channel (the UNIX local channel, designated with the lowercase letter “l”). Blank lines then separate each channel block from one another. (An exception is the `defaults` channel, which can appear before the `l` channel).

Table 6-1 lists the routing and queuing of messages by the preceding configuration.

Table 6-1 Addresses and Associated Channels

Address	Queued to channel
u@a	a_channel
u@b	b_channel
u@c	b_channel
u@d	a_channel

Other MTA Configuration Files

In addition to the `imta.cnf` file, iPlanet Messaging Server provides several other configuration files to help you configure MTA services. These files are summarized in Table 6-2.

Table 6-2 MTA Configuration Files

File	Description
Autoreply Option File	Options used by the `autoreply` program. /*server-instance*/imta/config/autoreply_option
Alias File (mandatory)	Implements aliases not present in the directory. /*server-instance*/imta/config/aliases
TCP/IP Channel Option Files	Sets channel-specific options. /*server-instance*/imta/config/*channel*_option
Conversion File	Used by the conversion channel to control message body part conversions. /*server-instance*/imta/config/conversions
Dirsync Option File (mandatory)	Options used by the `dirsync` program. /*server-instance*/imta/config/dirsync.opt
Dispatcher Configuration File (mandatory)	Configuration file for the Dispatcher. /*server-instance*/imta/config/dispatcher.cnf
MTA Configuration File (mandatory)	Used for address rewriting and routing as well as channel definition. /*server-instance*/imta/config/imta.cnf
Mapping File (mandatory)	Repository of mapping tables. /*server-instance*/imta/config/mappings

Table 6-2 MTA Configuration Files

File	Description
Option File	File of global MTA options. /*server-instance*/imta/config/option.dat
Tailor File (mandatory)	File to specify locations and some tuning parameters. /*server-instance*/imta/config/imta_tailor
Job Controller File (mandatory)	Configuration file used by the Job Controller. /*server-instance*/imta/config/job_controller.cnf

Autoreply Option File

The autoreply option file, autoreply_option, sets options for the autoreply or vacation program. For more information about the syntax of this file, see the *Messaging Server Reference Manual*.

Alias File

The alias file, aliases, sets aliases not set in the directory. In particular, the address for root is a good example. Aliases set in this file will be ignored if the same aliases exist in the directory. For more information about aliases and the aliases file, see “Aliases,” on page 113.

After making changes to the aliases file, you must restart the MTA.

TCP/IP Channel Option Files

TCP / IP channel option files control various characteristics of TCP / IP channels. Channel option files must be stored in the MTA configuration directory and named *x*_option, where *x* is the name of the channel. For example, /*server-instance*/config/imta/tcp_local_option.

The option file consists of one or more keywords and an associated value. For example you can disable the SMTP EXPN command on your server by including the DISABLE_EXPAND keyword in the option file and setting the value to 1.

Other option file keywords allow you to:

- Set a limit on the number of recipients allowed per message (ALLOW_RECIPIENTS_PER_TRANSACTION)

- Set a limit on the number of messages allowed per connection (ALLOW_TRANSACTIONS_PER_SESSION)
- Control the type of information logged to the MTA log file (LOG_CONNECTION, LOG_TRANSPORTINFO)
- Specify the maximum number of simultaneous outbound connection that the client channel program allows (MAX_CLIENT_THREADS)

For information about all channel option keywords and syntax, see the *Messaging Server Reference Manual*.

Conversion File

The conversion file, `conversions`, specifies how the conversion channel performs conversions on messages flowing through the MTA. Any subset of MTA traffic can be selected for conversion and any set of programs or command procedures can be used to perform conversion processing. The MTA looks at the conversion file to choose an appropriate conversion for each body part.

For more information about the syntax of this file, see the *Messaging Server Reference Manual*.

Dirsync Option File

The dirsync option file, `dirsync.opt`, sets options for the `dirsync` program that cannot be set through the command line.

For more information about the syntax of this file, see the *Messaging Server Reference Manual*.

Dispatcher Configuration File

The Dispatcher configuration file, `dispatcher.cnf`, specifies Dispatcher configuration information. A default configuration file is created at installation time and can be used without any changes made. However, if you want to modify the default configuration file for security or performance reasons, you can do so by editing the `dispatcher.cnf` file.

The Dispatcher configuration file format is similar to the format of other MTA configuration files. Lines specifying options have the following form:

option=value

The *option* is the name of an option and *value* is the string or integer to which the options is set. If the *option* accepts an integer *value*, a base may be specified using notation of the form *b%v*, where *b* is the base expressed in base 10 and *v* is the actual value expressed in base *b*. Such option specifications are grouped into sections corresponding to the service to which the following option settings apply, using lines of the following form:

`[SERVICE=`*service-name*`]`

The *service-name* is the name of a service. Initial option specifications that appear before any such section tag apply globally to all sections.

The following is a sample Dispatcher configuration file (`dispatcher.cnf`).

```
! The first set of options, listed without a [SERVICE=xxx]
! header, are the default options that will be applied to all
! services.
!
MIN_PROCS=0
MAX_PROCS=5
MIN_CONNS=5
MAX_CONNS=20
MAX_LIFE_TIME=86400
MAX_LIFE_CONNS=100
MAX_SHUTDOWN=2
!
! Define the services available to Dispatcher
!
[SERVICE=SMTP]
PORT=25
IMAGE=server_root/msg-instance/imta/lib/tcp_smtp_server
LOGFILE=server_root/msg-instance/imta/log/tcp_smtp_server.log
```

For more information about the parameters for this file, see the *Messaging Server Reference Manual*.

Mapping File

The mapping file, `mappings`, defines how the MTA maps input strings to output strings.

Many components of the MTA employ table lookup-oriented information. Generally speaking, this sort of table is used to transform (that is, map) an input string into an output string. Such tables, called mapping tables, are usually presented as two columns, the first (or left-hand) column giving the possible input strings and the second (or right-hand) column giving the resulting output string for the input it is associated with. Most of the MTA databases are instances of this type of mapping table. The MTA database files, however, do not provide wildcard-lookup facilities, owing to inherent inefficiencies in having to scan the entire database for wildcard matches.

The mapping file provides the MTA with facilities for supporting multiple mapping tables. Full wildcard facilities are provided, and multi-step and iterative mapping methods can be accommodated as well. This approach is more compute-intensive than using a database, especially when the number of entries is large. However, the attendant gain in flexibility may actually serve to eliminate the need for most of the entries in an equivalent database, and this may actually result in lower overhead overall.

You can test mapping tables with the `imsimta test -mapping` command. For more information about the syntax of the mapping file and the `test -mapping` command, see the *Messaging Server Reference Manual*.

Option File

The options file, `option.dat`, specifies global MTA options—as opposed to channel options.

You can use the options file to override the default values of various parameters that apply to the MTA as a whole. In particular, the option file is used to establish sizes of the various tables into which the configuration and alias files are read. You can also use the options file to limit the size of messages accepted by the MTA, specify the number of channels allowed in the MTA configuration, set the number of rewrite rules allowed, and so on.

For more information about the syntax of the options file, see the *Messaging Server Reference Manual*.

Tailor File

The tailor file, `imta_tailor`, sets the location of various MTA components. For the MTA to function properly, the `imta_tailor` file must always reside in the *server-instance*`/imta/config` directory.

Although you can edit this file to reflect the changes in a particular installation, you must do so with caution. After making any changes to this file, you must restart the MTA. It is preferable to make the changes while the MTA is down.

NOTE Do not edit this file unless absolutely necessary.

For more information about the syntax of this file, see the *Messaging Server Reference Manual*.

Job Controller File

The Job Controller file, `job_controller.cnf`, specifies channel processing information. This file:

- Defines various pools
- Specifies for all channels the master program name and the slave program name, if applicable

In the `imta.cnf` file, you can specify the name of a process pool (that was defined in `job_controller.cnf`) by using the `pool` *keyword*. For example, the following fragment from a sample `job_controller.cnf` file defines the pool `MY_POOL`:

```
[POOL=MY_POOL]
job_limit = 12
```

The following fragment from a sample `imta.cnf` file specifies the pool `MY_POOL` in a channel block:

```
channel_x pool MY_POOL
channel_x-daemon
```

If you want to modify the parameters associated with the default pool configuration or add additional pools, you can do so by editing the `job_controller.cnf` file, then stopping and restarting the Job Controller.

A new Job Controller process is created, using the new configuration, and receives subsequent requests. The old Job Controller process continues to execute any requests it has queued until they are all finished, at which time it exits.

The first pool in the Job Controller configuration file is used for any requests that do not specify the name of a pool. The MTA channels defined in the MTA configuration file (`imta.cnf`) can have their processing requests directed to a specific pool by using the `pool` channel keyword followed by the name of the pool. The pool name must match the name of a pool in the Job Controller configuration. If the Job Controller does not recognize the requested pool name, the request is ignored.

In the initial configuration, the following pools are defined: `DEFAULT`, `LOCAL_POOL`, `IMS_POOL`, `SMTP_POOL`.

Examples of Use

Typically, you would add additional pool definitions to the Job Controller configuration if you wanted to differentiate processing of some channels from that of other channels. You might also choose to use pools with different characteristics. For example, you might need to control the number of simultaneous requests that some channels are allowed to process. You can do this by creating a new pool with the job limit, then use the `pool` channel keyword to direct those channels to the new, more appropriate pool.

In addition to the definition of pools, the Job Controller configuration file also contains a table of the MTA channels and the commands that the Job Controller must use to process requests for each channel. The two types of requests are termed “master” and “slave.” Typically, a channel master program is invoked when there is a message stored in an MTA message queue for the channel. The master program dequeues the message.

A slave program is invoked to poll a channel and pick up any messages inbound on that channel. While nearly all MTA channels have a master program, many do not have or need a slave program. For example, a channel that handles SMTP over TCP/IP doesn’t use a slave program because a network service, the SMTP server, receives incoming SMTP messages upon request by any SMTP server. The SMTP channel's master program is the MTA’s SMTP client.

If the destination system associated with the channel cannot handle more than one message at a time, you need to create a new type of pool whose job limit is one:

```
[POOL=single_job]
job_limit=1
```

On the other hand, if the destination system has enough parallelism, you can set the job limit to a higher value.

Figure 6-5 shows a sample Job Controller configuration file.

Figure 6-5 Sample Job Controller Configuration file on UNIX

```
!MTA Job Controller configuration file
!
!Global defaults
tcp_port=27442(1)
secret=never mind
return_job=server_root/bin/msg/imta/bin/return.sh
return_time=00:30/24:00
purge_job=server_root/bin/msg/imta/bin/purge
purge_argv=-num=5
slave_command=NULL(2)
max_life_age=3600(3)
!
!
!Pool definitions
!
[POOL=DEFAULT](4)
job_limit=10(5)
!
[POOL=LOCAL_POOL]
job_limit=10
!
[POOL=IMS_POOL]
job_limit=1
!
[POOL=SMTP_POOL]
job_limit=1
!
!Channel definitions
!
!
[CHANNEL=l](6)
master_command=server_root/bin/msg/imta/bin/l_master
!
[CHANNEL=ims-ms]
master_command=server_root/bin/msg/imta/bin/ims_master
!
[CHANNEL=tcp_*](7)
anon_host=0
master_command=server_root/bin/msg/imta/bin/tcp_smtp_client
```

The key items in the preceding example (numbered, enclosed in parentheses, and in bold font) are:

1. This global option defines the TCP port number on which the Job Controller listens for requests.
2. Sets a default `SLAVE_COMMAND` for subsequent [`CHANNEL`] sections.
3. Sets a default `MAX_LIFE_AGE` for subsequent [`CHANNEL`] sections.
4. This [`POOL`] section defines a pool named `DEFAULT`.
5. Set the `JOB_LIMIT` for this pool to 10.
6. This [`CHANNEL`] section applies to a channel named `l`, the UNIX local channel. The only definition required in this section is the `master_command`, which the Job Controller issues to run this channel. Since no wildcard appears in the channel name, the channel must match exactly.
7. This [`CHANNEL`] section applies to any channel whose name begins with `tcp_*`. Since this channel name includes a wildcard, it will match any channel whose name begins with `tcp_`.

For more information about the syntax of the Job Controller file, see the *Messaging Server Reference Manual*.

Aliases

The MTA provides a facility to support mailbox names associated with the local system that do not necessarily correspond to actual users: *aliases*. Aliases are useful for constructing mailing lists, forwarding mail, and providing synonyms for user names.

Aliases apply only to addresses that match the `l` channel or to any channel marked with the `aliaslocal` keyword. Each time an address that matches the `l` channel or any channel marked with the `aliaslocal` keyword is encountered by the MTA's message submission logic, the mailbox (for example, username) specified in the address is compared against each entry in the alias database or alias file. If a match occurs the alias address is replaced by the translation value or values specified by the alias. An alias can translate into any combination and number of additional aliases or real addresses. The real addresses need not themselves be associated with the `l` channel or any channel marked with the `aliaslocal` keyword and thus aliases can be used to forward mail to remote systems.

Since the only addresses truly considered to match a channel are `Envelope To` addresses, aliases can apply only to `Envelope To` addresses. The MTA performs alias translation and expansion only after address rewriting is completed. The translation values produced by an alias are treated as completely new addresses and are reprocessed from scratch.

The Alias Database

The MTA uses the information in the directory and creates the alias database. The alias database is consulted once each time the regular alias files is consulted. However, the alias database is checked before the regular alias file is used. In effect, the database acts as a sort of address rewriter that is invoked prior to using the alias file. Refer to the *iPlanet Messaging Server 5.0 Provisioning Guide* for information on what directory attributes are used to create user and distribution list entries in the alias database.

NOTE The format of the database itself is private. Do not try to edit the database directly. Make all required changes in the directory.

The Alias File

The `aliases` file is used to set aliases not set in the directory. In particular, the postmaster alias is a good example. Aliases set in this file will be ignored, if the same aliases exist in the directory. The MTA has to be restarted for any changes to take effect. Any line that begins with an exclamation point is considered to be a comment and is ignored. Blank lines are also ignored.

A physical line in this file is limited to 252 characters. You can split a logical line into multiple physical lines using the backslash (\) continuation character.

The format of the file is as follows:

```
user@domain: <address> (for users in hosted domains)

user@domain: <address> (for users in non-hosted domains. For example,
default-domain)
```

For example:

```
! A /var/mail/ user
inetmail@siroe.com: inetmail@native-daemon

! A message store user
ms_testuser@siroe.com: mstestuser@ims-ms-daemon
```

Including Other Files in the Alias File

Other files can be included in the primary `aliases` file. A line of the following form directs the MTA to read the `file-spec` file:

`<file-spec`

The file specification must be a complete file path specification and the file must have the same protections as the primary `aliases` file; for example, it must be world readable.

The contents of the included file are inserted into the `aliases` file at its point of reference. The same effect can be achieved by replacing the reference to the included file with the file's actual contents. The format of include files is identical to that of the primary `aliases` file itself. Indeed, include files may themselves include other files. Up to three levels of include file nesting are allowed.

Command Line Utilities

iPlanet Messaging Server provides several command-line utilities that enable you to perform various maintenance, testing, and management tasks for the MTA. For example, you use the `imsimta cnbuild` command to compile the MTA configuration, alias, mapping, security, system wide filter, and option files. You use the `imsimta dirsync` command to recreate or update the MTA directory cache. For complete information on the MTA command-line utilities, see the *Messaging Server Reference Manual*.

The MTA Directory Cache

For each message that it processes, the MTA needs to access information about the users, groups (mailing lists, family accounts, organization), and domains that it supports. This information is stored in an LDAP directory service. Rather than querying the directory service each time it processes a message, the MTA caches the directory information; that is, it takes a snapshot of the directory information and stores it. The MTA then accesses the directory information in the cache.

The directory information stored in the directory service is continuously updated. As a result, the directory information in the MTA-directory cache must be updated periodically—that is, synchronized—with the current directory information in the directory service. Messaging Server supports two types of synchronization:

- **Full synchronization** - The existing cache is replaced with a new cache, completely rebuilt with the current user and group entries from the directory service. After the synchronization occurs, the MTA configuration file is rebuilt, then the MTA is automatically restarted.
- **Incremental synchronization** - The existing cache is updated with user and group entries that were created or modified since the last full or incremental synchronization. The MTA is not restarted.

By default, the MTA directory cache is fully synchronized every day at 2:00 am and incrementally synchronized every ten minutes.

Table 6-3 shows which updates occur on full and partial directory synchronizations.

Table 6-3 MTA Directory Cache Updates

MTA Directory Cache Update	Full Synchronization	Incremental Synchronization
New user entries added	Yes	Yes
Modified user entries updated	Yes	Yes
*Deleted user entries removed	Yes	No
New members added to existing distribution lists	Yes	Yes
Deleted members removed from existing distribution lists	Yes	Yes
New distribution lists added	Yes	Yes
*Deleted distribution lists removed	Yes	No

Table 6-3 MTA Directory Cache Updates

MTA Directory Cache Update	Full Synchronization	Incremental Synchronization
*For incremental directory synchronization to take account of deleted entries, the entry's status must first be marked as deleted. After performing an incremental synchronization, the MTA considers the user or group to be nonexistent. The actual directory deletion must be done only after the incremental synchronization.		

In general, directory synchronization occurs automatically. However, if necessary, you can use the `imsimta dirsync` command to recreate or update the MTA directory cache. For more information on the `imsimta dirsync` command, see the *Messaging Server Reference Manual*.

Synchronization Configuration Parameters

Table 6-4 lists the directory synchronization configuration parameters.

Table 6-4 Directory Synchronization Configuration Parameters

Parameter	Description
`local.imta. ldsearchtimeout`	Specifies the LDAP search timeout when searching for users and mailing list information. The default is no timeout.
`local.imta. hostnamealiase`	When checking the `mailhost` or `mailRoutingHosts` attribute of an LDAP entry to see if it is local, the `dirsync` process uses the `local.hostname` parameter to do the comparison. In addition, a comma separated list of hostname aliases can be provided through the `local.imta.hostnamealiases` parameter. The `dirsync` process will then use all the hostnames provided in those 2 parameters to check if an entry is local.
`local.imta. mailaliases`	By default, the MTA considers only the `mail` and `mailAlternateAdress` LDAP attributes as routable email addresses. Alternatively, a comma separated list of LDAP attributes can be provided through the `local.imta.mailaliases` parameter. This list overwrites the default attributes. For example, the MTA will consider the following four attributes when routing messages: `local.imta.mailaliases=mail,mailAlternateAdres, rfc822mailbox,rfc822mailalias`

Table 6-4 Directory Synchronization Configuration Parameters

Parameter	Description
`local.imta.ugfilter`	This parameter sets the LDAP search filter that `dirsync` uses when searching for users and mailing list information. The default filter is (`objectClass=inetLocalMailRecipient`). For example, if you want to consider only LDAP entries with the `inetLocalMailRecipient` AND `myispSubscriber` object classes, you would set this parameter to: `local.imta.ugfilter= (&(objectClass=inetLocalMailRecipient) (objectClass=myispSubscriber))` Note: A timestamp filter will be added to this filter in the case of an incremental synchronization. As a consequence, you need to wrap your custom filter with ().
`local.imta.statssamplesize`	If set, this parameter tells `dirsync` to print out on the standard output a summary of the number of user and mailing list entries proceeded since the beginning as well as an average rate in entries / second. Users and mailing lists are counted whether or not they are successfully synchronized.
`local.imta.reverseenabled`	Triggers the generation of the reverse database. The default value is `yes`. How the reverse database is actually used is controlled by the USE_REVERSE_DATABASE option.
`local.imta.ssrenabled`	Triggers the generation of the server side rule (SSR) database. The default value is `yes`. How the SSR database is actually used is controlled by the `ssr` channel keyword.
`local.imta.vanityenabled`	Controls whether or not vanity domains (`msgVanityDomain` user LDAP attribute) are enabled. The default is `yes`.
`local.imta.catchallenabled`	Controls whether or not catchall addresses (`mail` or `mailAlternateAddress` of the form `@domain`) are enabled. The default is `yes`.
`local.imta.scope`	This parameter tells `dirsync` which entries it should synchronize: Cache only user and mailing list entries for which the `mailhost` attribute is the local host: value = "local". Cache user and mailing list entries regardless of their `mailhost` attribute: value = "domains". This is the default value if the parameter is missing. Do not cache any domain, user, or mailing list: value = "nobody"

SMTP Security and Access Control

For information about SMTP security and access control, see Chapter 9, “Mail Filtering and Access Control,” and Chapter 11, “Configuring Security and Access Control.”

Log Files

All MTA specific log files are kept in the MTA log directory, (*server-instance*`/log/imta/`). This directory contains log files that describe message traffic through the MTA and log files that describe information about specific master or slave programs.

For more information about MTA log files, see Chapter 12, “Logging and Log Analysis.”

Chapter 7

Configuring Rewrite Rules

This chapter describes how to configure rewrite rules in the `imta.cnf` file. If you have not already read Chapter 6, “About MTA Services and Configuration,” you should do so before reading this chapter.

This chapter contains the following sections:

- Rewrite Rule Structure
- Rewrite Rule Patterns and Tags
- Rewrite Rule Templates
- How the MTA Applies Rewrite Rules to an Address
- Template Substitutions and Rewrite Rule Control Sequences
- Handling Large Numbers of Rewrite Rules
- Testing Rewrite Rules
- Rewrite Rules Example

Messaging Server’s address rewriting facility is the primary facility for manipulating and changing the host or domain portion of addresses. Messaging Server provides other facilities for address manipulation, such as aliases, the address reversal database, and specialized mapping tables. For best performance, however, rewrite rules should be used whenever possible to perform address manipulations.

NOTE When you make changes to rewrite rules in the `imta.cnf` file, you must restart any programs or channels that load the configuration data only once when they start up—for example, the SMTP server—by using the `imsimta start` command. If you are using a compiled configuration, you must recompile and then restart.

For more information about compiling configuration information and starting programs, see the *Messaging Server Reference Manual*.

Rewrite Rule Structure

Rewrite rules appear in the upper-half of the MTA configuration file, `imta.cnf`. Each rule in the configuration file appears on a single line. Comments, but not blank lines, are allowed between the rules. The rewrite rules end with a blank line, after which the channel definitions follow. Figure 7-1 shows the rewrite rule section of a partial configuration file.

Figure 7-1 Simple Configuration File - Rewrite Rules

```
! test.cnf - An example configuration file.
!
! This is only an example of a configuration file. It serves
! no useful purpose and should not be used in a real system.
!
a     $U@a-host
b     $U@b-host
c     $U%c@b-daemon
d     $U%d@a-daemon

! Begin channel definitions
```

Rewrite rules consist of two parts: a pattern, followed by an equivalence string or template. The two parts must be separated by spaces, although spaces are not allowed within the parts themselves. The structure for rewrite rules is as follows:

```
pattern   template
```

pattern

Indicates the string to search for in the domain name. In Figure 6-1, the patterns are a,b,c, and d.

If the pattern matches the domain part of the address, the rewrite rule is applied to the address. A blank space must separate the pattern from the template. For more information about pattern syntax, see “Rewrite Rule Patterns and Tags” on page 124.

template

Is one of the following. For more information about template syntax, see “Rewrite Rule Templates” on page 127.

UserTemplate%*DomainTemplate*@*ChannelTag*[*controls*]

UserTemplate@*ChannelTag*[*controls*]

UserTemplate%*DomainTemplate*[*controls*]

UserTemplate@*DomainTemplate*@*ChannelTag*[*controls*]

UserTemplate@*DomainTemplate*@*SourceRoute*@*ChannelTag*[*controls*]

UserTemplate	Specifies how the user part of the address is rewritten. Substitution sequences can be used to represent parts of the original address or the results of a database lookup. The substitution sequences are replaced with what they represent to construct the rewritten address. In Figure 6-1, the $U substitution sequence is used. For more information, see “Template Substitutions and Rewrite Rule Control Sequences” on page 136.
DomainTemplate	Specifies how the domain part of the address is rewritten. Like the *UserTemplate*, the *DomainTemplate* can contain substitution sequences.

ChannelTag	Indicates the channel to which this message is sent. (All channel definitions must include a channel tag as well as a channel name. The channel tag typically appears in rewrite rules, as well as in its channel definition.)
controls	The applicability of a rule can be limited using controls. Some control sequences must appear at the beginning of the rule; other controls must appear at the end of the rule. For more information about controls, see “Template Substitutions and Rewrite Rule Control Sequences” on page 136.

Rewrite Rule Patterns and Tags

Most rewrite rule patterns consist either of a specific host name that will match only that host or of a subdomain pattern that will match any host / domain in the entire subdomain.

For example, the following rewrite rule pattern contains a specific host name that will match the specified host only:

```
host.siroe.com
```

The next rewrite rule pattern contains a subdomain pattern that will match any host or domain in the entire subdomain:

```
.siroe.com
```

This pattern will not, however, match the exact host name `siroe.com`; to match the exact host name `siroe.com`, a separate `siroe.com` pattern would be needed.

The MTA attempts to rewrite host / domain names starting from the specific host name and then incrementally generalizing the name to make it less specific. This means that a more specific rewrite rule pattern will be preferentially used over more general rewrite rule patterns. For example, assume the following rewrite rule patterns are present in the configuration file:

```
hosta.subnet.siroe.com
.subnet.siroe.com
.siroe.com
```

Based on the rewrite rule patterns, an address of `jdoe@hosta.subnet.siroe.com` will match the `hosta.subnet.siroe.com` rewrite rule pattern; an address of `jdoe@hostb.subnet.siroe.com` will match the `.subnet.siroe.com` rewrite rule pattern; and an address of `jdoe@hostc.siroe.com` will match the `.siroe.com` rewrite rule pattern.

In particular, the use of rewrite rules incorporating subdomain rewrite rule patterns is common for sites on the Internet. Such a site will typically have a number of rewrite rules for their own internal hosts and subnets, and then will include rewrite rules for the top-level Internet domains into their configuration from the file `internet.rules` (*server-instance*`/imta/config/internet.rules`).

To ensure that messages to Internet destinations (other than to the internal host destinations handled via more specific rewrite rules) will be properly rewritten and routed to an outgoing TCP/IP channel, ensure that the `imta.cnf` file contains:

- Rewrite rules with patterns that match the top level Internet domains
- Templates that rewrite addresses matching such patterns to an outgoing TCP/IP channel

```
!    Ascension Island
.AC                                          $U%$H$D@TCP-DAEMON
. [text
.    removed for
.             brevity]
!    Zimbabwe
.ZW                                          $U%$H$D@TCP-DAEMON
```

IP domain literals follow a similar hierarchical matching pattern, though with right-to-left (rather than left-to-right) matching. For example, the following pattern matches only and exactly the IP literal `[1.2.3.4]`:

```
[1.2.3.4]
```

The next pattern matches anything in the `1.2.3.0` subnet:

```
[1.2.3.]
```

In addition to the more common sorts of host or subdomain rewrite rule patterns already discussed, rewrite rules may also make use of several special patterns, summarized in Table 7-1, and discussed in the following subsections.

Table 7-1 Summary of Special Patterns for Rewrite Rules

Pattern	Description/Usage
$%	Percent Hack Rule. Matches any host/domain specification of the form A%B.

Table 7-1 Summary of Special Patterns for Rewrite Rules

Pattern	Description/Usage
$!	Bang-style Rule. Matches any host / domain specification of the form B!A.
[]	IP literal match-all rule. Matches any IP domain literal.
.	Matches any host / domain specification. For example, `joe@[129.165.12.11]`

In addition to these special patterns, Messaging Server also has the concept of *tags*, which may appear in rewrite rule patterns. These tags are used in situations where an address may be rewritten several times and, based upon previous rewrites, distinctions must be made in subsequent rewrites by controlling which rewrite rules match the address. For more information, see “Tagged Rewrite Rule Sets” on page 127.

A Rule to Match Percent Hacks

If the MTA tries to rewrite an address of the form A%B and fails, it tries one extra rule before falling through and treating this address form as A%B@localhost. (For more information about these address forms, see “Rewrite Rule Templates” on page 127.) This extra rule is the *percent hack rule*. The pattern is $%. The pattern never changes. This rule is only activated when a local part containing a percent sign has failed to rewrite any other way (including the match all rule described below).

The percent hack rule is useful for assigning some special, internal meaning to percent hack addresses.

A Rule to Match Bang-Style (UUCP) Addresses

If the MTA tries to rewrite an address of the form B!A and fails, it tries one extra rule before falling through and treating this address form as B!A@localhost. This extra rule is the *bang-style rule*. The pattern is $!. The pattern never changes. This rule is only activated when a local part containing an exclamation point has failed to rewrite any other way (including the default rule described below).

The bang-style rule can be used to force UUCP style addresses to be routed to a system with comprehensive knowledge of UUCP systems and routing.

A Rule to Match Any Address

The special pattern "." (a single period) will match any host/domain specification if no other rule matches and the host/domain specification cannot be found anywhere in the channel table. In other words, the "." rule is used as a last resort when address rewriting would fail otherwise.

NOTE Regarding substitution sequences, when the match-all rule matches and its template is expanded, $H expands to the full host name and $D expands to a single dot ".". Thus, $D is of limited use in a match-all rule template!

Tagged Rewrite Rule Sets

As the rewrite process proceeds it may be appropriate to bring different sets of rules into play. This is accomplished by the use of the rewrite rule tag. The current tag is prepended to each pattern before looking it up in the configuration file or domain database. The tag can be changed by any rewrite rule that matches by using the $T substitution string in the rewrite rule template (described below).

Tags are somewhat sticky; once set they will continue to apply to all hosts that are extracted from a single address. This means that care must be taken to provide alternate rules that begin with the proper tag values once any tags are used. In practice this is rarely a problem since tags are usually used in only very specialized applications. Once the rewriting of the address is finished the tag is reset to the default tag—an empty string.

By convention all tag values end in a vertical bar |. This character is not used in normal addresses and thus is free to delineate tags from the rest of the pattern.

Rewrite Rule Templates

The following sections describe in more detail template formats for rewrite rules. Table 7-2 summarizes the template formats.

Table 7-2 Summary of Template Formats for Rewrite Rules

Template	Usage
A%B	A becomes the new user/mailbox name, B becomes the new host/domain specification, rewrite again.

Table 7-2 Summary of Template Formats for Rewrite Rules

Template	Usage
A@B	Treated as A%B@B.
A%B@C	A becomes the new user / mailbox name, B becomes the new host / domain specification, route to the channel associated with the host C.
A@B@C	Treated as A@B@C@C.
A@B@C@D	A becomes the new user / mailbox name, B becomes the new host / domain specification, insert C as a source route, route to the channel associated with the host D.

Ordinary Rewriting Templates: A%B@C or A@B

The following template is the most common form of template. The rule is applied to the user part of the address and to the domain part of the address. The new address is then used to route the message to a specific channel (indicated by *ChannelTag*).

UserTemplate%*DomainTemplate*@*ChannelTag*[*controls*]

The next form of template is identical in application to the most common form of template. However, this form of template is possible only if *DomainTemplate* and *ChannelTag* are identical.

UserTemplate@*ChannelTag*[*controls*]

Repeated Rewrites Template, A%B

The following template format is used for meta-rules that require additional rewriting after application of the rule. After the rule is applied, the entire rewriting process is repeated on the resulting new address. (All other rewrite rule formats cause the rewriting process to terminate after the rule has been applied.)

UserTemplate%*DomainTemplate*[*controls*]

For example, the following rule has the effect of removing all occurrences of the .removable domain from the ends of addresses:

```
.removable      $U%$H
```

Extreme care must be taken when using these repeating rules; careless use can create a “rules loop.” For this reason meta-rules should only be used when absolutely necessary. Be sure to test meta-rules with the `imsimta test -rewrite` command. For more information on the `test -rewrite` command, see the *Messaging Server Reference Manual*.

Specified Route Rewriting Templates, A@B@C@D or A@B@C

The following template format works in the same way as the more common template *UserTemplate%DomainTemplate@ChannelTag* (note the difference in the first separator character), except that *ChannelTag* is inserted into the address as a source route. The message is then routed to *ChannelTag*:

UserTemplate@*DomainTemplate*@*Source-Route*
    @*ChannelTag*[*controls*]

The rewritten address becomes @*route*:*user*@*domain*. The following template is also valid:

UserTemplate@*DomainTemplate*@*ChannelTag*[*controls*]

For example, the following rule rewrites the address `jdoe@com1` into the source-routed address `@siroe.com:jdoe@com1`. The channel tag becomes `siroe.com`:

```
com1 $U@com1@siroe.com
```

Case Sensitivity in Rewrite Rule Templates

Unlike the patterns in rewrite rules, character case in templates is preserved. This is necessary when using rewrite rules to provide an interface to a mail system that is sensitive to character case. Note that substitution sequences like $U and $D that substitute material extracted from addresses also preserve the original case of characters.

When it is desirable to force substituted material to use a particular case, for example, to force mailboxes to lowercase on UNIX systems, special substitution sequences can be used in templates to force substituted material to the desired case. Specifically, $\ forces subsequent substituted material into lower case, $^ forces subsequent substituted material into upper case, and $_ says to use the original case.

For example, you can use the following rule to force mailboxes to lowercase for `unix.siroe.com` addresses:

```
unix.siroe.com    $\$U$_%unix.siroe.com
```

How the MTA Applies Rewrite Rules to an Address

The following steps describe how the MTA applies rewrite rules to a given address:

1. The MTA extracts the first host or domain specification from an address.

   An address can specify more than one host or domain name as in the case:

   `jdoe%hostname@siroe.com.`

2. After identifying the first host or domain name, the MTA conducts a search that scans for a rewrite rule whose pattern matches the host or domain name.

3. When the matching rewrite rule is found, the MTA rewrites the address according to the template portion of that rule.

4. Finally, the MTA compares the channel tag with the host names that are associated with each channel.

   If a match is found, the MTA enqueues the message to the associated channel; otherwise, the rewrite process fails. If the matching channel is the local channel, some additional rewriting of the address may take place by looking up the alias database and alias file.

These steps are described in more detail in the subsections that follow.

NOTE Using a channel tag that does not belong to any existing channel will cause messages whose addresses match this rule to be bounced. That is, it makes the matching messages nonroutable.

Step 1. Extract the First Host or Domain Specification

The process of rewriting an address starts by extracting the first host or domain specification from the address. (Readers not familiar with RFC 822 address conventions are advised to read that standard to understand the following discussion.) The order in which host / domain specifications in the address are scanned is as follows:

1. Hosts in source routes (read from left to right)
2. Hosts appearing to the right of the "at" sign (@)
3. Hosts appearing to the right of the last single percent sign (%)
4. Hosts appearing to the left of the first exclamation point (!)

The order of the last two items is switched if the `bangoverpercent` keyword is in effect on the channel that is doing the address rewriting. That is, if the channel attempting to enqueue the message is, itself, marked with the `bangoverpercent` channel keyword.

Some examples of addresses and the host names that could be extracted first are shown in Table 7-3.

Table 7-3 Extracted Addresses and Host Names

Address	First Host Domain Specification	Comments
`user@a`	`a`	A "short-form" domain name.
`user@a.b.c`	`a.b.c`	A "fully qualified" domain name (FQDN).
`user@[0.1.2.3]`	`[0.1.2.3]`	A "domain literal."
`@a:user@b.c.d`	`a`	Source-routed address with a short-form domain name, the "route."
`@a.b.c:user@d.e.f`	`a.b.c`	Source-routed address; route part is fully qualified.
`@[0.1.2.3]:user@d.e.f`	`[0.1.2.3]`	Source-routed address; route part is a domain literal.
`@a,@b,@c:user@d.e.f`	`a`	Source-routed address with an a to b to c routing.

Table 7-3 Extracted Addresses and Host Names *(Continued)*

Address	First Host Domain Specification	Comments
`@a,@[0.1.2.3]:user@b`	a	Source-routed address with a domain literal in the route part.
`user%A@B`	B	This nonstandard form of routing is called a "percent hack."
`user%A`	A	
`user%A%B`	B	
`user%%A%B`	B	
`A!user`	A	"Bang-style" addressing; commonly used for UUCP.
`A!user@B`	B	
`A!user%B@C`	C	
`A!user%B`	B	`nobangoverpercent` keyword active; the default.
`A!user%B`	A	`bangoverpercent` keyword active.

RFC 822 does not address the interpretation of exclamation points (!) and percent signs (%) in addresses. Percent signs are customarily interpreted in the same manner as at signs (@) if no at sign is present, so this convention is adopted by the Messaging Server MTA.

The special interpretation of repeated percent signs is used to allow percent signs as part of local user names; this might be useful in handling some foreign mail system addresses. The interpretation of exclamation points conforms to RFC 976's "bang-style" address conventions and makes it possible to use UUCP addresses with the Messaging Server MTA.

The order of these interpretations is not specified by either RFC 822 or RFC 976, so the `bangoverpercent` and `nobangoverpercent` keywords can be used to control the order in which they are applied by the channel doing the rewriting. The default is more "standard," although the alternate setting may be useful under some circumstances.

NOTE The use of exclamation points (!) or percent signs (%) in addresses is not recommended.

Step 2. Scan the Rewrite Rules

Once the first host or domain specification has been extracted from the address, the MTA consults the rewrite rules to find out what to do with it. The host / domain specification is compared with the pattern part of each rule (that is, the left side of each rule). The comparison is case insensitive. Case insensitivity is mandated by RFC 822. The MTA is insensitive to case but preserves it whenever possible.

If the host or domain specification does not match any pattern, in which case it is said to "not match any rule," the first part of the host or domain specification—the part before the first period, usually the host name—is removed and replaced with an asterisk (*) and another attempt is made to locate the resulting host or domain specification, but only in the configuration file rewrite rules (the domain database is not consulted).

If this fails, the first part is removed and the process is repeated. If this also fails the next part is removed (usually a subdomain) and the rewriter tries again, first with asterisks and then without. All probes that contain asterisks are done only in the configuration file rewrite rules table; the domain database is not checked. This process proceeds until either a match is found or the entire host or domain specification is exhausted. The effect of this procedure is to try to match the most specific domain first, working outward to less specific and more general domains.

A more algorithmic view of this matching procedure is:

- The host / domain specification is used as the initial value for the comparison strings `spec_1` and `spec_2`. (For example, `spec_1 = spec_2 = a.b.c`).
- The comparison string `spec_1` is compared with the pattern part of each rewrite rule in the configuration file and then the domain database until a match is found. The matching procedure is exited if a match is found.
- If no match is found, then the left-most, nonasterisk part of `spec_2` is converted to an asterisk. For example, if `spec_2` is `a.b.c` then it is changed to `*.b.c`; if `spec_2` is `*.b.c`, then it is changed to `*.*.c`. The matching procedure is exited if a match is found.
- If no match is found then the first part, including any leading period, of the comparison string `spec_1` is removed. Where `spec_1` has only one part (for example, `.c` or `c`), the string is replaced with a single period, ".". If the resulting string `spec_1` is of nonzero length, then you return to step 1. If the resulting string has zero length (for example, was previously "."), then the lookup process has failed and you exit the matching procedure.

For example, suppose the address `dan@sc.cs.siroe.edu` is to be rewritten. This causes the MTA to look for the following patterns in the given order:

```
sc.cs.siroe.edu
*.cs.siroe.edu
.cs.siroe.edu
*.*.siroe.edu
.siroe.edu
*.*.*.edu
.edu
*.*.*.*
.
```

Step 3. Rewrite Address According to Template

Once the host / domain specification matches a rewrite rule, it is rewritten using the template part of the rule. The template specifies three things:

1. A new user name for the address.
2. A new host / domain specification for the address.
3. A channel tag that identifies an existing MTA channel to which messages to this address should be sent.

Step 4. Finish the Rewrite Process

One of two things can happen once the host / domain specification is rewritten.

- If the channel tag is associated neither with the local channel nor a channel marked with the `routelocal` channel keyword, or there are no additional host / domain specifications in the address, the rewritten specification is substituted into the address replacing the original specification that was extracted for rewriting, and the rewriting process terminates.
- If the channel tag matches the local channel or a channel marked `routelocal` and there are additional host / domain specifications that appear in the address, the rewritten address is discarded, the original (initial) host / domain specification is removed from the address, a new host / domain specification is extracted from the address, and the entire process is repeated. Rewriting will continue until either all the host / domain specifications are gone or a route

through a non-local, non-routelocal channel is found. This iterative mechanism is how the MTA provides support for source routing. In effect, superfluous routes through the local system and routelocal systems are removed from addresses by this process.

Rewrite Rule Failure

If a host / domain specification fails to match any rewrite rule and no default rule is present, the MTA uses the specification "as-is"; for example, the original specification becomes both the new specification and the routing system. If the address has a nonsensical host / domain specification it will be detected when the routing system does not match any system name associated with any channel and the message will be bounced.

Syntax Checks After Rewrite

No additional syntax checking is done after the rewrite rules have been applied to an address. This is deliberate—it makes it possible for rewrite rules to be used to convert addresses into formats that do not conform to RFC 822. However, this also means that mistakes in the configuration file may result in messages leaving the MTA with incorrect or illegal addresses.

Handling Domain Literals

Domain literals are handled specially during the rewriting process. If a domain literal appearing in the domain portion of an address does not match a rewrite rule pattern as is, the literal is interpreted as a group of strings separated by periods and surrounded by square brackets. The right-most string is removed and the search is repeated. If this does not work, the next string is removed, and so on until only empty brackets are left. If the search for empty brackets fails, the entire domain literal is removed and rewriting proceeds with the next section of the domain address, if there is one. No asterisks are used in the internal processing of domain literals; when an entire domain literal is replaced by an asterisk, the number of asterisks corresponds to the number of elements in the domain literal.

Like normal domain or host specifications, domain literals are also tried in most specific to least specific order. The first rule whose pattern matches will be the one used to rewrite the host or domain specification. If there are two identical patterns in the rules list, the one which appears first will be used.

As an example, suppose the address dan@[128.6.3.40] is to be rewritten. The rewriter looks for [128.6.3.40], then [128.6.3.], then [128.6.], then [128.], then [], then [*.*.*.*], and finally the match-all rule ".".

When domain literals are combined with domain names the number of lookup attempts gets to be quite large. This is not normal usage and its use is strongly discouraged. For example, the address dan@[1.2].a.[3.4].b would generate requests for:

```
[1.2].a.[3.4].b
[1.].a.[3.4].b
[].a.[3.4].b
[*.*].a.[3.4].b
.a.[3.4].b
[*.*].*.[3.4].b
.[3.4].b
[*.*].*.[3.].b
.[3.].b
[*.*].*.[].b
.[].b
[*.*].*.[*.*].b
.b
[*.*].*.[*.*].*
.
```

Template Substitutions and Rewrite Rule Control Sequences

Substitutions are used to rewrite user names or addresses by inserting a character string into the rewritten address, the value of which is determined by the particular substitution sequence used.

For example, in the following template, the $U is a substitution sequence. It causes the *username* portion of the address being rewritten to be substituted into the output of the template. Thus, if jdoe@mailhost.siroe.com was being rewritten by this template, the resulting output would be jdoe@siroe.com, the $U substituting in the *username* portion, jdoe, of the original address:

```
$U@siroe.com
```

Control sequences impose additional conditions to the applicability of a given rewrite rule. Not only must the pattern portion of the rewrite rule match the host or domain specification being examined, but other aspects of the address being rewritten must meet conditions set by the control sequence or sequences. For example, the $E control sequence requires that the address being rewritten be an envelope address, while the $F control sequence requires that it be a forward pointing address. The following rewrite rule only applies to (rewrite) envelope To: addresses of the form `user@siroe.com`:

```
siroe.com $U@mail.siroe.com$E$F
```

If a domain or host specification matches the pattern portion of a rewrite rule but doesn't meet all of the criteria imposed by a control sequences in the rule's template, then the rewrite rule fails and the rewriter continues to look for other applicable rules.

Table 7-4 summarizes the template substitutions and control sequences..

Table 7-4 Summary of Template Substitutions and Control Sequences

Substitution Sequence	Substitutes
`$D`	Portion of domain specification that matched.
`$H`	Unmatched portion of host / domain specification; left of dot in pattern.
`$L`	Unmatched portion of domain literal; right of dot in pattern literal.
`$U`	User name from original address.
`$0U`	Local part (username) from original address, minus any subaddress.
`$1U`	Subaddress, if any, from local part (username) of original address.
`$$`	Inserts a literal dollar sign ($).
`$%`	Inserts a literal percent sign (%).
`$@`	Inserts a literal at sign (@).
`$\`	Forces material to lowercase.
`$^`	Forces material to uppercase.
`$_`	Uses original case.
`$W`	Substitutes in a random, unique string.
`$]...[`	LDAP search URL lookup.

Table 7-4 Summary of Template Substitutions and Control Sequences *(Continued)*

Substitution Sequence	Substitutes
`$(text)`	General database substitution; rule fails if lookup fails.
`${...}`	Applies specified mapping to supplied string.
`$[...]`	Invoke customer supplied routine; substitute in result.
$&*n*	The *nth* part of unmatched (or wildcarded) host, counting from left to right, starting from 0.
$!*n*	The *nth* part of unmatched (or wildcarded) host, as counted from right to left, starting from 0.
$***n*	The *nth* part of matching pattern, counting from left to right, starting from 0.
$#*n*	The *nth* part of matching pattern, counted from right to left, starting from 0.
$*n*D	Portion of domain specification that matched, preserving from the nth leftmost part starting from 0
$*n*H	Portion of host / domain specification that didn't match, preserving from the nth leftmost part starting from 0
Control Sequence	**Effect on Rewrite Rule**
$E	Apply only to envelope addresses
$B	Apply only to header / body addresses
$F	Apply only to forward-directed (e.g., To:) addresses
$R	Apply only to backwards-directed (e.g., From:) addresses
$M *channel*	Apply only if *channel* is rewriting the address
$N *channel*	Fail if *channel* is rewriting the address
$Q *channel*	Apply if sending to *channel*
$C *channel*	Fail if sending to *channel*
$S	Apply if host is from a source route
$A	Apply if host is to the right of the at sign
$P	Apply if host is to the right of a percent sign
$X	Apply if host is to the left of an exclamation point
$T*newtag*	Set the rewrite rule tag to newtag
$?*errmsg*	If rewriting fails return errmsg instead of the default error message

Username and Subaddress Substitution, $U, $0U, $1U

Any occurrences of $U in the template are replaced with the username (RFC 822 “local-part”) from the original address. Note that user names of the form a."b" will be replaced by "a.b" as current Internet standardization work is deprecating the former syntax from RFC 822 and it is expected that the latter usage will become mandatory in the future.

Any occurrences of $0U in the template are replaced with the username from the original address, minus any subaddress and subaddress indication character (+). Any occurrences of $1U in the template are replaced with the subaddress and subaddress indication character, if any, from the original address. So note that $0U and $1U are complementary pieces of the username, with $0U$1U being equivalent to a simple $U.

Host/Domain and IP Literal Substitutions, $D, $H, $nD, $nH, $L

Any occurrences of $H are replaced with the portion of the host / domain specification that was not matched by the rule. Any occurrences of $D are replaced by the portion of the host / domain specification that was matched by the rewrite rule. The $nH and $nD characters are variants that preserve the normal $H or $D portion from the nth leftmost part starting counting from 0. That is, $nH and $nD omit the leftmost n parts (starting counting from 1) of what would normally be a $H or $D, substitution, respectively. In particular, $0H is equivalent to $H and $0D is equivalent to $D.

For example, assume the address `jdoe@host.siroe.com` matches the following rewrite rule:

```
host.siroe.com     $U%$1D@TCP-DAEMON
```

The resulting address is `jdoe@siroe.com` with `TCP-DAEMON` used as the outgoing channel. Here where $D would have substituted in the entire domain that matched, `host.siroe.com`, the $1D instead substitutes in the portions of the match starting from part 1 (part 1 being `siroe`), so substitutes in `siroe.com`.

$L substitutes the portion of a domain literal that was not matched by the rewrite rule.

Literal Character Substitutions, $$, $%, $@

The $, %, and @ characters are normally metacharacters in rewrite rule templates. To insert a literal such character, quote it with a dollar character, $. That is, $$ expands to a single dollar sign, $; $% expands to a single percent, % (the percent is not interpreted as a template field separator in this case); and $@ expands to a single at sign, @ (also not interpreted as a field separator).

LDAP Query URL Substitutions, $]...[

A substitution of the form `$]ldap-url[` is interpreted as an LDAP query URL and the result of the LDAP query is substituted. Standard LDAP URLs are used with the host and port omitted. The host and port are instead specified in the `msg.conf` file (`local.ldaphost` and `local.ldapport` attributes).

That is, the LDAP URL should be specified as follows where the square bracket characters, [ ], indicate optional portions of the URL:

`ldap:///dn[?attributes[?scope?filter]]`

The `dn` is required and is a distinguished name specifying the search base. The optional attributes, scope, and filter portions of the URL further refine what information to return. For a rewrite rule, the desired attributes to specify returning might be a `mailRoutingSystem` attribute (or some similar attribute). The scope may be any of base (the default), one, or sub. And the desired filter might be to request the return of the object whose `mailDomain` value matches the domain being rewritten.

If the LDAP directory schema includes attributes `mailRoutingSystem` and `mailDomain`, then a possible rewrite rule to determine to which system to route a given sort of address might appear as the following where here the LDAP URL substitution sequence $D is used to substitute in the current domain name into the LDAP query constructed:

```
.siroe.com \
  $U%$H$D@$]ldap:///o=siroe.com?mailRoutingSystem?sub? \
  (mailDomain=$D)[
```

For ease in reading, the backslash character is used to continue the single logical rewrite rule line onto a second physical line. Table 7-5 lists the LDAP URL Substitution Sequences.

Table 7-5 LDAP URL Substitution Sequences

Substitution Sequence	Description
$$	Literal $ character
$~ *account*	Home directory of user account
$A	Address
$D	Domain name
$H	Host name (first portion of fully qualified domain name)
$L	Username minus any special leading characters such as ~ or _
$S	Subaddress
$U	Username

General Database Substitutions, $(...)

A substitution of the form $(text) is handled specially. The text part is used as a key to access the special general database. This database consists of the file specified with the IMTA_GENERAL_DATABASE option in the `/imta/config/imta_tailor` file, which is usually the file `/imta/db/generaldb.db`.

This database is generated with the `imta crdb` utility. If “text-string” is found in the database, the corresponding template from the database is substituted. If “text-string” does not match an entry in the database, the rewrite process fails; it is as if the rewrite rule never matched in the first place. If the substitution is successful, the template extracted from the database is re-scanned for additional substitutions. However, additional $(text) substitutions from the extracted template are prohibited in order to prevent endless recursive references.

As an example, suppose that the address `jdoe@siroe.siroenet` matches the following rewrite rule:

```
.SIROENET $($H)
```

Then, the text string `siroe` will be looked up in the general database and the result of the look up, if any, is used for the rewrite rule's template. Suppose that the result of looking up `siroe` is `$u%eng.siroe.com@siroenet`. Then the output of the template will be `jdoe@eng.siroe.com` (i.e., username = `jdoe`, host / domain specification = `eng.siroe.com`), and the routing system will be `siroenet`.

If a general database exists it should be world readable to insure that it operates properly.

Apply Specified Mapping, ${...}

A substitution of the form `${mapping,argument}` is used to find and apply a mapping from the MTA mapping file. The `mapping` field specifies the name of the mapping table to use while `argument` specifies the string to pass to the mapping. The mapping must exist and must set the $Y flag in its output if it is successful; if it doesn't exist or doesn't set $Y the rewrite will fail. If successful the result of the mapping is merged into the template at the current location and re-expanded.

This mechanism allows the MTA rewriting process to be extended in various complex ways. For example, the username part of an address can be selectively analyzed and modified, which normally isn't a feature the MTA rewriting process is capable of.

Customer-supplied Routine Substitutions, $[...]

A substitution of the form `$[`*image*`,`*routine*`,`*argument*`]` is used to find and call a customer-supplied routine. At run-time on UNIX, the MTA uses `dlopen` and `dlsym` to dynamically load and call the specified routine from the shared library image. The routine is then called as a function with the following argument list:

`status :=` *routine* `(`*argument*`,` *arglength*`,` *result*`,` *reslength*`)`

argument and *result* are 252 byte long character string buffers. On UNIX, *argument* and *result* are passed as a pointer to a character string, (for example, in C, as `char*`.) *arglength* and *reslength* are signed, long integers passed by reference. On input, *argument* contains the argument string from the rewrite rule template, and *arglength* the length of that string. On return, the resultant string should be placed in *result* and its length in *reslength*. This resultant string will then replace the "$[*image*,*routine*,*argument*]" in the rewrite rule template. The routine should return 0 if the rewrite rule should fail and -1 if the rewrite rule should succeed.

This mechanism allows the rewriting process to be extended in all sorts of complex ways. For example, a call to some type of name service could be performed and the result used to alter the address in some fashion. Directory service lookups for forward pointing addresses (e.g., To: addresses) to the host `siroe.com` might be performed as follows with the following rewrite rule. The $F, described in "Direction-and-Location-Specific Rewrite Rules ($B, $E, $F, $R)" on page 146 causes this rule to be used only for forward pointing addresses:

```
siroe.com $F$[LOOKUP_IMAGE,LOOKUP,$U]
```

A forward pointing address `jdoe@siroe.com` will, when it matches this rewrite rule, cause `LOOKUP_IMAGE` (which is a shared library on UNIX) to be loaded into memory, and then cause the routine `LOOKUP` called with `jdoe` as the argument parameter. The routine `LOOKUP` might then return a different address, say, `John.Doe%eng.siroe.com` in the result parameter and the value -1 to indicate that the rewrite rule succeeded. The percent sign in the result string (see "Repeated Rewrites Template, A%B" on page 128), causes the rewriting process to start over again using `John.Doe@eng.siroe.com` as the address to be rewritten.

On UNIX systems, the site-supplied shared library image should be world readable.

NOTE This facility is not designed for use by casual users; it is intended to be used to extend Messaging Server's capabilities system-wide.

Single Field Substitutions, $&, $!, $*, $#

Single field substitutions extract a single subdomain part from the host / domain specification being rewritten. The available single field substitutions are shown in Table 7-6.

Table 7-6 Single Field Substitutions

Control Sequence	Usage
$&n	Substitute the nth element, n=0,1,2,..,9, in the host specification (the part that did not match or matched a wildcard of some kind). Elements are separated by dots; the first element on the left is element zero. The rewrite fails if the requested element does not exist.

Table 7-6 Single Field Substitutions

Control Sequence	Usage
$!n	Substitute the nth element, n=0,1,2,..,9, in the host specification (the part that did not match or matched a wildcard of some kind). Elements are separated by dots; the first element on the right is element zero. The rewrite fails if the requested element does not exist.
$*n	Substitute the nth element, n=0,1,2,...,9, in the domain specification (the part that did match explicit text in the pattern). Elements are separated by dots; the first element on the left is element zero. The rewrite fails if the requested element does not exist.
$#n	Substitute the nth element, n=0,1,2,...,9, in the domain specification (the part that did match explicit text in the pattern). Elements are separated by dots; the first element on the right is element zero. The rewrite fails if the requested element does not exist.

Suppose the address `jdoe@eng.siroe.com` matches the following rewrite rule:

```
*.SIROE.COM      $U%$&0.siroe.com@mailhub.siroe.com
```

Then the result from the template will be `jdoe@eng.siroe.com` with `mailhub.siroe.com` used as the routing system.

Unique String Substitutions

Each use of the $W control sequence inserts a text string composed of upper case letters and numbers that is designed to be unique and not repeatable. $W is useful in situation where nonrepeating address information must be constructed.

Source-Channel-Specific Rewrite Rules ($M, $N)

It is possible to have rewrite rules that act only in conjunction with specific source channels. This is useful when a short-form name has two meanings:

1. When it appears in a message arriving on one channel.
2. When it appears in a message arriving on a different channel.

Source-channel-specific rewriting is associated with the channel program in use and the channel keywords `rules` and `norules`. If `norules` is specified on the channel associated with an MTA component that is doing the rewriting, no channel-specific rewrite checking is done. If `rules` is specified on the channel, then channel-specific rule checks are enforced. The keyword `rules` is the default.

Source-channel-specific rewriting is not associated with the channel that matches a given address. It depends only on the MTA component doing the rewriting and that component's channel table entry.

Channel-specific rewrite checking is triggered by the presence of a `$N` or `$M` control sequence in the template part of a rule. The characters following the `$N` or `$M`, up until either an at sign (`@`), percent sign (`%`), or subsequent `$N`, `$M`, `$Q`, `$C`, `$T`, or `$?` are interpreted as a channel name.

For example, `$M`*channel* causes the rule to fail if *channel* is not currently doing the rewriting. `$N`*channel* causes the rule to fail if *channel* is doing the rewriting. Multiple `$M` and `$N` clauses may be specified. If any one of multiple `$M` clauses matches, the rule succeeds. If any of multiple `$N` clauses matches, the rules will fail.

Destination-Channel-Specific Rewrite Rules ($C, $Q)

It is possible to have rewrite rules whose application is dependent upon the channel to which a message is being enqueued. This is useful when there are two names for some host, one known to one group of hosts and one known to another. By using different channels to send mail to each group, addresses can be rewritten to refer to the host under the name known to each group.

Destination channel-specific rewriting is associated with the channel to which the message is to be dequeued and processed by, and the channel keywords `rules` and `norules` on that channel. If `norules` is specified on the destination channel, no channel-specific rewrite checking is done. If `rules` is specified on the destination channel, channel-specific rule checks are enforced. The keyword `rules` is the default.

Destination channel-specific rewriting is not associated with the channel matched by a given address. It depends only on the message's envelope `To:` address. When a message is enqueued, its envelope `To:` address is first rewritten to determine to which channel the message is enqueued. During the rewriting of the `envelope`

`To:` address, any `$C` and `$Q` control sequences are ignored. After the `envelope To:` address is rewritten and the destination channel determined, then the `$C` and `$Q` control sequences are honored, as other addresses associated with the message are rewritten.

Destination-channel-specific rewrite checking is triggered by the presence of a `$C` or `$Q` control sequence in the template part of a rule. The characters following the `$C` or `$Q`, up until either an at sign (@), percent sign (%), or subsequent `$N`, `$M`, `$C`, `$Q`, `$T`, or `$?` are interpreted as a channel name.

For example, `$Q`*channel* causes the rule to fail if *channel* is not the destination. For another example, `$C`*channel* causes the rule to fail if *channel* is the destination. Multiple `$Q` and `$C` clauses may be specified. If any one of multiple `$Q` clauses matches, the rule succeeds. If any of multiple `$C` clauses matches, the rule fails.

Direction-and-Location-Specific Rewrite Rules ($B, $E, $F, $R)

Sometimes you need to specify rewrite rules that apply only to envelope addresses or, alternately, only to header addresses. The control sequence `$E` forces a rewrite to fail if the address being rewritten is not an envelope address. The control sequence `$B` forces a rewrite to fail if the address being rewritten is not from the message header or body. These sequences have no other effects on the rewrite and may appear anywhere in the rewrite rule template.

Addresses may also be categorized by direction. A forward pointing address is one that originates on a `To:`, `Cc:`, `Resent-to:`, or other header or envelope line that refers to a destination. A backward pointing address is something like a `From:`, `Sender:`, or `Resent-From:`, that refers to a source. The control sequence `$F` causes the rewrite to be applied if the address is forward pointing. The control sequence `$R` causes the rewrite to be applied if the address is reverse pointing.

Host-Location-Specific Rewrites ($A, $P, $S, $X)

Circumstances occasionally require rewriting that is sensitive to the location where a host name appears in an address. Host names can appear in several different contexts in an address:

- In a source route

- To the right of the at sign (@)
- To the right of a percent sign (%) in the local-part
- To the left of an exclamation point in the local-part

Under normal circumstances, a host name should be handled in the same way, regardless of where it appears. Some situations might require specialized handling.

Four control sequences are used to control matching on the basis of the host's location in the address.

- $S specifies that the rule can match a host extracted from a source route.
- $A specifies that the rule can match a host found to the right of the @ sign.
- $P specifies that the rule can match a host found to the right of a % sign.
- $X specifies that the rule can match a host found to the left of an exclamation point (!).

The rule fails if the host is from a location other than the one specified. These sequences can be combined in a single rewrite rule. For example, if $S and $A are specified, the rule matches hosts specified in either a source route or to the right of the at sign. Specifying none of these sequences is equivalent to specifying all of them; the rule can match regardless of location.

Changing the Current Tag Value, $T

The $T control sequence is used to change the current rewrite rule tag. The rewrite rule tag is prepended to all rewrite rule patterns before they are looked up in the configuration file and domain database. Text following the $T, up until either an at sign, percent sign, $N, $M, $Q, $C, $T, or $? is taken to be the new tag.

Tags are useful in handling special addressing forms where the entire nature of an address is changed when a certain component is encountered. For example, suppose that the special host name internet, when found in a source route, should be removed from the address and the resulting address forcibly matched against the TCP-DAEMON channel.

This could be implemented with rules like the following (localhost is assumed to be the official name of the local host):

```
internet                $S$U@localhost$Tmtcp-force|
mtcp-force|.            $U%$H@TCP-DAEMON
```

The first rule will match the special host name internet if it appears in the source route. It forcibly matches internet against the local channel, which insures that it will be removed from the address. A rewrite tag is then set. Rewriting proceeds, but no regular rule will match because of the tag. Finally, the default rule is tried with the tag, and the second rule of this set fires, forcibly matching the address against the TCP-DAEMON channel regardless of any other criteria.

Controlling Error Messages Associated with Rewriting ($?)

The MTA provides default error messages when rewriting and channel matching fail. The ability to change these messages can be useful under certain circumstances. For example, if someone tries to send mail to an Ethernet router box, it may be considered more informative to say something like “our routers cannot accept mail” rather than the usual “illegal host/domain specified.”

A special control sequence can be used to change the error message that is printed if the rule fails. The sequence $? is used to specify an error message. Text following the $?, up to either an at sign (@), percent sign (%), $N, $M, $Q, $C, $T, or $? is taken to be the text of the error message to print if the result of this rewrite fails to match any channel. The setting of an error message is “sticky” and lasts through the rewriting process.

A rule that contains a $? operates just like any other rule. The special case of a rule containing only a $? and nothing else receives special attention --- the rewriting process is terminated without changing the mailbox or host portions of the address and the host is looked up as-is in the channel table. This lookup is expected to fail and the error message will be returned as a result.

For example, assume the final rewrite rule in the MTA configuration file is as follows:

```
. $?Unrecognized address; contact postmaster@siroe.com
```

In this example, any unrecognized host or domain specifications that can fail will, in the process of failing, generate the error message: `Unrecognized address; contact postmaster@siroe.com`.

Handling Large Numbers of Rewrite Rules

The MTA always reads in all the rewrite rules from the `imta.cnf` file and stores them in memory in a hash table. Use of a compiled configuration bypasses the overhead associated with reading the configuration file each and every time the information is needed; a hash table is still used to store all of the rewrite rules in memory. This scheme is adequate for small to medium numbers of rewrite rules. However, some sites may require as many as 10,000 rewrite rules or more, which can consume prohibitive amounts of memory.

The MTA solves this problem by providing an optional facility for storing large numbers of rewrite rules in an ancillary indexed data file. Whenever the regular configuration file is read, the MTA checks for the existence of the domain database. If this database exists, it is opened and consulted whenever an attempted match fails on the rules found in the configuration file. The domain database is only checked if a given rule is not found in the configuration file, so rules can always be added to the configuration file to override those in the database. By default, the domain database is used to store rewrite rules associated with hosted domains. The IMTA_DOMAIN_DATABASE attribute is stored in the `imta_tailor` file. The default location for the database is *server-instance*`/imta/db/domaindb.db`.

NOTE DO NOT EDIT THIS FILE BY HAND. When a hosted domain is created in the Directory Server, the `dirsync` process overwrites any existing domain database, so any custom edits will be lost.

Testing Rewrite Rules

You can test rewrite rules with the `imsimta test -rewrite` command. The `-noimage` qualifier will allow you to test changes made to the configuration file prior to recompiling the new configuration.

You may find it helpful to rewrite a few addresses using this utility with the `-debug` qualifier. This will show you step-by-step how the address is rewritten. For example, issue the following command:

```
% imsimta test -rewrite -debug joe@siroe.com
```

For a detailed description of the `imsimta test -rewrite` utility, see the *Messaging Server Reference Manual*.

Rewrite Rules Example

The following example provides sample rewrite rules and how sample addresses would be rewritten by the rules.

Suppose the configuration file for the system SC.CS.SIROE.EDU contained the rewrite rules shown in Figure 7-2.

Figure 7-2 Rewrite Rules Example

```
sc                      $U@sc.cs.siroe.edu
sc1                     $U@sc1.cs.siroe.edu
sc2                     $U@sc2.cs.siroe.edu
*                       $U%$&0.cs.siroe.edu
*.cs                    $U%$&0.cs.siroe.edu
*.cs.siroe              $U%$&0.cs.siroe.edu
*.cs.siroe.edu          $U%$&0.cs.siroe.edu@ds.adm.siroe.edu
sc.cs.siroe.edu         $U@$D
sc1.cs.siroe.edu        $U@$D
sc2.cs.siroe.edu        $U@$D
sd.cs.siroe.edu         $U@sd.cs.siroe.edu
.siroe.edu              $U%$H.siroe.edu@cds.adm.siroe.edu
.edu                    $U@$H$D@gate.adm.siroe.edu
[]                      $U@[$L]@gate.adm.siroe.edu
```

Table 7-7 shows some sample addresses and how they would be rewritten and routed according to the rewrite rules.

Table 7-7 Sample Addresses and Rewrites

Initial address	Rewritten as	Routed to
user@sc	user@sc.cs.siroe.edu	sc.cs.siroe.edu
user@sc1	user@sc1.cs.siroe.edu	sc1.cs.siroe.edu
user@sc2	user@sc2.cs.siroe.edu	sc2.cs.siroe.edu
user@sc.cs	user@sc.cs.siroe.edu	sc.cs.siroe.edu
user@sc1.cs	user@sc1.cs.siroe.edu	sc1.cs.siroe.edu
user@sc2.cs	user@sc2.cs.siroe.edu	sc2.cs.siroe.edu
user@sc.cs.siroe	user@sc.cs.siroe.edu	sc.cs.siroe.edu
user@sc1.cs.siroe	user@sc1.cs.siroe.edu	sc1.cs.siroe.edu
user@sc2.cs.siroe	user@sc2.cs.siroe.edu	sc2.cs.siroe.edu
user@sc.cs.siroe.edu	user@sc.cs.siroe.edu	sc.cs.siroe.edu
user@sc1.cs.siroe.edu	user@sc1.cs.siroe.edu	sc1.cs.siroe.edu
user@sc2.cs.siroe.edu	user@sc2.cs.siroe.edu	sc2.cs.siroe.edu
user@sd.cs.siroe.edu	user@sd.cs.siroe.edu	sd.cs.siroe.edu
user@aa.cs.siroe.edu	user@aa.cs.siroe.edu	ds.adm.siroe.edu
user@a.eng.siroe.edu	user@a.eng.siroe.edu	cds.adm.siroe.edu
user@a.cs.sesta.edu	user@a.cs.sesta.edu	gate.adm.siroe.edu —route inserted
user@b.cs.sesta.edu	user@b.cs.sesta.edu	gate.adm.siroe.edu —route inserted
user@[1.2.3.4]	user@[1.2.3.4]	gate.adm.siroe.edu —route inserted

Basically, what these rewrite rules say is: If the host name is one of our short-form names (`sc`, `sc1` or `sc2`) or if it is one of our full names (`sc.cs.siroe.edu`, and so on), expand it to our full name and route it to us. Append `cs.cmu.edu` to one part short-form names and try again. Convert one part followed by `.cs` to one part followed by `.cs.siroe.edu` and try again. Also convert `.cs.siroe` to `.cs.siroe.edu` and try again.

If the name is `sd.cs.siroe.edu` (some system we connect to directly, perhaps) rewrite and route it there. If the host name is anything else in the `.cs.siroe.edu` subdomain, route it to `ds.cs.siroe.edu` (the gateway for the `.cs.siroe.edu` subdomain). If the host name is anything else in the `.siroe.edu` subdomain route it to `cds.adm.siroe.edu` (the gateway for the `.siroe.edu` subdomain). If the host name is anything else in the `.edu` top-level domain route it to `gate.adm.siroe.edu` (which is presumably capable of routing the message to its proper destination). If a domain literal is used send it to `gate.adm.siroe.edu` as well.

Most applications of rewrite rules (like the previous example) will not change the username (or mailbox) part of the address in any way. The ability to change the username part of the address is used when the MTA is used to interface to mailers that do not conform to RFC 822—mailers where it is necessary to stuff portions of the host / domain specification into the username part of the address. This capability should be used with great care if it is used at all.

Chapter 8

Configuring Channel Definitions

This chapter describes how to configure channel definitions in the MTA configuration file called `imta.cnf`.

If you have not already read Chapter 6, “About MTA Services and Configuration,” you should do so before reading this chapter. For information about configuring the rewrite rules in the `imta.cnf` file, see Chapter 7, “Configuring Rewrite Rules.”

This chapter contains the following sections:

- Channel Structure
- Predefined Channels
- Configuring SMTP Channels
- Configuring Message Processing and Delivery
- Configuring Messages Sent to the Postmaster
- Configuring Channel Options
- Configuring Channel Defaults
- Configuring Logging for Channels
- Configuring Debugging for Channels
- Setting Up Program Delivery
- Using the Hold Channel
- Using the Conversion Channel
- Understanding Conversions

NOTE When you make changes to channel definitions in the `imta.cnf` file, you must restart any programs or channels that load the configuration data only once when they start up—for example, the SMTP server—by using the `imsimta start` command. If you are using a compiled configuration, you must recompile and then restart.

For more information about compiling configuration information and starting programs, see the *Messaging Server Reference Manual*.

Channel Structure

Channel definitions appear in the lower half of the MTA configuration file following the rewrite rules. The first blank line to appear in the file signifies the end of the rewrite rules section and the start of the channel definitions.

A channel definition contains the name of the channel followed by an optional list of keywords that define the configuration of the channel, and a unique channel tag, which is used in rewrite rules to route messages to the channel. Channel definitions are separated by single blank lines. Comments, but no blank lines, may appear inside a channel definition.

```
[blank line]
! sample channel definition
ChannelName  keyword1  keyword2
Channel-Tag
[blank line]
```

Collectively, the channel definitions are referred to as the channel host table. An individual channel definition is also called a channel block. For example, in Figure 8-1 the channel host table contains three channel definitions or blocks.

Figure 8-1 Simple Configuration File - Channel Definitions

```
! test.cnf - An example configuration file.
!
! Rewrite Rules
      .
      .
      .

! BEGIN CHANNEL DEFINITIONS
! FIRST CHANNEL BLOCK
l
local-host

! SECOND CHANNEL BLOCK
a_channel defragment charset7 usascii
a-daemon

! THIRD CHANNEL BLOCK
b_channel noreverse notices 1 2 3
b-daemon
```

The channel host table defines the channels Messaging Server can use and the names of the systems associated with each channel.

On UNIX systems, the first channel block in the file always describes the local channel, `l`. (An exception is a `defaults` channel, which can appear before the local channel.) The local channel is used to make routing decisions and for sending mail sent by UNIX mail tools.

Predefined Channels

When you first install iPlanet Messaging Server, several channels are already defined. These channels are described in Table 8-1.

Table 8-1 Predefined Channels

Channel	Definition
`l`	**UNIX only.** Used to make routing decisions and for sending mail using UNIX mail tools.
`ims-ms`	Delivers mail to the local store.
`native`	**UNIX only.** Delivers mail to `/var/mail`. (Note that Messaging Server does not support `/var/mail` access. User must use UNIX tools to access mail from the `/var/mail` store.)
`pipe`	Used to perform delivery via a site-supplied program or script. Commands executed by the pipe channel are controlled by the administrator via the imsimta program interface. For more information, see “Setting Up Program Delivery” on page 187.

`tcp_local`
`tcp_intranet`
`tcp_auth`
`tcp_submit`
`tcp_tas`
	Implements SMTP over TCP/IP. The multithreaded TCP SMTP channel includes a multithreaded SMTP server that runs under the control of the Dispatcher. Outgoing SMTP mail is processed by the channel program tcp_smtp_client, and runs as needed under the control of the Job Controller.
	
	`tcp_local` receives inbound messages from remote SMTP hosts. Depending on whether you use a smarhost/firewall configuration, either sends outbound messages directly to remote SMTP hosts or sends outbound messages to the smarthost/firewall system.
	
	`tcp_intranet` receives and sends messages within the intranet.
	
	`tcp_auth` is used as a switch channel for tcp_local; authenticated users switch to the tcp_auth channel to avoid realy-blocking restrictions.
	
	`tcp_submit` accepts message submissions—usually from user agents—on the reserved submission port 587 (see RFC 2476).
	
	`tcp_tas` is a special channel used by sites doing Unified Messaging.

Table 8-1 Predefined Channels

Channel	Definition
`reprocess` `process`	These channels are used for deferred, offline message processing. The `reprocess` channel is normally invisible as a source or destination channel; the `process` channel is visible like other MTA channels.
`defragment`	Provides the means to reassemble MIME fragmented messages.
`conversion`	Performs body-part-by-body-part conversions on messages flowing through the MTA.
`bitbucket`	Used for messages that need to be discarded.
`inactive/deleted`	Used to process messages for users who have been marked as inactive / deleted in the directory. Typically, bounces the message and returns custom bounce message to the sender of the message.
`hold`	Used to hold messages for users. For example, when a user is migrated from one mail server to another.
`autoreply`	Used to process autoreply and vacation notice requests.

Configuring SMTP Channels

Depending on the type of installation, Messaging Server provides several SMTP channels at installation time: `tcp_local`, `tcp_intranet`, `tcp_submit`, `tcp_auth`, `tcp_tas`. You can modify the definitions of these channels or create new channels.

This section is divided into the following subsections:

- SMTP Command and Protocol Support
- TCP / IP Connection and DNS Lookup Support
- SMTP Authentication and SASL
- Transport Layer Security
- Channel Operation Type

SMTP Command and Protocol Support

You can specify whether an SMTP channel supports certain SMTP commands, such as EHLO, ETRN, and VRFY. You can also specify whether the channel support DNS domain verification, which characters the channel accepts as line terminators, and so on. This section describes the following:

- Channel Protocol Selection and Line Terminators
- EHLO Command Support
- ETRN Command Support
- VRFY Command Support
- DNS Domain Verification
- Character Set Labeling and Eight-Bit Data
- Protocol Streaming

Table 8-2 summarizes the keywords described in this section.

Table 8-2 SMTP Command and Protocol Keywords

Channel Keyword(s)	Description
Protocol Selection and Line Terminators	**Specifies whether the channel supports the SMTP protocol and specifies the character sequences accepted as line terminators.**
`smtp`	Supports the SMTP protocol. The keyword `smtp` is mandatory for all SMTP channels. (This keyword is equivalent to `smtp_crorlf`.)
`nosmtp`	Does not support the SMTP protocol. This is the default.
`smtp_cr`	Accepts lines terminated with a carriage return (CR) without a following line feed (LF).
`smtp_crlf`	Lines must be terminated with a carriage return (CR) line feed (LF) sequence.
`smtp_lf`	Accepts lines terminated with a linefeed (LF) without a preceding CR.
`smtp_crorlf`	Lines may be terminated with any of a carriage return (CR), or a line feed (LF) sequence, or a full CRLF.
EHLO keywords	**Specifies how the channel handles EHLO commands**
`ehlo`	Uses the SMTP EHLO command on initial connections.
`checkehlo`	Checks the SMTP response banner to determine whether to use EHLO or HELO.

Table 8-2 SMTP Command and Protocol Keywords

Channel Keyword(s)	Description
noehlo	Does not use the EHLO command.
ETRN keywords	**Specifies how the channel handles ETRN commands (requests for queue processing)**
allowetrn	Honors ETRN commands.
blocketrn	Blocks ETRN commands.
domainetrn	Honors only those ETRN commands that specify a domain.
silentetrn	Honors ETRN commands without echoing channel information.
sendetrn	Sends ETRN commands.
nosendetrn	Does not send ETRN commands.
VRFY keywords	**Specifies how the channel handles VRFY commands**
domainvrfy	Issues VRFY commands using a full address.
localvrfy	Issues VRFY commands using a local address.
novrfy	Does not issue VRFY commands.
vrfyallow	Provides informative responses to VRFY commands.
vrfydefault	Provides default responses to VRFY command, according to channel's HIDE_VERIFY option setting.
vrfyhide	Provides obfuscatory responses to SMTP VRFY command.
DNS Domain Verification	**Specifies whether the channel performs DNS domain verification**
mailfromdnsverify	Verifies that the domain used on the MAIL FROM: command exists in the DNS.
nomailfromdnsverify	Does not verify that the domain used on the MAIL FROM: command exists in the DNS.
Character Sets and Eight-bit data	**Specifies how the channel handles eight-bit data** Note: Although these keywords are commonly used on SMTP channels, they are potentially relevant to any sort of channel.
charset7	Default character set to associate with 7-bit text messages
charset8	Default character set to associate with 8-bit text messages
charsetesc	Default character set to associate with 7-bit text containing the escape character
eightbit	Channel supports eight-bit characters.

Table 8-2 SMTP Command and Protocol Keywords

Channel Keyword(s)	Description
`eightnegotiate`	Channel should negotiate use of eight-bit transmission if possible.
`eightstrict`	Channel should reject messages that contain unnegotiated eight-bit data.
`sevenbit`	Channel does not support eight-bit characters; eight-bit characters must be encoded.
Protocol streaming `streaming`	**Specify degree of protocol streaming for channel to use**

Channel Protocol Selection and Line Terminators

The `smtp` and `nosmtp` keywords specify whether or not a channel supports the SMTP protocol. The `smtp` keyword, or one of its variations, is mandatory for all SMTP channels.

The keywords `smtp_crlf`, `smtp_cr`, `smtp_crorlf`, and `smtp_lf` can be used on SMTP channels to specify the character sequences that the MTA will accept as line terminators. The keyword `smtp_crlf` means that lines must be terminated with a carriage return (CR) line feed (LF) sequence. The keyword `smtp_lf` or `smtp` means that an LF without a preceding CR is accepted. Finally, `smtp_cr` means that a CR is accepted without a following LF. These option affect only the handling of incoming material.

Because the SMTP standard requires CRLF as the line terminator, the MTA always generates the standard CRLF sequence. The various `smtp` keywords merely control whether the MTA will accept additional non-standard line terminators. For example, you can specify stmp_crlf if you want the MTA to accept only strictly legal SMTP messages and reject any messages with nonstandard line terminators.

EHLO Command Support

The SMTP protocol has been extended (RFC 1869) to allow for negotiation of additional commands. This is done by using the new `EHLO` command, which replaces RFC 821's `HELO` command. Extended SMTP servers respond to `EHLO` by providing a list of the extensions they support. Unextended servers return an unknown command error and the client then sends the old `HELO` command instead.

This fallback strategy normally works well with both extended and unextended servers. Problems can arise, however, with servers that do not implement SMTP according to RFC 821. In particular, some noncompliant servers are known to drop the connection on receipt of an unknown command.

The SMTP client implements a strategy whereby it attempts to reconnect and use HELO when any server drops the connection on receipt of an EHLO. However, this strategy might not work if the remote server not only drops the connection but also goes into a problematic state upon receipt of EHLO.

The channel keywords ehlo, noehlo, and checkehlo are provided to deal with such situations. The ehlo keyword tells the MTA to use the EHLO command on all initial connection attempts. The noehlo keyword disables all use of the EHLO command. The checkehlo keyword tests the response banner returned by the remote SMTP server for the string “ESMTP”. If this string is found EHLO is used; if not, HELO is used. The default behavior is to use EHLO on all initial connection attempts, unless the banner line contains the string “fire away”, in which case HELO is used; note that there is no keyword corresponding to this default behavior, which lies between the behaviors resulting from the ehlo and checkehlo keywords.

ETRN Command Support

The ETRN command, defined in RFC 1985, provides an extension to the SMTP service whereby an SMTP client and server can interact to give the server an opportunity to start the processing of its queues for messages to go to a given host.

Using ETRN, an SMTP client can request that a remote SMTP server start processing the message queues destined for sending to the SMTP client. Thus, ETRN provides a way to implement “polling” of remote SMTP systems for messages incoming to one’s own system. This can be useful for systems that have only transient connections between each other, for example, sites that are set up as secondary mail exchange (MX) hosts for other sites that only have a dial-up connection to the Internet. By enabling this command, you permit remote, possibly dial-up, servers to request delivery of their mail.

The SMTP client specifies on the SMTP ETRN command line the name of the system to which to send messages (generally the SMTP client system’s own name). If the remote SMTP server supports the ETRN command, it will trigger execution of a separate process to connect back to the named system and send any messages awaiting delivery for that named system.

Responding to ETRN Commands

The `allowetrn`, `blocketrn`, `domainetrn`, and `silentetrn` keywords control the MTA response when a sending SMTP client issues the `ETRN` command, requesting that the MTA attempt to deliver messages in the MTA queues.

By default, the MTA will attempt to honor all `ETRN` commands; that is, the `allowetrn` keyword is enabled. You can specify that the MTA not honor ETRN commands by including the `blocketrn` keyword in the channel definition.

You can specify that the MTA honor all `ETRN` commands, but without echoing the name of the channel that the domain matched and that the MTA will be attempting to run by including the `silentetrn` keyword. The `domainetrn` keyword specifies that the MTA honor only `ETRN` commands that specify a domain; it also causes the MTA not to echo back the name of the channel that the domain matched and that the MTA will be attempting to run.

Sending ETRN Commands

The `sendetrn` and `nosendetrn` channel keywords control whether the MTA sends an `ETRN` command at the beginning of an SMTP connection. The default is `nosendetrn`, meaning that the MTA will not send an `ETRN` command. The `sendetrn` keyword tells the MTA to send an `ETRN` command, if the remote SMTP server says it supports `ETRN`. The `sendetrn` keyword should be followed by the name of the system requesting that its messages receive a delivery attempt.

VRFY Command Support

The `VRFY` command enables SMTP clients to send a request to an SMTP server to verify that mail for a specific user name resides on the server. The `VRFY` command is defined in RFC 821.

The server sends a response indicating whether the user is local or not, whether mail will be forwarded, and so on. A response of 250 indicates that the user name is local; a response of 251 indicates that the user name is not local, but the server can forward the message. The server response includes the mailbox name.

Sending a VRFY Command

Under normal circumstances there is no reason to issue a `VRFY` command as part of an SMTP dialogue. The SMTP `RCPT TO` command should perform the same function that `VRFY` does and return an appropriate error. However, servers exist that can accept any address in a `RCPT TO` (and bounce it later), whereas these same servers perform more extensive checking as part of a `VRFY` command.

By default, the MTA does not send a `VRFY` command (the `novrfy` keyword is enabled).

If necessary, the MTA can be configured to issue the SMTP VRFY command by including the domainvrfy or localvrfy keyword in the channel definition. The keyword domainvrfy causes a VRFY command to be issued with a full address (user@host) as its argument. The localvrfy keyword causes the MTA to issue a VRFY command with just the local part of the address (user).

Responding to a VRFY Command

The vrfyallow, vrfydefault, and vrfyhide keywords control the SMTP server's response when a sending SMTP client issues an SMTP VRFY command.

The vrfyallow keyword tells the MTA to issue a detailed, informative response. The vrfydefault tells the MTA to provide a detailed, informative response, unless the channel option HIDE_VERIFY=1 has been specified. The vrfyhide keyword tells the MTA to issue only a vague, ambiguous response. These keywords allow per-channel control of VRFY responses, as opposed to the HIDE_VERIFY option, which normally applies to all incoming TCP/IP channels handled through the same SMTP server.

DNS Domain Verification

Setting mailfromdnsverify on an incoming TCP/IP channel causes the MTA to verify that an entry in the DNS exists for the domain used on the SMTP MAIL FROM command, and to reject the message if no such entry exists. The default, nomailfromdnsverify, means that no such check is performed. Note that performing DNS checks on the return address domain may result in rejecting some desired valid messages (for instance, from legitimate sites that simply have not yet registered their domain name, or at times of bad information in the DNS); it is contrary to the spirit of being generous in what you accept and getting the e-mail through, expressed in RFC 1123, Requirements for Internet Hosts. However, some sites may desire to perform such checks in cases where unsolicited bulk email (UBE) is being sent with forged e-mail addresses from non-existent domains.

Character Set Labeling and Eight-Bit Data

Character Set Labeling

The charset7, charset8, and charsetesc channel keywords provide a per-channel mechanism to specify character set names to be inserted into message headers which lack character set labelling. Each keyword requires a single argument giving the character set name. The names are not checked for validity. Note, however, that character set conversion can only be done on character sets specified in the character set definition file charsets.txt found in the MTA table directory. The names defined in this file should be used if possible.

The charset7 character set name is used if the message contains only seven bit characters; the charset8 character set name will be used if eight bit data is found in the message; charsetesc will be used if a message containing only seven bit data happens to contain escape characters also. If the appropriate keyword is not specified no character set name will be inserted into Content-type: header lines.

These character set specifications never override existing labels; that is, they have no effect if a message already has a character set label or is of a type other than text.

The charsetesc keyword tends to be particularly useful on channels that receive unlabeled messages using Japanese or Korean character sets that contain the escape character.

Eight-Bit Data

Some transports restrict the use of characters with ordinal values greater than 127 (decimal). Most notably, some SMTP servers will strip the high bit and thus garble messages that use characters in this eight-bit range.

Messaging Server provides facilities to automatically encode such messages so that troublesome eight bit characters do not appear directly in the message. This encoding can be applied to all messages enqueued to a given channel by specifying the sevenbit keyword. A channel should be marked eightbit if no such restriction exists.

The SMTP protocol disallows eightbit "unless the remote SMTP server explicitly says it supports the SMTP extension allowing eightbit." Some transports such as extended SMTP may actually support a form of negotiation to determine if eight bit characters can be transmitted. Therefore, the use of the eightnegotiate keyword is strongly recommended to instruct the channel to encode messages when negotiation fails. This is the default for all channels; channels that do not support negotiation will simply assume that the transport is capable of handling eight bit data.

The eightstrict keyword tells Messaging Server to reject any incoming messages that contain unnegotiated eight bit data.

Protocol Streaming

Some mail protocols support streaming operations. This means that the MTA can issue more than one operation at a time and wait for replies to each operation to arrive in batches. The streaming keyword controls the degree of protocol streaming used in the protocol associated with a channel. This keyword requires an integer parameter; how the parameter is interpreted is specific to the protocol in use.

Currently the MTA only supports the experimental use of streaming on SMTP channels. Implementation of this feature is experimental; it may change in future releases.

The streaming values available range from 0 to 3. A value of 0 specifies no streaming, a value of 1 causes groups of RCPT TO commands to stream, a value of 2 causes MAIL FROM / RCPT TO to stream, and a value of 3 causes HELO / MAIL FROM / RCPT TO or RSET / MAIL FROM / RCPT TO streaming to be used. The default value is 0.

Some SMTP implementations are known to react badly to streaming. In particular, sendmail is known to be incapable of handling streaming levels greater than 1. The iPlanet Messaging Server implementation of SMTP should work properly at any streaming level.

TCP/IP Connection and DNS Lookup Support

You can specify information about how the server handles TCP / IP connections and address lookups. This section describes the following:

- TCP / IP Port Number and Interface Address
- Caching for Channel Connection Information
- DNS Lookups
- IDENT Lookups
- TCP / IP MX Record Support
- Nameserver Lookups
- Last Resort Host
- Alternate Channels for Incoming Mail
- Target Host Choice

Table 8-3 lists the TCP/IP connection and DNS lookup keywords described in this section.

Table 8-3 TCP/IP Connection and DNS Lookup Keywords

Channel Keyword(s)	Description
Port Selection and Interface Address	**Specifies the default port number and interface address for SMTP connections**
`port`	Specifies the default port number for SMTP connections. The standard port is 25.
`interfaceaddress`	Binds to the specified TCP/IP interface address.
Cache Keywords	**Specifies how connection information is cached**
`cacheeverything`	Caches all connection information.
`cachefailures`	Caches only connection failure information.
`cachesuccesses`	Caches only connection success information.
`nocache`	Does not cache any connection information.
DNS Lookups	**Specifies how to handle DNS lookups on incoming SMTP connections**
`forwardcheckdelete`	If a reverse DNS lookup has been performed, next performs a forward lookup on the returned name to check that the returned IP number matches the original; if not, deletes the name and use the IP address.
`forwardchecknone`	Does not perform a forward lookup after a DNS reverse lookup.
`forwardchecktag`	If a reverse DNS lookup has been performed, next performs a forward lookup on the returned name to check that the returned IP number matches the original; if not, tags the name with *.
IDENT Lookups/DNS Reverse Lookups	**Specifies how to handle IDENT lookups and DNS Reverse Lookups on incoming SMTP connections**
`identnone`	Does not perform IDENT lookups; does perform IP to hostname translation; includes both hostname and IP address in `Received:` header.
`identnonelimited`	Does not perform IDENT lookups; does perform IP to hostname translation, but does not use the hostname during channel switching; includes both hostname and IP address in `Received:` header.
`identnonenumeric`	Does not perform IDENT lookups or IP to hostname translation.

Table 8-3 TCP / IP Connection and DNS Lookup Keywords

Channel Keyword(s)	Description
identnonesymbolic	Does not perform IDENT lookups; does perform IP to hostname translation; includes only the hostname in Received: header.
identtcp	Performs IDENT lookups on incoming SMTP connections and IP to hostname translation; include both hostname and IP address in Received: header
identtcplimited	Performs IDENT lookups on incoming SMTP connections and IP to hostname translation, but do not use the hostname during channel switching. Includes both hostname and IP address in Received: header.
indenttcpnumeric	Performs IDENT lookups on incoming SMTP connections, but does not perform IP to hostname translation.
identtcpsymbolic	Performs IDENT lookups on incoming SMTP connections and IP to hostname translation; includes only hostname in Received: header.
MX Record Support and TCP/IP Nameserver	**Specifies whether and how the channel supports MX record lookups**
mx	TCP / IP network and software supports MX records lookup.
nomx	TCP / IP network does not support MX lookups.
defaultmx	Channel determines whether to do MX lookups from network.
randommx	Does MX lookups; randomizes returned entries with equal precedence.
nonrandomemx	Does MX lookups; does not randomize returned entries with equal precedence.
nameservers	Specifies a list of nameservers to consult rather than consulting the TCP / IP stack's own choice of nameservers; nameservers requires a space separated list of IP addresses for the nameservers.
defaultnameservers	Consults TCP / IP stack's choice of nameservers.
lastresort	Specifies a last resort host.
Switch keywords	**Controls selection of alternate channels for incoming mail**
allowswitchchannel	Allows switching to this channel from a switchchannel channel

Table 8-3 TCP/IP Connection and DNS Lookup Keywords

Channel Keyword(s)	Description
`noswitchchannel`	Stays with the server channel; does not switch to the channel associated with the originating host; does not permit being switched to.
`switchchannel`	Switches from the server channel to the channel associated with the originating host.
`tlsswitchchannel`	Switches to another channel upon successful TLS negotiation.
`saslswitchchannel`	Switches to another channel when SASL authentication is successful.
Target Host Choice and Storage of Message Copies	**Specifies a target host system and how message copies are stored.**
`daemon`	Connects to a specific host system regardless of the envelope address.
`single`	Specifies that a separate copy of the message should be created for each destination address on the channel.
`single_sys`	Creates a single copy of the message for each destination system used.

TCP/IP Port Number and Interface Address

The SMTP over TCP/IP channels normally connect to port 25 when sending messages. The `port` keyword can be used to instruct an SMTP over TCP/IP channel to connect to a nonstandard port. Note that this keyword complements the Dispatcher option `PORT`, which controls which ports the MTA listens on for accepting SMTP connections.

The `interfaceaddress` keyword controls the address to which a TCP/IP channel binds as the source address for outbound connections; that is, on a system with multiple interface addresses this keyword controls which address will be used as the source IP address when the MTA sends outgoing SMTP messages. Note that this keyword complements the Dispatcher option `INTERFACE_ADDRESS`, which controls which interface address a TCP/IP channel listens on for accepting incoming connections and messages.

Caching for Channel Connection Information

Channels using the SMTP protocol maintain a cache containing a history of prior connection attempts. This cache is used to avoid reconnecting multiple times to inaccessible hosts, which can waste lots of time and delay other messages. The cache is a per process cache and only persists during a single run of the outbound SMTP delivery channel.

The cache normally records both connection successes and failures. (Successful connection attempts are recorded in order to offset subsequent failures—a host that succeeded before but fails now doesn't warrant as long of a delay before making another connection attempt as does one that has never been tried or one that has failed previously.)

However, the caching strategy used by the MTA is not necessarily appropriate for all situations. Therefore channel keywords are provided to adjust the MTA cache.

The `cacheeverything` keyword enables all forms of caching and is the default. The `nocache` keyword disables all caching.

The `cachefailures` keyword enables caching of connection failures but not successes—this forces a somewhat more restricted retry than `cacheeverything` does. Finally, `cachesuccesses` caches only successes. This last keyword is effectively equivalent to `nocache` for SMTP channels.

DNS Lookups

The `forwardchecknone`, `forwardchecktag`, and `forwardcheckdelete` channel keywords can modify the effects of doing reverse DNS lookups. These keywords can control whether the MTA does a forward lookup of an IP name found using a DNS reverse lookup, and if such forward lookups are requested, specify what the MTA does if the forward lookup of the IP name does not match the original IP number of the connection.

The `forwardchecknone` keyword is the default, and means that no forward lookup is done. The `forwardchecktag` keyword tells the MTA to do a forward lookup after each reverse lookup and to tag the IP name with an asterisk (*), if the number found using the forward lookup does not match that of the original connection. The `forwardcheckdelete` keyword tells the MTA to do a forward lookup after each reverse lookup and to ignore (delete) the reverse lookup returned name if the forward lookup of that name does not match the original connection IP address; in this case, the MTA uses the original IP address instead.

NOTE Having the forward lookup not match the original IP address is normal at many sites, where a more “generic” IP name is used for several different IP addresses.

IDENT Lookups

The IDENT keywords control how the MTA handles connections and lookups using the IDENT protocol. The IDENT protocol is described in RFC 1413.

The `identtcp`, `identtcpsymbolic`, and `identtcpnumeric` keywords tell the MTA to perform a connection and lookup using the IDENT protocol. The information obtained from the IDENT protocol (usually the identity of the user making the SMTP connection) is inserted into the `Received:` header of the message as follows:

- `identtcp` inserts the host name corresponding to the incoming IP number, as reported from a DNS reverse lookup and the IP number itself.
- `identtcpsymbolic` inserts the host name corresponding to the incoming IP number, as reported from a DNS reverse lookup; the IP number itself is not included in the `Received: header`.
- `identtcpnumeric` inserts the actual incoming IP number—no DNS reverse lookup on the IP number is performed.

NOTE The remote system must be running an `IDENT` server for the `IDENT` lookup caused by `identtcp`, `identtcpsymbolic`, or `identtcpnumeric` to be useful.

Be aware that `IDENT` query attempts may incur a performance hit. Increasingly routers will “black hole” attempted connections to ports that they don’t recognize. If this happens on an `IDENT` query, then the MTA does not hear back until the connection times out (a TCP/IP stack controlled time-out, typically on the order of a minute or two).

Another performance factor occurs when comparing `identtcp`, `indenttcplimited`, or `identtcpsymbolic` to `identtcpnumeric`. The DNS reverse lookup called for with `identtcp`, `identtcplimited`, or `identtcpsymbolic` incurs some additional overhead to obtain the more user-friendly host name.

The identnone keyword disables IDENT lookup, but does specify IP to host name translation, and both IP number and host name will be included in the Received: header for the message. This is the default.

The identnonesymbolic keyword disables IDENT lookup, but does do IP to host name translation; only the host name will be included in the Received: header for the message.

The identnonenumeric keyword disables this IDENT lookup and inhibits the usual DNS reverse lookup translation of IP number to host name, and might result in a performance improvement at the cost of less user-friendly information in the Received: header.

The identtcplimited and identnonelimited keywords have the same effect as identtcp and identnone, respectively, as far as IDENT lookups, reverse DNS lookups, and information displayed in Received: header. Where they differ is that with identtcplimited or identnonelimited the IP literal address is always used as the basis for any channel switching due to use of the switchchannel keyword, regardless of whether the DNS reverse lookup succeeds in determining a host name.

TCP/IP MX Record Support

Some TCP / IP networks support the use of MX (mail forwarding) records and some do not. Some TCP / IP channel programs can be configured not to use MX records if they are not provided by the network that the MTA system is connected to. The mx, nomx, defaultmx, randommx, nonrandommx keywords control MX record support.

The keyword randommx specifies that MX lookups should be done and MX record values of equal precedence should be processed in random order. The keyword nonrandommx specifies that MX lookups should be done and MX values of equal precedence should be processed in the same order in which they were received.

The mx keyword is currently equivalent to nonrandommx; it might change to be equivalent to randommx in a future release. The nomx keyword disables MX lookups. The defaultmx keyword specifies that mx should be used if the network says that MX records are supported. The keyword defaultmx is the default on channels that support MX lookups in any form.

Nameserver Lookups

When nameserver lookups are being performed, the `nameservers` channel keyword may be used to specify a list of nameservers to consult rather than consulting the TCP/IP stack's own choice of nameservers. The `nameservers` keyword requires a space separated list of IP addresses for the nameservers, as shown in the following example:

```
nameservers 1.2.3.1 1.2.3.2
```

The default, `defaultnameservers`, means use the TCP/IP stack's own choice of nameservers.

To prevent nameserver lookups on UNIX, you can modify the `nsswitch.conf` file. On NT, modify the TCP/IP configuration.

Last Resort Host

The `lastresort` keyword is used to specify a host to connect to even when all other connection attempts fail. In effect this acts as an MX record of last resort. This is only useful on SMTP channels.

Alternate Channels for Incoming Mail

The following keywords control selection of an alternate channel for incoming mail: `switchchannel`, `allowswitchchannel`, `noswitchchannel`.

When the MTA accepts an incoming connection from a remote system, it must choose a channel with which to associate the connection. Normally this decision is based on the transfer used; for example, an incoming SMTP over TCP/IP connection is automatically associated with the `tcp_local` channel.

This convention breaks down, however, when multiple outgoing channels with different characteristics are used to handle different systems over the same transfer. When this happens, incoming connections are not associated with the same channel as outgoing connections, and the result is that the corresponding channel characteristics are not associated with the remote system.

The `switchchannel` keyword provides a way to eliminate this difficulty. If switchchannel is specified on the initial channel the server uses, the IP address of the connecting (originating) host will be matched against the channel table and if it matches the source channel will change accordingly. If no IP address match is found or if a match is found that matches the original default incoming channel, the MTA may optionally try matching using the host name found by doing a DNS

reverse lookup. The source channel may change to any channel marked `switchchannel` or `allowswitchchannel` (the default). The `noswitchchannel` keyword specifies that no channel switching should be done to or from the channel.

Specification of `switchchannel` on anything other than a channel that a server associates with by default has no effect. At present, `switchchannel` only affects SMTP channels, but there are actually no other channels where `switchchannel` would be reasonable.

Target Host Choice

The `daemon` keyword is used on SMTP channels to control the choice of a target host.

Normally, channels connect to whatever host is listed in the envelope address of the message being processed. The `daemon` keyword is used to tell the channel to instead connect to a specific remote system, generally a firewall or mailhub system, regardless of the envelope address. The actual remote system name should appear directly after the daemon keyword, as shown in the following example:

```
tcp_firewall smtp mx daemon firewall.acme.com
TCP-DAEMON
```

If the argument after the daemon keyword is not a fully qualified domain name, the argument will be ignored and the channel will connect to the channel's official host. When specifying the firewall or gateway system name as the official host name, the argument given to the `daemon` keyword is typically specified as router, as shown in the following example:

```
tcp_firewall smtp mx daemon router
firewall.acme.com
TCP-DAEMON
```

Other keywords of interest are `single` and `single_sys`. The `single` keyword specifies that a separate copy of the message should be created for each destination address on the channel. The `single_sys` keyword creates a single copy of the message for each destination system used. Note that at least one copy of each message is created for each channel the message is queued to, regardless of the keywords used.

SMTP Authentication and SASL

You can control whether the Messaging Server supports authentication to the SMTP server using SASL (Simple Authentication and Security Layer). SASL is defined in RFC 2222. For more information about SASL, SMTP authentication, and security, see Chapter 11, “Configuring Security and Access Control.”

Table 8-4 lists the SASL keywords described in this section.

Table 8-4 SMTP Authentication with SASL

Keyword	Definition
`maysaslserver`	SMTP server supports SASL authentication. Clients can attempt to connect to the server using SASL authentication.
`mustsaslserver`	SMTP server supports SASL authentication. Clients must use SASL authentication; the SMTP server will not accept messages unless the remote client successfully authenticates.
`nosasl`	SASL authentication is not to be permitted or attempted. It subsumes `nosaslserver`, which means that SASL authentication will not be permitted. This is the default.
`nosaslserver`	SMTP server does not permit SASL authentication.
`saslswitchchannel`	Causes incoming connections to be switched to a specified channel upon a client’s successful use of SASL. It takes a required value, specifying the channel to which to switch.

Transport Layer Security

The `maytls`, `maytlsclient`, `maytlsserver`, `musttls`, `musttlsclient`, `musttlsserver`, `notls`, `notlsclient`, `notlsserver`, and `tlsswitchchannel` channel keywords are used to configure TLS use during the SMTP protocol by SMTP based channels such as TCP / IP channels.

The default is `notls`, and means that TLS will not be permitted or attempted. It subsumes the `notlsclient` keyword, which means that TLS use will not be attempted by the MTA SMTP client on outgoing connections (the `STARTTLS` command will not be issued during outgoing connections) and the `notlsserver` keyword, which means that TLS use will not be permitted by the MTA SMTP server on incoming connections (the `STARTTLS` extension will not be advertised by the SMTP server nor the command itself accepted).

Specifying `maytls` causes the MTA to offer TLS to incoming connections and to attempt TLS upon outgoing connections. It subsumes `maytlsclient`, which means that the MTA SMTP client will attempt TLS use when sending outgoing messages, if sending to an SMTP server that supports TLS, and `maytlsserver`, which means that the MTA SMTP server will advertise support for the `STARTTLS` extension and will allow TLS use when receiving messages.

Specifying `musttls` will cause the MTA to insist upon TLS in both outgoing and incoming connections; email will not be exchanged with remote systems that fail to successfully negotiate TLS use. It subsumes `musttlsclient`, which means that the MTA SMTP client will insist on TLS use when sending outgoing messages and will not send to SMTP servers that do not successfully negotiate TLS use (the MTA will issue the `STARTTLS` command and that command must succeed). It also subsumes `musttlsserver`, which means that the MTA SMTP server will advertise support for the `STARTTLS` extension and will insist upon TLS use when receiving incoming messages and will not accept messages from clients that do not successfully negotiate TLS use.

The `tlsswitchchannel` keyword is used to cause incoming connections to be switched to a specified channel upon a client's successful TLS negotiation. It takes a required value, specifying the channel to which to switch.

Channel Operation Type

Messaging Server supports RFC 2476's Message Submission protocol. The `submit` keyword may be used to mark a channel as a submit-only channel. This is normally useful mostly on TCP/IP channels, such as an SMTP server run on a special port used solely for submitting messages; RFC 2476 establishes port 587 for such message submission use.

Configuring Message Processing and Delivery

You can configure when the server attempts to deliver messages based on certain criteria. You can also specify parameters for job processing, such as processing limits for service jobs, or when to spawn a new SMTP channel thread. This section describes the following:

- Delivery of Messages
- Processing Pools for Channel Execution Jobs
- Service Job Limits

- SMTP Channel Threads
- Expansion of Multiple Addresses
- Undeliverable Message Notification Times

Table 8-5 summarizes the keywords described in this section.

Table 8-5 Message Processing and Delivery Keywords

Keyword	Definition
Immediate Delivery	**Defines specification for immediate delivery of messages.**
`immnonurgent`	Starts delivery immediately after submission for urgent, normal, and non-urgent messages.
Deferred Delivery	**Defines specification for delivery of deferred jobs.**
`backoff`	Specifies the frequency for attempted delivery of deferred messages. This keyword specifies the value for normal, urgent, and nonurgent mail unless one of the other backoff keywords is used. By default, a message that cannot be delivered will be retried as follows: after 1 hour, after 2 hours, after another 2 hours, and then every 4 hours for three more tries, at which point the server tries every 8 hours.
`nonurgentbackoff`	Specifies the frequency for attempted delivery of nonurgent messages.
`normalbackoff`	Specifies the frequency for attempted delivery of normal messages.
`urgentbackoff`	Specifies the frequency for attempted delivery of urgent messages.
Message Priority Based on Size	**Defines message priority based on message size.**
`nonurgentblocklimit`	Forces messages above this size to lower than nonurgent priority (second class priority), meaning that the messages will always wait for the next periodic job for further processing.
`normalblocklimit`	Forces messages above this size to nonurgent priority.
`urgentblocklimit`	Forces messages above this size to normal priority.
Processing Pools for Channel Execution Jobs	**Specifies the pools for processing messages of different urgencies and deferral of jobs**
`pool`	Specifies the pool in which channels run.
`after`	Specifies a time delay before channels run.

Table 8-5 Message Processing and Delivery Keywords

Keyword	Definition
Service Job Limits	**Specifies the number of service jobs and the maximum number of message files to handle per job**
`maxjobs`	Specifies the maximum number of jobs that can be running concurrently for the channel.
`filesperjob`	Specifies the number of queue entries to be processed by a single job.
SMTP Channel Threads `threaddepth`	Number of messages triggering new thread with multithreaded SMTP client.
Multiple Address Expansion	**Defines processing specification for messages with multiple recipient addresses**
`expandlimit`	Processes an incoming message “off-line” when the number of addressees exceeds this limit.
`expandchannel`	Specifies channel in which to perform deferred expansion due to application of expandlimit.
`holdlimit`	Holds an incoming message when the number of addresses exceeds this limit.
Undeliverable Message Notifications	**Specifies when undeliverable message notifications are sent.**
`notices`	Specifies the amount of time that may elapse before notices are sent and messages returned.
`nonurgentnotices`	Specifies the amount of time that may elapse before notices are sent and messages returned for messages of non-urgent priority.
`normalnotices`	Specifies the amount of time that may elapse before notices are sent and messages returned for messages of normal priority.
`urgentnotices`	Specify the amount of time which may elapse before notices are sent and messages returned for messages of urgent priority.

Delivery of Messages

Each time a message is enqueued to a channel, the Job Controller ensures that there is a job running to deliver the message. This might involve starting a new job process, adding a thread, or simply noting that a job is already running. If a job cannot be started because the job limit for the channel or pool has been reached, the Job Controller waits until another job has exited, then, when the job limit is no longer exceeded, starts another job. (Job limits for the channel are determined by the `maxjobs` channel keyword. Job limits for the pool are determined by the `JOB_LIMIT` option for the pool.)

Messaging Server attempts immediate delivery of all messages. If a message cannot be delivered on the first attempt, however, the message is delayed for a period of time determined by the appropriate `backoff` keyword. As soon as the time specified in the `backoff` keyword has elapsed, the delayed message is available for delivery, and if necessary, a channel job is started to process the message.

The Job Controller creates and manages channel jobs for delivering messages. These channel jobs run inside processing pools within the Job Controller. For more information about these pools, see “Processing Pools for Channel Execution Jobs” on page 179.

The Job Controller’s in-memory data structure of messages currently being processed and awaiting processing typically reflects the full set of message files stored on disk in the MTA queue area. However, if a backlog of message files on disk builds up large enough to exceed the Job Controller’s in-memory data structure size limit, then the Job Controller tracks in memory only a subset of the total number of messages files on disk. The Job Controller processes only those messages it is tracking in memory. After a sufficient number of messages have been delivered to free enough in-memory storage, the Job Controller automatically refreshes its in-memory store by scanning the MTA queue area to update its list of messages. The Job Controller then begins processing the additional message files it just retrieved from disk. The Job Controller performs these scans of the MTA queue area automatically.

If your site routinely experiences heavy message backlogs, you might want to tune the Job Controller by using the `MAX_MESSAGES` option. By increasing the `MAX_MESSAGES` option value to allow Job Controller to use more memory, you can reduce the number of occasions when message backlogs overflow the Job Controller’s in-memory cache. This reduces the overhead involved when the Job Controller must scan the MTA queue directory. Keep in mind, however, that when the Job Controller does need to rebuild the in-memory cache, the process will take

longer because the cache is larger. Note also that because the Job Controller must scan the MTA queue directory every time it is started or restarted, large message backlogs means that starts or restarts of the Job Controller will incur more overhead than starts or restarts when no such backlog exists.

Processing Pools for Channel Execution Jobs

You can configure various channels to share resources by running within the same pool. You might want to configure other channels to run in pools dedicated to a particular channel. Within each pool, messages are automatically sorted into different processing queues according to message priority. Higher priority messages in the pool are process before lower-priority messages. (See “Message Priority Based on Size” on page 181.)

The pools where the jobs are created can be selected on a channel by channel basis by using the `pool` keyword. The `pool` keyword must be followed by the name of the pool to which delivery jobs for the current channel should be pooled. The name of the pool should not contain more than twelve characters.

Execution of service jobs can be deferred using the `after` keyword. The `after` keyword must be followed by a specification of the amount of time to delay. If the value following the keyword is an unsigned integer value, it is interpreted as a number of seconds by which to defer delivery of the message—a delta time value.

Service Job Limits

Each time a message is enqueued to a channel, the Job Controller ensures that there is a job running to deliver the message. This might involve starting a new job process, adding a thread, or simply noting that a job is already running. A single service job may not be sufficient to ensure prompt delivery of all messages, however.

For any given installation, there is a reasonable maximum number of processes and threads to be started for delivering messages. This maximum number depends on factors such as the number of processors, the speed of the disks, and the characteristics of the connections. In the MTA configuration, it is possible to control the following:

- The maximum number of processes to start running for a given channel (the `maxjobs` channel keyword)

- The maximum number of processes to start for a set of channels (the `JOB_LIMIT` parameter in the relevant pool section of the Job Controller configuration file)
- The number of queued messages received before a new thread or process is started (the `threaddepth` channel keyword)
- For some channels, the maximum number of threads that will run within a given delivery program (`max_client_threads` parameter in the channel option file)

The maximum number of processes to start running for a given channel is the minimum of the `maxjobs` set on the channel and the `JOB_LIMIT` set for the pool that the channel runs in.

Assume a message needs processing. In general, the Job Controller starts new processes as follows:

- If no process is running for a channel and the pool job limit has not been reached, then the Job Controller starts a new process.
- If a channel program is single-threaded or the thread limit has been reached and the backlog increases past a multiple of threads (specified with `threaddepth`) and neither the channel nor pool job limit has been reached, the Job Controller starts a new process
- If a channel program is multithreaded and the thread limit has not been reached and the backlog of messages increase past a multiple of `threaddepth`, a new thread is started.

For SMTP channels in particular, new threads or processes are started as messages are enqueued for different hosts. Thus, for SMTP channels, the Job Controller starts new processes as follows. Assume a message needs processing:

- If no process is running for an SMTP channel and the pool limit has not been reached, the Job Controller starts a new process.
- If the thread limit (`MAX_CLIENT_THREADS`) has been reached, a message is enqueued for a host not yet being serviced, and neither the channel (`maxjobs`) nor pool job limit (`JOB_LIMIT`) has been reached then a new process is started.
- If the thread limit has not been reached and a message is enqueued for a host not yet being serviced, then a new thread is started.
- If the thread limit has not been reached and a message is enqueued that takes the backlog of messages for that host increase past a multiple of `threaddepth`, a new thread is started.

See also “SMTP Channel Threads” on page 181.

The `filesperjob` keyword can be used to cause the MTA to create additional service jobs. This keyword takes a single positive integer parameter which specifies how many queue entries (that is, files) must be sent to the associated channel before more than one service job is created to handle them. If a value less than or equal to zero is given it is interpreted as a request to queue only one service job. Not specifying a keyword is equivalent to specifying a value of zero. The effect of this keyword is maximized; the larger number computed will be the number of service jobs that are actually created.

The `filesperjob` keyword divides the number of actual queue entries or files by the given value. Note that the number of queue entries resulting from a given message is controlled by a large number of factors, including but not limited to the use of the `single` and `single_sys` keywords and the specification of header-modifying actions in mailing lists.

The `maxjobs` keyword places an upper bound on the total number of service jobs that can be running concurrently. This keyword must be followed by an integer value; if the computed number of service jobs is greater than this value only `maxjobs` jobs will actually be created. The default for this value if maxjobs is not specified is 100. Normally `maxjobs` is set to a value that is less than or equal to the total number of jobs that can run simultaneously in whatever service pool or pools the channel uses.

Message Priority Based on Size

The `urgentblocklimit`, `normalblocklimit`, and `nonurgentblocklimit` keywords may be used to instruct the MTA to downgrade the priority of messages based on size. These keywords affect the priority that the Job Controller applies when processing the message.

SMTP Channel Threads

The multithreaded SMTP client sorts outgoing messages to different destinations to different threads. The `threaddepth` keyword may be used to instruct the multithreaded SMTP client to handle only the specified number of messages in any one thread, using additional threads even for messages all to the same destination (hence normally all handled in one thread).

Use of `threaddepth` may be of particular interest for achieving multithreading on a daemon router TCP/IP channel—a TCP/IP channel that connects to a single specific SMTP server—when the SMTP server to which the channel connects can handle multiple simultaneous connections.

Each time the backlog for a channel increases past a multiple of `threaddepth`, the Job Controller tries to increase the amount of processing dedicated to processing messages queued for that channel. For multithreaded channels, the Job Controller advises any job processing messages for that channel to start a new thread, or if all jobs have the maximum threads allowed for the channel (`MAX_CLIENT_THREADS` in the option for the `tcp_*` channels) it will start a new process. For single-threaded channels it will start new process. Note that the Job Controller will not start a new job if the job limit for the channel (`maxjobs`) or the pool (`JOB_LIMIT`) has been reached.

Expansion of Multiple Addresses

Most channels support the specification of multiple recipient addresses in the transfer of each inbound message. The specification of many recipient addresses in a single message may result in delays in message transfer processing (online delays). If the delays are long enough, network timeouts can occur, which in turn can lead to repeated message submission attempts and other problems.

The MTA provides a special facility to force deferred (offline) processing if more than a given number of addresses are specified for a single message. Deferral of message processing can decrease on-line delays enormously. Note, however, that the processing overhead is deferred, not avoided completely.

This special facility is activated by using a combination of, for instance, the generic `reprocessing` channel and the `expandlimit` keyword. The `expandlimit` keyword takes an integer argument that specifies how many addresses should be accepted in messages coming from the channel before deferring processing. The default value is infinite if the expandlimit keyword is not specified. A value of 0 will force deferred processing on all incoming addresses from the channel.

The `expandlimit` keyword must not be specified on the local channel or the `reprocessing` channel itself; the results of such a specification are unpredictable.

The channel actually used to perform the deferred processing may be specified using the `expandchannel` keyword; the `reprocessing` channel is used by default, if `expandchannel` is not specified, but use of some other reprocessing or processing channel may be useful for special purposes. If a channel for deferred processing is specified via `expandchannel`, that channel should be a reprocessing or processing channel; specification of other sorts of channels may lead to unpredictable results.

The `reprocessing` channel, or whatever channel is used to perform the deferred processing, must be added to the MTA configuration file in order for the `expandlimit` keyword to have any effect. If your configuration was built by the MTA configuration utility, then you should already have a reprocessing channel.

Extraordinarily large lists of recipient addresses are often a characteristic of unsolicited bulk email. The `holdlimit` keyword tells the MTA that messages coming in the channel that result in more than the specified number of recipients should be marked as `.HELD` messages and enqueued to the `reprocess` channel (or to whatever channel is specified via the expandchannel keyword). The files will sit unprocessed in the `reprocess` queue awaiting manual intervention by the MTA postmaster.

Undeliverable Message Notification Times

The `notices`, `nonurgentnotices`, `normalnotices`, and `urgentnotices` keywords control the amount of time an undeliverable message is silently retained in a given channel queue. Messaging Server is capable of returning a series of warning messages to the originator and, if the message remains undeliverable, Messaging Server will eventually return the entire message.

Different return handling for messages of different priorities may be explicitly set using the `nonurgentnotices`, `normalnotices`, or `urgentnotices` keywords. Otherwise, the `notices` keyword values will be used for all messages.

The keyword is followed by a list of up to five monotonically increasing integer values. These values refer to the message ages at which warning messages are sent. The ages have units of days if the MTA option RETURN_UNITS is 0 or not specified in the MTA option file, or hours if the MTA option RETURN_UNITS is 1.

When an undeliverable message attains or exceeds the last listed age, it is returned (bounced). When it attains any of the other ages, a warning notice is sent. The default if no `notices` keyword is given is to use the notices setting. If no setting has been made, then the defaults 3, 6, 9, 12 are used meaning that warning messages are sent when the message attains the ages 3, 6, and 9 days (or hours) and the message is returned after remaining in the channel queue for more than 12 days (or hours).

The syntax for the `notices` keyword uses no punctuation. For example, the default return policy would be expressed as follows:

```
notices 3 6 9 12
```

If you wish to change the notification ages for all of your channels, then the simplest thing to do is to add a `defaults` channel block to the start of the channel block section of your MTA configuration file or to add the `notices` setting to your local channel. For example, the following command adds a `defaults` channel that specifies notification ages for all channels:

```
defaults notices 1 3 6 9 12
```

The `defaults` channel would appear immediately after the first blank line in the MTA configuration file.

Configuring Messages Sent to the Postmaster

A channel program may be unable to deliver a message because of long-term service failures or invalid addresses. When this failure occurs, the MTA channel program returns the message to the sender with an accompanying explanation of why the message was not delivered.

In addition to returning messages, the MTA sometimes sends warnings detailing messages that it has been unable to deliver. This is generally due to timeouts based on the setting of the notices channel keyword, although in some cases channel programs may produce warning messages after failed delivery attempts. The warning messages contain a description of what's wrong and how long delivery attempts will continue. In most cases they also contain the headers and the first few lines of the message in question.

Optionally, a copy of all failed messages and warning messages is sent to the local postmaster. This is useful for monitoring message failures and the state of the various queues, but it can result in lots of traffic for the postmaster.

You can control which messages are sent to the postmaster with the keywords described in Table 8-6.

Table 8-6 Messages Sent to the Postmaster Keywords

Keyword	Description
Returned Messages	**How to handle failure notices for returned messages.**
`sendpost`	Enables sending a copy of all failed messages to the postmaster.
`copysendpost`	Sends a copy of the failure notice to the postmaster unless the originator address on the failing message is blank, in which case, the postmaster gets copies of all failed messages except those messages that are actually themselves bounces or notifications.
`errsendpost`	Sends a copy of the failure notice to the postmaster only when the notice cannot be returned to the originator. If nosendpost is specified, failed messages are never sent to the postmaster.
`nosendpost`	Disables sending a copy of all failed messages to the postmaster.
Warning Messages	**How to handle warning messages.**

Table 8-6 Messages Sent to the Postmaster Keywords

Keyword	Description
warnpost	Enables sending a copy of warning messages to the postmaster. The default, if no keyword is specified, is to send a copy of warnings to the postmaster unless warnings are completely suppressed with a blank Warnings-to: header or a blank envelope From: address.
copywarnpost	Sends a copy of the warning message to the postmaster unless the originator address on the undelivered message is blank.
errwarnpost	Sends a copy of the warning message to the postmaster when the notice cannot be returned to the originator.
nowarnpost	Disables sending a copy of warning messages to the postmaster.
Returned Message Content	**Specifies whether to send entire message or just headers to the postmaster.**
postheadonly	Returns only headers to the postmaster. Restricting the postmaster copy to just the headers adds an additional level of privacy to user mail. However, this by itself does not guarantee message security; postmasters and system managers are typically in a position where the contents of messages can be read using root system privileges, if they so choose.
postheadbody	Returns both the headers and the contents of the message.

Configuring Channel Options

TCP/IP channel option files control various characteristics of TCP/IP channels. Channel option files must be stored in the MTA configuration directory and named *x*_option, where *x* is the name of the channel. For example, */server-instance*/imta/config/tcp_local_option.

The option file consists of one or more keywords and an associated value. For example you can disable mailing list expansion on your server by including the DISABLE_EXPAND keyword in the option file and setting the value to 1.

Other option file keywords allow you to:

- Set a limit on the number of recipients allowed per message (ALLOW_RECIPIENTS_PER_TRANSACTION)

- Set a limit on the number of messages allowed per connection (`ALLOW_TRANSACTIONS_PER_SESSION`)
- Fine tune the type of information logged to the MTA log file (`LOG_CONNECTION`, `LOG_TRANPORTINFO`)
- Specify the maximum number of simultaneous outbound connections that the client channel program allows (`MAX_CLIENT_THREADS`)

For information about all channel option keywords and syntax, see the *Messaging Server Reference Manual*.

Configuring Channel Defaults

Many configurations involve repetition of various channel keywords on all or nearly all channels. Maintaining such a configuration is both tedious and error-prone. To simplify some configurations, you can specify which keywords are defaults for various channels.

For example, the following line in a configuration file indicates that all channel blocks following the line will inherit the keywords specified in the line:

`defaults` *`keyword1 keyword2 keyword3 ...`*

The `defaults` line can be thought of as a special channel block that changes the keyword defaults without actually specifying a channel. The `defaults` line also does not require any additional lines of channel block information (if any are specified they will be ignored).

There is no limit on the number of `defaults` lines that can be specified—the effects of multiple defaults lines are cumulative with the most recently encountered (reading from top to bottom) line having precedence.

It may be useful to unconditionally eliminate the effects of any `defaults` lines starting at some point in the configuration file (at the start of a standalone section of channel blocks in an external file, for example). The nodefaults line is provided for this purpose. For example, inserting the following line in the configuration file nullifies all settings established by any previous defaults channel and returns the configuration to the state that would apply if no defaults had been specified:

```
nodefaults
```

Like regular channel blocks, a blank line must separate each `defaults` or `nodefaults` channel block from other channel blocks. The `defaults` and `nodefaults` channel blocks are the only channel blocks which may appear before the local channel in the configuration file. However, like any other channel block, they must appear after the last rewrite rule.

Configuring Logging for Channels

The MTA provides facilities for logging each message as it is enqueued and dequeued. The `logging` and `nologging` keywords control logging for messages on a per-channel basis. By default, the initial configuration turns on logging for all channels. You can disable logging for a particular channel by substituting the `nologging` keyword in the channel definition.

For more information about logging, see Chapter 12, “Logging and Log Analysis.”

Configuring Debugging for Channels

Some channel programs include optional code to assist in debugging by producing additional diagnostic output. Two channel keywords are provided to enable generation of this debugging output on a per-channel basis. The keywords are `master_debug`, which enables debugging output in master programs, and `slave_debug`, which enables debugging output in slave programs. Both types of debugging output are disabled by default, corresponding to `nomaster_debug` and `noslave_debug`.

When activated, debugging output ends up in the log file associated with the channel program. The location of the log file may vary from program to program. Log files are usually kept in the log directory. Master programs usually have log file names of the form *x*`_master.log`, where *x* is the name of the channel. Slave programs usually have log file names of the form *x*`_slave.log`.

Setting Up Program Delivery

Users might want incoming mail passed to a program instead of to their mailbox. For example, users might want their incoming mail sent to a mail sorting program or to an autoreply agent like Vacation Notice. The `pipe` channel performs delivery of messages using per-user, site-supplied programs.

To facilitate program delivery, you must first register programs as invokable by the pipe channel. You do this by using the imsimta program utility. This utility gives a unique name to each command that you register as invokable by the pipe channel. End users can then specify the program name as a value of their mailprogramdeliveryinfo LDAP attribute.

For example, to add a UNIX command myprocmail as a program that can be invoked by a user, you would first register the command by using the imsimta program utility as shown in the following example. This example registers a program called myprocmail that executes the program procmail with the arguments -d *username* and executes as the user:

```
imsimta program -a -m myprocmail -p procmail -g "-d %s" -e user
```

Make sure the executable exists in the programs directory—*server-instance*/imta/programs—and that the execute permissions are set to "others."

To enable a user to access the program, the user's LDAP entry must contain the following attributes and values:

```
maildeliveryoption: program
mailprogramdeliveryinfo: myprocmail
```

For more information about the imsimta program utility, see the *Messaging Server Reference Manual*.

Alternative delivery programs must conform to the following exit code and command-line argument restrictions:

Exit Code Restrictions. Delivery programs invoked by the pipe channel must return meaningful error codes so that the channel knows whether to dequeue the message, deliver for later processing, or return the message.

If the subprocess exits with an exit code of 0 (EX_OK), the message is presumed to have been delivered successfully and is removed from the MTA queues. If it exits with an exit code of 71, 74, 75, or 79 (EX_OSERR, EX_IOERR, EX_TEMPFAIL, or EX_DB), a temporary error is presumed to have occurred and delivery of the message is deferred. If any other exit code is returned, then the message will be returned to its originator as undeliverable. These exit codes are defined in the system header file sysexits.h.

Command Line Arguments. Delivery programs can have any number of fixed arguments as well as the variable argument, %s, representing the user name for programs executed by the user or username+domain for programs executed by the postmaster, "inetmail." For example, the following command line delivers a recipient's mail using the program procmail:

```
/usr/lib/procmail -d %s
```

Using the Hold Channel

The `hold` channel is used to hold the messages of a recipient temporarily halted from receiving new messages. Messages may be halted because a user's name is being changed, or their mailbox is being moved from one mailhost or domain to another. There may also be other reasons to temporarily halt a user from receiving messages, but those are the most common.

Messages are placed in the hold channel in two ways:

- Setting one of the `maildeliveryoption` values of a user to `hold`. All other `maildeliveryoption` values are ignored (`maildeliveryoption` is a multi-valued attribute), and messages to the user are routed to the hold channel.
- Executing the `hold_slave` program. This program steps through all other channels and moves the existing messages whose recipient(s) matches those specified by the arguments into the hold channel.

Unlike most channels, the `hold` channel master program is not configured to run automatically. Messages queued in the hold channel will remain there until the `hold_master` program is invoked by the administrator.

To migrate user, first mark the user as being moved (use `imadmin modify user` to set `maildeliveryoption` to `hold`). Then invoke `hold_slave` to move any messages already in the other queues to the hold queue. At this point, perform the remaining migration steps. Once you have completed these steps, remove `maildeliveryoption=hold`, and then invoke `hold_master` to enqueue messages to their proper channels.

Using the Conversion Channel

The `conversion` channel performs arbitrary body-part-by-body-part conversions on messages flowing through the MTA. Any subset of the MTA traffic can be selected for conversion and any set of programs or command procedures can be used to perform conversion processing. (The MTA's native conversion facilities are fairly limited, so the ability to call external converters is crucial.) A special `conversion` channel configuration is consulted to choose an appropriate conversion for each body part.

Selecting Traffic for Conversion Processing

Although conversion processing is done using a regular MTA channel program, under normal circumstances this channel is never specified directly either in an address or in an MTA rewrite rule. The MTA controls access to the conversion channel using the CONVERSIONS mapping table in the MTA mappings file (*server_root*/msg-*instance*/imta/config/mappings).

As the MTA processes each message it probes the CONVERSIONS mapping (if one is present) with a string of the form:

IN-CHAN=*source-channel*;OUT-CHAN=*destination-channel*;CONVERT

The *source-channel* is the channel from which the message is coming and *destination-channel* is the channel to which the message is heading. If the mapping produces a result, it should either be the string Yes or No. If Yes is produced, the MTA will divert the message from its regular destination to the conversion channel. If No is produced or if no match is found, the message will be queued to the regular destination channel.

For example, for all messages that do not originate from the tcp_intranet channel and that are going to require conversion processing, the following mapping would then be appropriate:

```
CONVERSIONS

  IN-CHAN=tcp_intranet;OUT-CHAN=tcp_intranet;CONVERT NO
  IN-CHAN=*;OUT-CHAN=tcp_intranet;CONVERT YES
```

Configuration of the Conversion Channel

Configuration of the conversion channel in the MTA configuration file (imta.cnf) is performed by default. An address of the form user@conversion.*localhostname* or user@conversion will be routed through the conversion channel, regardless of what the CONVERSIONS mapping states.

Conversion Control

The actual conversions performed by the conversion channel are controlled by rules specified in the MTA conversion file. This is the file specified by the IMTA_CONVERSION_FILE option in the MTA tailor file. By default, this is the file *server_root*/msg-*instance*/imta/conversions.

The MTA conversion file is a text file containing entries in a format that is modeled after MIME Content-Type parameters. Each entry consists of one or more lines grouped together; each line contains one or more `name=`*`value`*`;` parameter clauses. Quoting rules conform to MIME conventions for Content-Type header line parameters. Every line except the last must end with a semicolon (`;`). A physical line in this file is limited to 252 characters. You can split a logical line into multiple physical lines using the backslash (`\`) continuation character. Entries are terminated either by a line that does not end in a semicolon, one or more blank lines, or both. For example, the following entry specifies that `application/wordperfect5.1` parts in messages sent to the `ims-ms` channel should be converted to MS Word using a hypothetical converter called “convert”:

```
out-chan=l; in-type=application; in-subtype=wordperfect5.1;
  out-type=application; out-subtype=ddif; out-mode=block;
  command="/usr/bin/convert -in=wordp -out=msword <$INPUT_FILE>$OUTPUT_FILE"
```

Understanding Conversions

There are two broad categories of conversions in the MTA, controlled by two corresponding mapping tables and the MTA `conversions` file.

The first category is that of character set, formatting, and labelling conversions performed internally by the MTA. The application of such conversions is controlled by the `CHARSET-CONVERSION` mapping table.

The second category is that of conversions of message attachments using external, third-party programs and site-supplied procedures, such as document convertors. The application of such conversions is controlled by the `CONVERSIONS` mapping table, and messages requiring such conversions are thereby routed through the MTA conversion channel; the conversion channel executes the site-specified external conversion procedure.

The MTA `conversions` file is used to specify the details of external `CONVERSION` table triggered conversions and to specify the details of some internal `CHARSET-CONVERSION` table triggered conversions.

Character Set Conversion and Message Reformatting Mapping

One very basic mapping table in Messaging Server is the character set conversion table. The name of this table is `CHARSET-CONVERSION`. It is used to specify what sorts of channel-to-channel character set conversions and message reformatting should be done.

On many systems there is no need to do character set conversions or message reformatting and therefore this table is not needed. Situations arise, however, where character conversions must be done.

The `CHARSET-CONVERSION` mapping table can also be used to alter the format of messages. Facilities are provided to convert a number of non-MIME formats into MIME. Changes to MIME encodings and structure are also possible. These options are used when messages are being relayed to systems that only support MIME or some subset of MIME. And finally, conversion from MIME into non-MIME formats is provided in a small number of cases.

The MTA will probe the `CHARSET-CONVERSION` mapping table in two different ways. The first probe is used to determine whether or not the MTA should reformat the message and if so, what formatting options should be used. (If no reformatting is specified, the MTA does not bother to check for specific character set conversions.) The input string for this first probe has the general form:

`IN-CHAN=`*in-channel*`;OUT-CHAN=`*out-channel*`;CONVERT`

Here *in-channel* is the name of the source channel (where the message comes from) and *out-channel* is the name of the destination channel (where the message is going). If a match occurs the resulting string should be a comma-separated list of keywords. Table 8-7 lists the keywords.

Table 8-7 CHARSET-CONVERSION Mapping Table Keywords

Keyword	Description
`Always`	Force conversion even when conversion channel is an intermediate destination.
`Appledouble`	Convert other MacMIME formats to Appledouble format.
`Applesingle`	Convert other MacMIME formats to Applesingle format.
`BASE64`	Switch MIME encodings to BASE64.
`Binhex`	Convert other MacMIME formats, or parts including Macintosh type and Mac creator information, to Binhex format.

Table 8-7 CHARSET-CONVERSION Mapping Table Keywords

Keyword	Description
`Block`	"Flatten" any message / rfc822 body part (forwarded message) into a message content part and a header part.
`Bottom`	"Flatten" any message / rfc822 body part (forwarded message) into a message content part and a header part.
`Delete`	"Flatten" any message / rfc822 body part (forwarded message) into a message content part, deleting the forwarded headers.
`Level`	Remove redundant multipart levels from message.
`Macbinary`	Convert other MacMIME formats, or parts including Macintosh type and Macintosh creator information, to Macbinary format.
`No`	Disable conversion.
`QUOTED-PRINTABLE`	Switch MIME encodings to QUOTED-PRINTABLE.
`Record,Text`	Line wrap text / plain parts at 80 characters.
`Record,Text= n`	Line wrap text / plain parts at n characters.
`RFC1154`	Convert message to RFC 1154 format.
`Top`	"Flatten" any message / rfc822 body part (forwarded message) into a header part and a message content part.
`UUENCODE`	Switch MIME encodings to X-UUENCODE.
`Yes`	Enable conversion.

Character Set Conversion

If the MTA probes and finds that the message is to be reformatted, it will proceed to check each part of the message. Any text parts are found and their character set parameters are used to generate the second probe. Only when the MTA has checked and found that conversions may be needed does it ever perform the second probe. The input string in this second case looks like this:

`IN-CHAN=`*in-channel*`;OUT-CHAN=`*out-channel*`;IN-CHARSET=`*in-char-set*

The *in-channel* and *out-channel* are the same as before, and the *in-char-set* is the name of the character set associated with the particular part in question. If no match occurs for this second probe, no character set conversion is performed (although message reformatting, for example, changes to MIME structure, may be performed in accordance with the keyword matched on the first probe). If a match does occur it should produce a string of the form:

`OUT-CHARSET=`*out-char-set*

Here *out-char-set* specifies the name of the character set to which the `in-char-set` should be converted. Note that both of these character sets must be defined in the character set definition table, `charsets.txt`, located in the MTA table directory. No conversion will be done if the character sets are not properly defined in this file. This is not usually a problem since this file defines several hundred character sets; most of the character sets in use today are defined in this file. See the description of the `imsimta chbuild` (UNIX and NT) utility for further information on the `charsets.txt` file.

If all the conditions are met, the MTA will proceed to build the character set mapping and do the conversion. The converted message part will be relabelled with the name of the character set to which it was converted.

Message Reformatting

As described above, the `CHARSET-CONVERSION` mapping table is also used to effect the conversion of attachments between MIME and several proprietary mail formats.

The following sections give examples of some of the other sorts of message reformatting which can be affected with the `CHARSET-CONVERSION` mapping table.

Non-MIME Binary Attachment Conversion

Mail in certain non-standard (non-MIME) formats; for example, mail in certain proprietary formats or mail from the Microsoft Mail (MSMAIL) SMTP gateway is automatically converted into MIME format if `CHARSET-CONVERSION` is enabled for any of the channels involved in handling the message. If you have a `tcp_local` channel then it is normally the incoming channel for messages from a Microsoft Mail SMTP gateway, and the following will enable the conversion of messages delivered to your local users:

```
CHARSET-CONVERSION
   IN-CHAN=tcp_local;OUT-CHAN=ims-ms;CONVERT         Yes
```

Alternatively, to cover every channel you can simply specify `OUT-CHAN=*` instead of `OUT-CHAN=ims-ms`. However, this may bring about an increase in message processing overhead as all messages coming in the `tcp_local` channel will now be scrutinized instead of just those bound to specific channels.

More importantly, such indiscriminate conversions might place your system in the dubious and frowned upon position of converting messages—not necessarily your own site's—which are merely passing through your system, a situation in which you should merely be acting as a transport and not necessarily altering anything beyond the message envelope and related transport information.

To convert MIME into the format Microsoft Mail SMTP gateway understands, use a separate channel in your MTA configuration for the Microsoft Mail SMTP gateway; for example, `tcp_msmail`, and put the following in the mappings. file:

```
CHARSET-CONVERSION

   IN-CHAN=*;OUT-CHAN=tcp_msmail;CONVERT        RFC1154
```

Relabelling MIME Headers

Some user agents or gateways may emit messages with MIME headers that are less informative than they might be, but that nevertheless contain enough information to construct more precise MIME headers. Although the best solution is to properly configure such user agents or gateways, if they are not under your control, you can instead ask the MTA to try to reconstruct more useful MIME headers.

If the first probe of the `CHARSET-CONVERSION` mapping table yields a `Yes` or `Always` keyword, then the MTA will check for the presence of a `conversions` file. If a conversions file exists, then the MTA will look in it for an entry with `RELABEL=1` and if it finds such an entry, the MTA will then perform any MIME relabelling specified in the entry.

For example, the combination of a CHARSET-CONVERSION table and MTA conversions file entries such as the following will result in messages that arrive on the tcp_local channel and are routed to the ims-ms channel, and that arrive originally with MIME labelling of application/octet-stream but have a filename parameter with the extension ps or msw, being relabelled as application/postscript or application/msword, respectively. (Note that this more precise labelling is what the original user agent or gateway should have performed itself.)

CHARSET CONVERSION TABLE

```
CHARSET-CONVERSION

   IN-CHAN=tcp_local;OUT-CHAN=mr_local;CONVERT            Yes
```

MTA CONVERSIONS FILE ENTRIES

```
out-chan=ims-ms; in-type=application; in-subtype=octet-stream;
  in-parameter-name-0=name; in-parameter-value-0=*.ps;
  out-type=application; out-subtype=postscript;
     parameter-copy-0=*; relabel=1

out-chan=ims-ms; in-type=application; in-subtype=octet-stream;
  in-parameter-name-0=name; in-parameter-value-0=*.msw;
  out-type=application; out-subtype=msword;
     parameter-copy-0=* relabel=1
```

MacMIME Format Conversions

Macintosh files have two parts, a resource fork that contains Macintosh specific information, and a data fork that contains data usable on other platforms. This introduces an additional complexity when transporting Macintosh files, as there are four different formats in common use for transporting the Macintosh file parts. Three of the formats, Applesingle, Binhex, and Macbinary, consist of the Macintosh resource fork and Macintosh data fork encoded together in one piece. The fourth format, Appledouble, is a multipart format with the resource fork and data fork in separate parts. Appledouble is hence the format most likely to be useful on non-Macintosh platforms, as in this case the resource fork part may be ignored and the data fork part is available for use by non-Macintosh applications. But the other formats may be useful when sending specifically to Macintoshes.

The MTA can convert between these various Macintosh formats. The CHARSET-CONVERSION keywords Appledouble, Applesingle, Binhex, or Macbinary tell the MTA to convert other MacMIME structured parts to a MIME structure of multipart/appledouble, application/applefile, application/mac-binhex40, or application/macbinary, respectively. Further, the Binhex or Macbinary keywords also request conversion to the specified format of non-MacMIME format parts that do nevertheless contain X-MAC-TYPE and X-MAC-CREATOR parameters on the MIME Content-type: header. The CHARSET-CONVERSION keyword Block tells the MTA to extract just the data fork from MacMIME format parts, discarding the resource fork; (since this loses information, use of Appledouble instead is generally preferable).

For example, the following CHARSET-CONVERSION table would tell the MTA to convert to Appledouble format when delivering to the ims-ms channel.

```
CHARSET-CONVERSION

  IN-CHAN=*;OUT-CHAN=l;CONVERT            Appledouble
```

The conversion to Appledouble format would only be applied to parts already in one of the MacMIME formats.

When doing conversion to Appledouble or Block format, the MAC-TO-MIME-CONTENT-TYPES mapping table may be used to indicate what specific MIME label to put on the data fork of the Appledouble part, or the Block part, depending on what the Macintosh creator and Macintosh type information in the original Macintosh file were. Probes for this table have the form format|type|creator|filename where format is one of SINGLE, BINHEX or MACBINARY, where type and creator are the Macintosh type and Macintosh creator information in hex, respectively, and where filename is the filename.

For example, to convert to Appledouble when sending to the ims-ms channel and when doing so to use specific MIME labels for any MS Word or PostScript documents converted from MACBINARY or BINHEX parts, appropriate tables might be:

```
CHARSET-CONVERSION

  IN-CHAN=*;OUT-CHAN=ims-ms;CONVERT     Appledouble

MAC-TO-MIME-CONTENT-TYPES

! PostScript
    MACBINARY|45505346|76677264|*     APPLICATION/POSTSCRIPT$Y
    BINHEX|45505346|76677264|*        APPLICATION/POSTSCRIPT$Y
! Microsoft Word
    MACBINARY|5744424E|4D535744|*     APPLICATION/MSWORD$Y
    BINHEX|5744424E|4D535744|*        APPLICATION/MSWORD$Y
```

Note that the template (right hand side) of the mapping entry must have the $Y flag set in order for the specified labelling to be performed. Sample entries for additional types of attachments may be found in the file mac_mappings.sample in the MTA table directory.

If you wish to convert non-MacMIME format parts to Binhex or Macbinary format, such parts need to have X-MAC-TYPE and X-MAC-CREATOR MIME Content-type: parameter values provided. Note that MIME relabelling can be used to force such parameters onto parts that would not otherwise have them.

Service Conversions

The MTA's conversion service facility may be used to process with site-supplied procedures a message so as to produce a new form of the message. Unlike either the sorts of CHARSET-CONVERSION operations discussed above or the conversion channel, which operate on the content of individual MIME message parts, conversion services operate on entire MIME message parts (MIME headers and content) as well as entire MIME messages. Also, unlike other CHARSET-CONVERSION operations or conversion channel operations, conversion services are expected to do their own MIME disassembly, decoding, re-encoding, and reassembly.

Like other CHARSET-CONVERSION operations, conversion services are enabled through the CHARSET-CONVERSION mapping table. If the first probe of the CHARSET-CONVESION mapping table yields a Yes or Always keyword, then the MTA will check for the presence of an MTA conversions file. If a conversions file exists, then the MTA will look in it for an entry specifying a SERVICE-COMMAND, and if it finds such an entry, execute it. The conversions file entries should have the form:

```
in-chan=channel-pattern;
  in-type=type-pattern; in-subtype=subtype-pattern;
  service-command=command
```

Of key interest is the command string. This is the command that should be executed to perform a service conversion (for example, invoke a document converter). The command must process an input file containing the message text to be serviced and produce as output a file containing the new message text. On UNIX, the command must exit with a 0 if successful and a non-zero value otherwise.

Environment variables are used to pass the names of the input and output files as well as the name of a file containing the list of the message's envelope recipient addresses. The names of these environment variables are:

- INPUT_FILE - Name of the input file to process
- OUTPUT_FILE - Name of the output file to produce
- INFO_FILE - Name of the file containing envelope recipient addresses

The values of these three environment variables may be substituted into the command line by using standard command line substitution: that is, preceding the variable's name with a dollar character on UNIX.

Chapter 9

Mail Filtering and Access Control

This chapter discusses how to control access to mail services and how to filter mail using mapping tables and server-side rules (SSR).

You might want to reject messages from (or to) certain users at the system level, or to institute more complex restrictions of message traffic between certain users, or to allow users to set up filters on their own incoming messages (including rejecting messages based on contents of the message headers).

If envelope-level controls are desired, you can use mapping tables to filter mail. If header-based controls are desired or if users wish to implement their own personalized controls, the more general mail filtering approach using server-side rules is likely appropriate.

This chapter is divided into two parts:

PART 1. MAPPING TABLES

PART 2. MAILBOX FILTERS

PART 1. MAPPING TABLES

Part 1 contains the following sections:

- Controlling Access with Mapping Tables
- When Access Controls Are Applied
- Testing Access Control Mappings
- Adding SMTP Relaying
- Configuring SMTP Relay Blocking
- Handling Large Numbers of Access Entries

- Mapping Table Flags

Controlling Access with Mapping Tables

You can control access to your mail services by configuring mapping tables. Mapping tables allow you to control who can or cannot send mail, receive mail, or both. For general information on the format and usage of the mapping files, see the *Messaging Server Reference Manual*.

Table 9-1 lists the mapping tables described in this section.

Table 9-1 Mapping Tables

Mapping Table	Description
SEND_ACCESS	Used to block incoming connections based on envelope From address, envelope To address, source and destination channels. The To address is checked after rewriting, alias expansion, and so on, have been performed See "SEND_ACCESS and ORIG_SEND_ACCESS Tables," on page 203.
ORIG_SEND_ACCESS	Used to block incoming connections based on envelope From address, envelope To address, source and destination channels. The To address is checked after rewriting but before alias expansion. See "SEND_ACCESS and ORIG_SEND_ACCESS Tables," on page 203.
MAIL_ACCESS	Used to block incoming connections based on combined information found in SEND_ACCESS and PORT_ACCESS tables: that is, the channel and address information found in SEND_ACCESS combined with the IP address and port number information found in PORT_ACCESS. See "MAIL_ACCESS and ORIG_MAIL_ACCESS Mapping Tables," on page 205.
ORIG_MAIL_ACCESS	Used to block incoming connections based on combined information found in ORIG_SEND_ACCESS and PORT_ACCESS tables: that is, the channel and address information found in ORIG_SEND_ACCESS combined with the IP address and port number information found in PORT_ACCESS. See "MAIL_ACCESS and ORIG_MAIL_ACCESS Mapping Tables," on page 205.

Table 9-1 Mapping Tables

Mapping Table	Description
FROM_ACCESS	Used to filter mail based on envelope From addresses. Use this table if the To address is irrelevant. See “FROM_ACCESS Mapping Table,” on page 207
PORT_ACCESS	Used to block incoming connections based on IP number. See “PORT_ACCESS Mapping Table,” on page 209.

The MAIL_ACCESS and ORIG_MAIL_ACCESS mappings are the most general, having available not only the address and channel information available to SEND_ACCESS and ORIG_SEND_ACCESS, but also any information that would be available via the PORT_ACCESS mapping table, including IP address and port number information.

SEND_ACCESS and ORIG_SEND_ACCESS Tables

You can use the SEND_ACCESS and ORIG_SEND_ACCESS mapping tables to control who can or cannot send mail, receive mail, or both. The access checks have available a message’s envelope From: address and envelope To: addresses, and knowledge of what channel the message came in, and what channel it would attempt to go out.

If a SEND_ACCESS or ORIG_SEND_ACCESS mapping table exists, then for each recipient of every message passing through the MTA, the MTA will scan the table with a string of the following form (note the use of the vertical bar character, |):

src-channel|*from-address*|*dst-channel*|*to-address*

The *src-channel* is the channel queueing the message; *from-address* is the address of the message's originator; *dst-channel* is the channel to which the message will be queued; and *to-address* is the address to which the message is addressed. Use of an asterisk in any of these four fields causes that field to match any channel or address, as appropriate.

The addresses here are envelope addresses; that is, envelope From: address and envelope To: address. In the case of SEND_ACCESS, the envelope To: address is checked after rewriting, alias expansion, etc., have been performed; in the case of ORIG_SEND_ACCESS the originally specified envelope To: address is checked after rewriting, but before alias expansion.

If the search string matches a pattern (that is, the left-hand side of an entry in the table), then the resulting output of the mapping is checked. If the output contains the flags $Y or $y, then the enqueue for that particular To: address is permitted. If the output contains any of the flags $N, $n, $F, or $f, then the enqueue to that particular address is rejected. In the case of a rejection, optional rejection text may be supplied in the mapping output. This string will be included in the rejection error the MTA issues. If no string is output (other than the $N, $n, $F, or $f flag), then default rejection text will be used. For descriptions of additional flags, see "Mapping Table Flags," on page 227.

In the following example, mail sent from UNIX user agents such as mail, Pine, and so on, originates from the local, `l`, channel and messages to the Internet go out a TCP/IP channel of some sort. Suppose that local users, with the exception of the postmaster, are not allowed to send mail to the Internet but can receive mail from there. Then the `SEND_ACCESS` mapping table shown in Figure 9-1 is one possible way to enforce this restriction. In the mapping table, the local host name is assumed to be `sesta.com`. In the channel name "tcp_*", a wild card is used so as to match any possible TCP/IP channel name (for example, `tcp_local`).

In the rejection message, dollar signs are used to quote spaces in the message. Without those dollar signs, the rejection would be ended prematurely and only read "Internet" instead of "Internet postings are not permitted." Note that this example ignores other possible sources of "local" postings such as from PC-based mail systems or from POP or IMAP clients.

Figure 9-1 SEND_ACCESS Mapping Table

```
SEND_ACCESS

  *|postmaster@sesta.com|*|*    $Y
  *|*|*|postmaster@sesta.com    $Y
  l|*@sesta.com|tcp_*|*         $NInternet$ postings$ are$ not$ \
    permitted
```

NOTE The client attempting to send the message determines whether the MTA rejection error text is actually presented to the user who attempted to send the message. If SEND_ACCESS is used to reject an incoming SMTP message, the MTA merely issues an SMTP rejection code including the optional rejection text; it is up to the sending SMTP client to use that information to construct a bounce message to send back to the original sender.

MAIL_ACCESS and ORIG_MAIL_ACCESS Mapping Tables

The MAIL_ACCESS mapping table is a superset of the SEND_ACCESS and PORT_ACCESS mapping tables. It combines both the channel and address information of SEND_ACCESS with the IP address and port number information of PORT_ACCESS. Similarly, the ORIG_MAIL_ACCESS mapping table is a superset of the ORIG_SEND_ACCESS and PORT_ACCESS mapping tables. The format for the probe string for MAIL_ACCESS is:

port-access-probe-info | *app-info* | *submit-type* | *send_access-probe-info*

Similarly, the format for the probe string for ORIG_MAIL_ACCESS is:

port-access-probe-info | *app-info* | *submit-type* | *orig_send_access-probe-info*

Here *port-access-probe-info* consists of all the information usually included in a PORT_ACCESS mapping table probe in the case of incoming SMTP messages; otherwise, it is blank. *app-info* is usually SMTP in the case of messages submitted via SMTP; otherwise it is blank. *submit-type* may be one of MAIL, SEND, SAML, or SOML, corresponding to how the message was submitted into Messaging Server. Normally the value is MAIL, meaning it was submitted as a message; SEND, SAML, or SOML can occur in the case of broadcast requests (or combined broadcast / message requests) submitted to the SMTP server. And for the MAIL_ACCESS mapping, *send-access-probe-info* consists of all the information usually included in a SEND_ACCESS mapping table probe. Similarly for the ORIG_MAIL_ACCESS mapping, *orig-send-access-probe-info* consists of all the information usually included in an ORIG_SEND_ACCESS mapping table probe.

Having the incoming TCP / IP connection information available in the same mapping table as the channel and address information makes it more convenient to impose certain sorts of controls, such as enforcing what envelope From: addresses are allowed to appear in messages from particular IP addresses. This can be

desirable to limit cases of email forgery, or to encourage users to configure their POP and IMAP clients' From: address appropriately. For example, a site that wishes to allow the envelope From: address `vip@siroe.com` to appear only on messages coming from the IP address 1.2.3.1 and 1.2.3.2, and to ensure that the envelope From: addresses on messages from any systems in the 1.2.0.0 subnet are from `siroe.com`, might use a `MAIL_ACCESS` mapping table as shown in Figure 9-2.

Figure 9-2 MAIL_ACCESS Mapping Table

```
MAIL_ACCESS

  ! Entries for vip's two systems
  !
  TCP|*|25|1.2.3.1|*|SMTP|MAIL|tcp_*|vip@siroe.com|*|*  $Y
  TCP|*|25|1.2.3.2|*|SMTP|MAIL|tcp_*|vip@siroe.com|*|*  $Y
  !
  ! Disallow attempts to use vip's From: address from other
  ! systems
  !
  TCP|*|25|*|*|SMTP|MAIL|tcp_*|vip@siroe.com|*|*  \
      $N500$ Not$ authorized$ to$ use$ this$ From:$ address
  !
  ! Allow sending from within our subnet with siroe.com From:
  ! addresses
  !
  TCP|*|25|1.2.*.*|*|SMTP|MAIL|tcp_*|*@siroe.com|*|*  $Y
  !
  ! Allow notifications through
  !
  TCP|*|25|1.2.*.*|*|SMTP|MAIL|tcp_*||*|*  $Y
  !
  ! Block sending from within our subnet with non-siroe.com
  ! addresses
  !
  TCP|*|25|1.2.*.*|*|SMTP|MAIL|tcp_*|*|*|*  \
      $NOnly$ siroe.com$ From:$ addresses$ authorized
```

FROM_ACCESS Mapping Table

The FROM_ACCESS mapping table may be used to control who can send mail, or to override purported From: addresses with authenticated addresses, or both.

The input probe string to the FROM_ACCESS mapping table is similar to that for a MAIL_ACCESS mapping table, minus the destination channel and address, and with the addition of authenticated sender information, if available. Thus, if a FROM_ACCESS mapping table exists, then for each attempted message submission, Messaging Server will search the table with a string of the form (note the use of the vertical bar character, |):

port-access-probe-info|*app-info*|*submit-type*|*src-channel*|*from-address*|*auth-from*

Here *port-access-probe-info* consists of all the information usually included in a PORT_ACCESS mapping table probe in the case of incoming SMTP messages; otherwise, it is blank. *app-info* is usually SMTP in the case of messages submitted via SMTP; otherwise, it is blank. *submit-type* may be one of MAIL, SEND, SAML, or SOML, corresponding to how the message was submitted into the MTA. Normally the value is MAIL, meaning it was submitted as a message; SEND, SAML, or SOML can occur in the case of broadcast requests (or combined broadcast / message requests) submitted to the SMTP server. *src-channel* is the channel originating the message (i.e., queueing the message); *from-address* is the address of the message's purported originator; and *auth-from* is the authenticated originator address, if such information is available, or blank if no authenticated information is available.

If the probe string matches a pattern (that is, the left-hand side of an entry in the table), the resulting output of the mapping is checked. If the output contains the flags $Y or $y, then the enqueue for that particular To: address is permitted. If the output contains any of the flags $N, $n, $F, or $f, then the enqueue to that particular address is rejected. In the case of a rejection, optional rejection text may be supplied in the mapping output. This string will be included in the rejection error Messaging Server issues. If no string is output (other than the $N, $n, $F, or $f flag), then default rejection text will be used. For descriptions of additional flags, see“Mapping Table Flags,” on page 227.

Besides determining whether to allow a message to be submitted based on the originator, FROM_ACCESS can also be used to alter the envelope `From:` address via the `$J` flag, or to modify the effect of the `authrewrite` channel keyword (adding a `Sender:` header address on an accepted message) via the `$K` flag. For instance, this mapping table can be used to cause the original envelope `From:` address to simply be replaced by the authenticated address:

```
FROM_ACCESS

  *|SMTP|*|tcp_local|*|       $Y
  *|SMTP|*|tcp_local|*|*      $Y$J$3
```

When using the `FROM_ACCESS` mapping table to modify the effect on having `authrewrite` set to a nonzero value on some source channel, it is not necessary to use `FROM_ACCESS` if the authenticated address is going to be used verbatim.

For example, with `authrewrite 2` set on the `tcp_local` channel, the following `FROM_ACCESS` mapping table would not be necessary because `authrewrite` alone is sufficient to get this effect (adding the authenticated address verbatim):

```
FROM_ACCESS

  *|SMTP|*|tcp_local|*|     $Y
  *|SMTP|*|tcp_local|*|*    $Y$K$3
```

However, the real purpose of `FROM_ACCESS` is to permit more complex and subtle alterations, as shown in Figure 9-3. The `authrewrite` keyword alone is appropriate if you want to add a `Sender:` header line (showing the SMTP AUTH authenticated submitter address) to incoming messages. However, suppose you want to force the addition of such a `Sender:` header line to incoming messages only if the SMTP AUTH authenticated submitter address differs from the envelope `From:` address (that is, not bother to add a `Sender:` header line if the addresses match), and suppose further that you wish the SMTP AUTH and envelope `From:` addresses will not be considered to differ merely because the envelope `From:` includes optional subaddress information.

Figure 9-3 FROM_ACCESS Mapping Table

```
FROM_ACCESS

  ! If no authenticated address is available, do nothing
  *|SMTP|*|tcp_local|*|              $Y
  ! If authenticated address matches envelope From:, do nothing
  *|SMTP|*|tcp_local|*|$2*           $Y
  ! If authenticated address matches envelope From: sans
  ! subaddress, do nothing
  *|SMTP|*|tcp_local|*+*@*|$2*@$4*   $Y
  ! Fall though to...
  ! ...authenticated address present, but didn't match, so force
  ! Sender: header
  *|SMTP|*|tcp_local|*|*             $Y$K$3
```

PORT_ACCESS Mapping Table

The Dispatcher is able to selectively accept or reject incoming connections based on IP address and port number. At Dispatcher startup time, the Dispatcher will look for a mapping table named PORT_ACCESS. If present, the Dispatcher will format connection information in the following form:

TCP|*server-address*|*server-port*|*client-address*|*client-port*

The Dispatcher tries to match against all PORT_ACCESS mapping entries. If the result of the mapping contains $N or $F, the connection will be immediately closed. Any other result of the mapping indicates that the connection is to be accepted. $N or $F may optionally be followed by a rejection message. If present, the message will be sent back down the connection just prior to closure. Note that a CRLF terminator will be appended to the string before it is sent back down the connection.

The flag $< followed by an optional string causes Messaging Server to send the string to syslog (UNIX) or to the event log (NT) if the mapping probe matches. The flag $> followed by an optional string causes Messaging Server to send the string as to syslog (UNIX) or to the event log (NT) if access is rejected. If bit 1 of the LOG_CONNECTION MTA option is set and the $N flag is set so that the connection is rejected, then also specifying the $T flag will cause a “T” entry to be written to the connection log. If bit 4 of the LOG_CONNECTION MTA option is set, then

site-supplied text may be provided in the PORT_ACCESS entry to include in the “C” connection log entries. To specify such text, include two vertical bar characters in the right-hand side of the entry, followed by the desired text. Table 9-2 lists the available flags.

Table 9-2 PORT_ACCESS Mapping Flags

Flag	Description
$Y	Allow access.
Flags with arguments, in argument reading order+	
$< string	Send string to syslog (UNIX) or to the event log (NT) if probe matches.
$> string	Send string to syslog (UNIX) or to the event log (NT) if access is rejected.
$N string	Reject access with the optional error text string
$F string	Synonym for $N string; that is, reject access with the optional error text string
$T text	If bit 1 of the LOG_CONNECTION MTA option is set and the $N flag is set so that the connection is rejected, then $T causes a “T” entry to be written to the connection log; the optional text (which must appear subsequent to two vertical bar characters) may be included in the connection log entry.

+To use multiple flags with arguments, separate the arguments with the vertical bar character, |, placing the arguments in the order listed in this table.

For example, the following mapping will only accept SMTP connections (to port 25, the normal SMTP port) from a single network, except for a particular host singled out for rejection without explanatory text:

```
PORT_ACCESS

  TCP|*|25|192.123.10.70|*  $N500
  TCP|*|25|192.123.10.*|*   $Y
  TCP|*|25|*|*              $N500$ Bzzzt$ thank$ you$ for$ \
     playing.
```

Note that you will need to restart the Dispatcher after making any changes to the PORT_ACCESS mapping table so that the Dispatcher will see the changes. (If you are using a compiled MTA configuration, you will first need to recompile your configuration to get the change incorporated into the compiled configuration.)

The PORT_ACCESS mapping table is specifically intended for performing IP-based rejections. For more general control at the email address level, the SEND_ACCESS or MAIL_ACCESS mapping table, might be more appropriate.

Limiting Specified IP Address Connections to the MTA

A particular IP address can be limited to how often it connects to the MTA by using the shared library, conn_throttle.so in the Port Access mapping table. Limiting connections by particular IP addresses may be useful for preventing excessive connections used in denial-of-service attacks.

conn_throttle.so is a shared library used in a PORT_ACCESS mapping table to limit MTA connections made from IP addresses too frequently. All configuration options are specified as parameters to the connection throttle shared library as follows:

$[<*Server_Root*>/lib/conn_throttle.so,throttle,IP-address,max-rate]

IP-address is the dotted-decimal address of the remote system. max-rate is the connections per minute that shall be the enforced maximum rate for this IP-address.

The routine name throttle_p may be used instead of throttle for a penalizing version of the routine. throttle_p will deny connections in the future if they've connected too many times in the past. If the maximum rate is 100, and 250 connections have been attempted in the past minute, not only will the remote site be blocked after the first 100 connections in that minute, but they'll also be blocked during the second minute. In other words, after each minute, max-rate is deducted from the total number of connections attempted and the remote system is blocked as long as the total number of connections is greater than the maximum rate.

If the IP-address specified has not exceeded the maximum connections per minute rate, the shared libray callout will fail.

If the rate has been exceeded, the callout will succeed, but will return nothing. This is done in a $C / $E combination as in the example:

```
PORT_ACCESS

  TCP|*|25|*|*   \
$C$[<Server_Root>/lib/conn_throttle.so,throttle,$1,10]   \
$N421$ Connection$ not$ accepted$ at$ this$ time$E
```

Where,

`$C` continues the mapping process starting with the next table entry; uses the output string of this entry as the new input string for the mapping process.

`$[<`*Server_Root*`>/lib/conn_throttle.so,throttle,$1,10]` is the library call with `throttle` as the library routine, `$1` as the server IP Address, and 10 the connections per minute threshold.

`$N421$ Connection$ not$ accepted$ at$ this$ time` rejects access and returns the 421 SMTP code (transient negative completion) along with the message "Connection not accepted at this time."

`$E` ends the mapping process now. It uses the output string from this entry as the final result of the mapping process.

When Access Controls Are Applied

Messaging Server checks access control mappings as early as possible. Exactly when this happens depends upon the email protocol in use—when the information that must be checked becomes available.

For the SMTP protocol, a `FROM_ACCESS` rejection occurs in response to the MAIL FROM: command, before the sending side can send the recipient information or the message data. A `SEND_ACCESS` or `MAIL_ACCESS` rejection occurs in response to the RCPT TO: command, before the sending side gets to send the message data. If an SMTP message is rejected, Messaging Server never accepts or sees the message data, thus minimizing the overhead of performing such rejections.

If multiple access control mapping tables exist, Messaging Server checks them all. That is, a `FROM_ACCESS`, a `SEND_ACCESS`, an `ORIG_SEND_ACCESS`, a `MAIL_ACCESS`, and `ORIG_MAIL_ACCESS` mapping tables may all be in effect.

Testing Access Control Mappings

The `imsimta test -rewrite` utility—particularly with the `-from`, `-source_channel`, and `-destination_channel` options—can be useful in testing access control mappings. For example, Figure 9-4 shows a sample `SEND_ACCESS` mapping table and the resulting probe.

Figure 9-4 Sample SEND_ACCESS Mapping Table and Probe

MAPPING TABLE

```
SEND_ACCESS

  tcp_local|friendly@siroe.com|l|User@sesta.com     $Y
  tcp_local|unwelcome@varrius.com|l|User@sesta.com  $NGo$ away!
```

PROBE

```
$ TEST/REWRITE/FROM="friendly@siroe.com" -
_$  /SOURCE=tcp_local/DESTINATION=l User@sesta.com
...
Submitted address list:
  l
    User (SESTA.COM) *NOTIFY FAILURES* *NOTIFY DELAYS* Submitted
notifications list:


$ TEST/REWRITE/FROM="unwelcome@varrius.com" -
_$  /SOURCE=tcp_local/DESTINATION=l User@sesta.com
...
Submitted address list:
Address list error -- 5.7.1 Go away! User@sesta.com

Submitted notifications list:
```

Adding SMTP Relaying

The iPlanet Messaging Server is by default configured to block attempted SMTP relays; that is, it rejects attempted message submissions to external addresses from unauthenticated external sources (external systems are any other system than the host on which the server itself resides). This default configuration is quite aggressive in blocking SMTP relaying in that it considers all other systems to be external systems.

IMAP and POP clients that attempt to submit messages via the iPlanet Messaging Server system's SMTP server destined for external addresses, and who do not authenticate using SMTP AUTH (SASL), will find their submission attempts rejected. Thus, you will likely want to modify your configuration so that it recognizes your own internal systems and subnets from which relaying should always be accepted.

Which systems and subnets are recognized as internal is normally controlled by the INTERNAL_IP mapping table, which may be found in the *<InstanceRoot>*`/imta/config/mappings` file.

For instance, on an iPlanet Messaging Server system whose IP address is 123.45.67.89, the default INTERNAL_IP mapping table would appear as follows:

```
INTERNAL_IP

   $(123.45.67.89/32)   $Y
   127.0.0.1   $Y
   *   $N
```

Here the initial entry, using the `$(IP-pattern/signicant-prefix-bits)` syntax, is specifying that any IP address that matches all 32 bits of 123.45.67.89 should match and be considered internal. The second entry recognizes the loopback IP address 127.0.0.1 as internal. The final entry specifies that all other IP addresses should not be considered internal. Note that all entries must be preceded by at least one space.

You may add additional entries by specifying additional IP addresses or subnets before the final $N entry. These entries must specify an IP address or subnet (using the $(.../...) syntax to specify a subnet) on the left side and $Y on the right side. Or you may modify the existing $(.../...) entry to accept a more general subnet.

For instance, if this same sample site has a class-C network, that is, it owns all of the 123.45.67.0 subnet, then the site would want to modify the initial entry by changing the number of bits used in matching the address. In the mapping table below, we change from 32 bits to 24 bits. This allows all clients on the class-C network to relay mail through this SMTP relay server.

```
INTERNAL_IP

   $(123.45.67.89/24)   $Y
   127.0.0.1   $Y
   *   $N
```

Or if the site owns only those IP addresses in the range 123.45.67.80-123.45.67.99, then the site would want to use:

```
INTERNAL_IP

! Match IP addresses in the range 123.45.67.80-123.45.67.95
   $(123.45.67.80/28)   $Y
! Match IP addresses in the range 123.45.67.96-123.45.67.99
   $(123.45.67.96/30)   $Y
   127.0.0.1   $Y
   *   $N
```

Note that the <*InstanceRoot*>`/imsimta test -match` utility can be useful for checking whether an IP address matches a particular $(.../...) test condition. The <*InstanceRoot*>`/imsimta test -mapping` utility can be more generally useful in checking that your INTERNAL_IP mapping table returns the desired results for various IP address inputs.

After modifying your INTERNAL_IP mapping table, be sure to issue the <*InstanceRoot*>`/imsimta restart` command (if you are not running with a compiled configuration) or the <*InstanceRoot*>`/imsimta refresh` command (if you are running with a compiled configuration) so that the changes take effect.

Further information on the mapping file and general mapping table format, as well as information on `imsimta` command line utilities, can be found in the iPlanet Messaging Server Reference Manual.

Allowing SMTP Relaying for External Sites

All internal IP addresses should be added to the INTERNAL_IP mapping table as discussed above. If you have friendly or companion systems / sites from which you wish to allow SMTP relaying, the simplest approach is to include them along with your true internal IP addresses in your INTERNAL_IP mapping table.

If you don't wish to consider these as true internal systems / sites, (for instance, if for logging or other control purposes you wish to distinguish between *true internal systems* versus the *friendly non-internal systems with relay privileges*), there are other ways to configure the system.

One approach is to set up a special channel for receiving messages from such friendly systems. Do this by creating a *tcp_friendly* channel akin to your existing *tcp_internal* channel with official host name *tcp_friendly-daemon*, and a FRIENDLY_IP mapping table akin to your INTERNAL_IP mapping table that lists the friendly system IP addresses. Then right after the current rewrite rule:

```
! Do mapping lookup for internal IP addresses []
$E$R${INTERNAL_IP,$L}$U%[$L]@tcp_intranet-daemon
```

add a new rewrite rule:

```
! Do mapping lookup for "friendly", non-internal IP addresses []
$E$R${FRIENDLY_IP,$L}$U%[$L]@tcp_friendly-daemon
```

An alternate approach is to add to your ORIG_SEND_ACCESS mapping table above the final $N entry, new entries of the form

```
  tcp_local|*@siroe.com|tcp_local|*    $Y
```

where siroe.com is the name of a friendly domain, and to add an ORIG_MAIL_ACCESS mapping table of the form:

```
ORIG_MAIL_ACCESS

   TCP|*|25|$(match-siroe.com-IP-addresses)|*|SMTP|MAIL|    \
tcp_local|*@siroe.com|tcp_local|*     $Y
   TCP|*|*|*|*|SMTP|MAIL|tcp_local|*|tcp_local|*    $N
```

table, where the $(...) IP address syntax is the same syntax described in the previous section. The ORIG_SEND_ACCESS check will succeed as long as the address is ok, so we can go ahead and also do the ORIG_MAIL_ACCESS check which is more stringent and will only succeed if the IP address also corresponds to an siroe.com IP address.

Configuring SMTP Relay Blocking

You can use access control mappings to prevent people from relaying SMTP mail through your Messaging Server system. For example, you can prevent people from using your mail system to relay junk mail to hundreds or thousands of Internet mailboxes.

By default, Messaging Server prevents all SMTP relaying activity, including relaying by local POP and IMAP users.

Blocking unauthorized relaying while allowing it for legitimate local users requires configuring Messaging Server to know how to distinguish between the two classes of users. For example, local users using POP or IMAP depend upon Messaging Server to act as an SMTP relay.

To prevent SMTP relay, you must be able to:

- Differentiate Between Internal and External Mail
- Differentiate Authenticated Users' Mail
- Prevent Mail Relay

To enable SMTP relay by internal hosts and clients, you must add your “internal” IP addresses or subnets to the INTERVAL_IP mapping table.

Differentiate Between Internal and External Mail

In order to block mail relaying activities, you must first be able to differentiate between internal mail originated at your site and external mail originated out on the Internet and passing through your system back out to the Internet. The former class of mail you want to permit; the latter class you want to block. This differentiation is achieved using the `switchchannel` keyword on your inbound SMTP channel, usually the `tcp_local` channel.

The `switchchannel` keyword works by causing the SMTP server to look at the actual IP address associated with the incoming SMTP connection. Messaging Server uses that IP address, in conjunction with your rewrite rules, to differentiate between an SMTP connection originated within your domain and a connection from outside of your domain. This information can then be used to segregate the message traffic between internal and external traffic.

The following steps describe how to change your MTA configuration so that the server can differentiate between your internal and external message traffic.

1. In your configuration file, locate your local store channel. Immediately before the local channel, add a `defaults` channel with the `noswitchchannel` keyword, as shown in the following example:

   ```
   ! final rewrite rules
   defaults noswitchchannel
   ! Local store
   ims-ms ...
   ```

   If you already have a `defaults` channel at this point in your configuration file, then simply add the `noswitchchannel` keyword to it.

2. Modify your incoming TCP/IP channel to specify the `switchchannel` and `remotehost` keywords; for example:

   ```
   tcp_local smtp single_sys mx switchchannel remotehost
   TCP-DAEMON
   ```

3. After your incoming TCP/IP channel definition, add a similar new channel but with a different name; for example:

   ```
   tcp_internal smtp single_sys mx allowswitchchannel routelocal
   TCP-INTERNAL
   ```

   The `routelocal` channel keyword is used to cause short-circuited rewriting of source routed addresses through this channel, thereby blocking possible attempts to relay by means of looping through internal SMTP hosts via explicitly source routed addresses.

4. Add rewrite rules to route hosts and IP addresses in your domain to the `tcp_internal` channel. For instance, if your domain name is sesta.com and your domain's IP numbers are in the [`a.b.subnet`] range, then add to the top of your configuration file the following rewrite rules

   ```
   .sesta.com      $U%$H$D@TCP-INTERNAL
   [a.b.]           $U%[a.b.$L]@TCP-INTERNAL$E$R
   ```

With the above configuration changes, SMTP mail generated within your domain will come in via the `tcp_internal` channel. All other SMTP mail will come in via the `tcp_local` channel. You can therefore distinguish between internal and external mail based upon which channel it comes in on.

How does the above work? The key is the `switchchannel` keyword. In Step 2, the keyword is applied to the `tcp_local` channel. When a message comes in your SMTP server, that keyword causes the server to look at the source IP address associated with the incoming connection. The server attempts a reverse-pointing

envelope rewrite of the literal IP address of the incoming connection, looking for an associated channel. If that rewrite matches a local host of yours, then the rewrite rules added in Step 4 cause the address to rewrite to the `tcp_internal` channel added in Step 3.

Since the `tcp_internal` channel is marked with the `allowswitchchannel` keyword, the message is switched to the `tcp_internal` channel and comes in on that channel. If the message comes in from an external source, the IP address will not correspond to an internal source. In that case the reverse-pointing envelope rewrite will either rewrite to the `tcp_local` channel or to some other channel. However, it will not rewrite to the `tcp_internal` channel and since all other channels were marked `noswitchchannel` in Step 1, the message will not switch to another channel and will remain with the `tcp_local` channel.

NOTE Note that any mapping table or conversion file entries which use the string “tcp_local” may need to be changed to either “tcp_*” or “tcp_internal” depending upon the usage.

Differentiate Authenticated Users' Mail

Your site might have “local” client users who are not part of your physical network. When these users submit mail, the message submissions come in from an external IP address—for instance, arbitrary Internet Service Providers. If your users use mail clients that can perform SASL authentication, then their authenticated connections can be distinguished from arbitrary other external connections. The authenticated submissions you can then permit, while denying non-authenticated relay submission attempts. Differentiating between authenticated and non-authenticated connections is achieved using the `saslswitchchannel` keyword on your inbound SMTP channel, usually the `tcp_local` channel.

The `saslswitchchannel` keyword takes an argument specifying the channel to switch to; if an SMTP sender succeeds in authenticating, then their submitted messages are considered to come in the specified switched to channel.

To add distinguishing authenticated submissions:

1. In your configuration file, add a new TCP / IP channel definition with a distinct name; for example:

   ```
   tcp_auth smtp single_sys mx mustsaslserver noswitchchannel
   TCP-INTERNAL
   ```

This channel should not allow regular channel switching (that is, it should have noswitchchannel on it either explicitly or implied by a prior defaults line). This channel should have mustsaslserver on it.

2. Modify your tcp_local channel by adding maysaslserver and saslswitchchannel tcp_auth, as shown in the following example:

```
tcp_local smtp mx single_sys maysaslserver saslswitchchannel
tcp_auth \
switchchannel
|TCP-DAEMON
```

With this configuration, SMTP mail sent by users who can authenticate with a local password will now come in the tcp_auth channel. Unauthenticated SMTP mail sent from internal hosts will still come in tcp_internal. All other SMTP mail will come in tcp_local.

Prevent Mail Relay

Now to the point of this example: preventing unauthorized people from relaying SMTP mail through your system. First, keep in mind that you want to allow local users to relay SMTP mail. For instance, POP and IMAP users rely upon using Messaging Server to send their mail. Note that local users may either be physically local, in which case their messages come in from an internal IP address, or may be physically remote but able to authenticate themselves as local users.

You want to prevent random people out on the Internet from using your server as a relay. With the configuration described in the following sections, you can differentiate between these classes of users and block the correct class. Specifically, you want to block mail from coming in your tcp_local channel and going back out that same channel. To that end, an ORIG_SEND_ACCESS mapping table is used.

An ORIG_SEND_ACCESS mapping table may be used to block traffic based upon the source and destination channel. In this case, traffic from and back to the tcp_local channel is to be blocked. This is realized with the following ORIG_SEND_ACCESS mapping table:

```
ORIG_SEND_ACCESS

   tcp_local|*|tcp_local|*        $NRelaying$ not$ permitted
```

In this example, the entry states that messages cannot come in the tcp_local channel and go right back out it. That is, this entry disallows external mail from coming in your SMTP server and being relayed right back out to the Internet.

An ORIG_SEND_ACCESS mapping table is used rather than a SEND_ACCESS mapping table so that the blocking will not apply to addresses that originally match the ims-ms channel (but which may expand via an alias or mailing list definition back to an external address). With a SEND_ACCESS mapping table one would have to go to extra lengths to allow outsiders to send to mailing lists that expand back out to external users, or to send to users who forward their messages back out to external addresses.

Allowing localhost Submissions to the SMTP Port

This section discusses a further refinement of the SMTP relay blocking approach described above.

Some sites might want to disallow submissions to the SMTP port from clients running on the system itself; for instance, if no legitimate applications are expected to attempt this, then blocking such submissions closes one avenue by which users can spoof (forge) email.

Other sites, however, might have legitimate applications that perform message submission by making a TCP/IP connection to the SMTP port of their own system. For instance, some third party mailing list expansion applications (though not Messaging Server's own mailing list expansion) operate in this way.

Further, to simplify their configuration such applications may connect using a loopback name or address, such as LOCALHOST or [127.0.0.1], rather than using the system's domain name. Depending on the underlying TCP/IP package, this may result in the incoming connection also appearing to come from LOCALHOST or [127.0.0.1], rather than coming from the system's specific domain name or IP address.

Using merely the internal versus external differentiation as described previously would result in treating SMTP submissions from clients on the host system itself as coming in the tcp_local channel. Thus if tcp_local to tcp_local message traffic is prevented, then any users or applications attempting to submit messages in such a way would not be allowed to submit messages to external addressees.

If you wish to treat such submissions as “internal” submissions, for instance, in order to allow third party mailing list applications to operate even if SMTP relay blocking has been implemented, then an additional configuration step should be performed.

At the top of the MTA configuration file, add rewrite rules for the local host name (or LOCALHOST or [127.0.0.1], depending on from where the underlying TCP/IP package sees the connection as having originated) of the following form:

```
localhostname           $E$R$U%localhostname@TCP-INTERNAL
[localhostipnumber]     $E$R$U%localhostname@TCP-INTERNAL
LOCALHOST               $E$R$U%localhostname@TCP-INTERNAL
[127.0.0.1]             $E$R$U%localhostname@TCP-INTERNAL
```

where *localhostname* is the official host name on the ims-ms channel.

The $E and $R control sequences on these rewrite rules, which limit their effect to envelope From: address rewriting, mean that normal rewriting will still apply to addresses addressed to the local system. Yet switchchannel rewriting will use these rules also, and hence will see message submissions from the system to its own SMTP port as internal submissions.

Using DNS Lookups Including RBL Checking for SMTP Relay Blocking

In the iPlanet Messaging Server, there are a number of different ways to ensure that all mail accepted for delivery or forwarding comes from an address with a valid DNS name. The simplest way is to put the mailfromdnsverify channel keyword on the tcp_local channel.

iPlanet Messaging Server also provides the dns_verify program which allows you to ensure that all mail accepted for delivery or forwarding comes from an address with a valid DNS name using the following rule in ORIG_MAIL_ACCESS:

```
ORIG_MAIL_ACCESS

  TCP|*|*|*|*|SMTP|MAIL|*|*@*|*|*    \
     $[<server-root>/bin/msg/imta/lib/dns_verify,     \
    dns_verify,$6|$$y|$$NInvalid$ host:$ $$6$ -$ %e]
```

The line breaks in the above example are syntactically significant in such mapping entries. The backslash character is a way of legally continuing on to the next line.

The dns_verify image can also be used to check incoming connections against things like the RBL (Realtime Blackhole List), MAPS (Mail Abuse Prevention System, DUL (Dial-up User List), or ORBS (Open Relay Behavior-modification System) lists as another attempt to protect against UBE. As with the new mailfromdnsverify keyword, there's also a separate "simpler to configure" approach one can use for such checks rather than doing the dns_verify callout. The simpler approach is to use the DNS_VERIFY_DOMAIN option in the dispatcher.cnf file. For example, in the [SERVICE=SMTP] section, set instances of the option to the various lists you want to check against:

```
[SERVICE=SMTP]
PORT=25
! ...rest of normal options...
DNS_VERIFY_DOMAIN=rbl.maps.vix.com
DNS_VERIFY_DOMAIN=dul.maps.vix.com
!...etc...
```

The disadvantage of this simpler approach is that it does the checks for all normal incoming SMTP messages including those from internal users. This is less efficient and potentially problematic if your Internet connectivity goes down. An alternative is to call out to dns_verify from a PORT_ACCESS mapping table or ORIG_MAIL_ACCESS mapping table. In the PORT_ACCESS mapping table, you can have an initial entry or entries that don't check for local internal IP addresses or message submitters and a later entry that does the desired check for everyone else. Or, in an ORIG_MAIL_ACCESS mapping table, if you only apply the check on messages coming in the tcp_local channel then you're skipping it for messages coming from your internal systems / clients. Examples using the entry points to dns_verify are shown below.

```
PORT_ACCESS

! Allow internal connections in unconditionally
  *|*|*|*|*  $C$|INTERNAL_IP;$3|$Y$E
! Check other connections against RBL list
  TCP|*|25|*|*    \
      $C$[<server-root>/bin/msg/imta/lib/dns_verify,  \
dns_verify_domain_port,$1,rbl.maps.vix.com]EXTERNAL$E


ORIG_MAIL_ACCESS

  TCP|*|25|*|*|SMTP|*|tcp_local|*@*|*|*    \
      $C$[<server-root>/bin/msg/imta/lib/dns_verif,    \
      dns_verify_domain,$1,rbl.maps.vix.com]$E
```

Handling Large Numbers of Access Entries

Sites that use very large numbers of entries in mapping tables should consider organizing their mapping tables to have a few general wildcarded entries that call out to the general database for the specific lookups. It is much more efficient to have a few mapping table entries calling out to the general database for specific lookups than to have huge numbers of entries directly in the mapping table.

One case in particular is that some sites like to have per user controls on who can send and receive Internet email. Such controls are conveniently implemented using an access mapping table such as `ORIG_SEND_ACCESS`. For such uses, efficiency and performance can be greatly improved by storing the bulk of the specific information (e.g., specific addresses) in the general database with mapping table entries structured to call out appropriately to the general database.

For example, consider the mapping table shown in Figure 9-5.

Figure 9-5 ORIG_SEND_ACCESS Mapping Table

```
ORIG_SEND_ACCESS

  ! Users allowed to send to Internet
  !
  *|adam@siroe.com|*|tcp_local    $Y
  *|betty@siroe.com|*|tcp_local   $Y
  ! ...etc...
  !
  ! Users not allowed to send to Internet
  !
  *|norman@siroe.com|*|tcp_local  $NInternet$ access$ not$
      permitted
  *|opal@siroe.com|*|tcp_local    $NInternet$ access$ not$
      permitted
  ! ...etc...
  !
  ! Users allowed to receive from the Internet
  !
  tcp_*|*|*|adam@siroe.com        $Y
  tcp_*|*|*|betty@siroe.com       $Y
  ! ...etc...
  !
  ! Users not allowed to receive from the Internet
  !
  tcp_*|*|*|norman@siroe.com      $NInternet$ e-mail$ not$
      accepted
  tcp_*|*|*|opal@siroe.com        $NInternet$ e-mail$ not$
      accepted
  ! ...etc...
```

Rather than using such a mapping table with each user individually entered into the table, a more efficient setup (much more efficient if hundreds or thousands of user entries are involved) is shown in Figure 9-6, which shows sample general database entries and a sample `ORIG_SEND_ACCESS` mapping table.

Figure 9-6 Sample Database Entries and Mapping Table

DATABASE ENTRIES

```
SEND|adam@domain.com    $Y
SEND|betty@domain.com   $Y
! ...etc...
SEND|norman@domain.com  $NInternet$ access$ not$ permitted
SEND|opal@domain.com    $NInternet$ access$ not$ permitted
! ...etc...
RECV|adam@domain.com    $Y
RECV|betty@domain.com   $Y
! ...etc...
RECV|norman@domain.com  $NInternet$ e-mail$ not$ accepted
RECV|opal@domain.com    $NInternet$ e-mail$ not$ accepted
```

MAPPING TABLE

```
ORIG_SEND_ACCESS

  ! Check if may send to Internet
  !
  *|*|*|tcp_local       $C${SEND|$1}$E
  !
  ! Check if may receive from Internet
  !
  tcp_*|*|*|*           $C${RECV|$3}$E
```

In this example, the use of the arbitrary strings SEND| and RECV| in the general database left-hand sides (and hence in the general database probes generated by the mapping table) provides a way to distinguish between the two sorts of probes being made. The wrapping of the general database probes with the $C and $E flags, as shown, is typical of mapping table callouts to the general database.

The above example showed a case of simple mapping table probes getting checked against general database entries. Mapping tables with much more complex probes can also benefit from use of the general database.

Mapping Table Flags

Table 9-3 shows the access mapping flags relevant for the SEND_ACCESS, ORIG_SEND_ACCESS, MAIL_ACCESS, ORIG_MAIL_ACCESS, and FROM_ACCESS mapping tables. Note that the PORT_ACCESS mapping table, supports a somewhat different set of flags (see Table 9-2).

Table 9-3 Access Mapping Flags

Flag	Description
$B	Redirect the message to the bitbucket.
$H	Hold the message as a .HELD file.
$Y	Allow access.
Flags with Arguments, in Argument Reading Order+	
$Jaddress	Replace original envelope From: address with specified address.*
$Kaddress	Replace original Sender: address with specified address.* ++
$Iuser | identifier	Check specified user for groupid.
$<string	Send string to syslog (UNIX) or to the event log (NT) if probe matches.+++
$>string	Send string to syslog (UNIX) or to the event log (NT) if access is rejected. +++
$Ddelay	Delay response for an interval of delay hundredths of seconds; a positive value causes the delay to be imposed on each command in the transaction; a negative value causes the delay to be imposed only on the address handover (SMTP MAIL FROM: command for the FROM_ACCESS table; SMTP RCPT TO: command for the other tables).
$Ttag	Prefix with tag.
$Aheader	Add the header line header to the message.
$Xerror-code	Issue the specified error-code extended SMTP error code if rejecting the message.
$Nstring	Reject access with the optional error text string.
$Fstring	Synonym for $N string; that is, reject access with the optional error text string.

Table 9-3 Access Mapping Flags

Flag	Description

* Available for FROM_ACCESS table only.

+ To use multiple flags with arguments, separate the arguments with the vertical bar character, |, placing the arguments in the order listed in this table.

++ For the $K flag to take effect in the FROM_ACCESS mapping table, the source channel must include the `authrewrite` keyword.

+++ It is a good idea to use the $D flag when dealing with problem senders, to prevent a denial of service attack. In particular, it is a good idea to use $D in any $> entry or $< entry rejecting access.

PART 2. MAILBOX FILTERS

This part contains the following sections:

- Introduction
- Creating Per-User Filters
- Creating Channel-Level Filters
- Creating MTA-Wide Filters
- Debugging User Filters

Introduction

A filter consists of one or more conditional actions to apply to a mail message. Messaging Server filters are stored on the server and evaluated by the server. Hence, they are sometimes called server-side rules (SSR). Messaging Server filters are based on the Sieve filtering language, Draft 9 of the Sieve Internet Draft.

As an administrator, you can create channel-level filters and MTA-wide filters to prevent delivery of unwanted mail. You can also create filter templates and make them available to end users via the Delegated Administrator for Messaging interface. End users use the templates to build personal mailbox filters to prevent delivery of unwanted mail messages to their mailboxes.

The server applies filters in the following priority:

1. Per-user filters

   If a personal mailbox filter explicitly accepts or rejects a message, then filter processing for that message finishes. But if the recipient user had no mailbox filter—or if the user's mailbox filter did not explicitly apply to the message in question—Messaging Server next applies the channel-level filter.

2. Channel-level filter

   If the channel-level filter explicitly accepts or rejects a message, then filter processing for that message finishes. Otherwise, Messaging Server next applies the MTA-wide filter, if there is one.

3. MTA-wide filter

By default, each user has no mailbox filter. When a user uses the Delegated Administrator interface to create one or more filters, then their filters are stored in the Directory and retrieved by the MTA during the directory synchronization process.

Creating Per-User Filters

Per-user filters apply to messages destined for a particular user’s mailbox. As an administrator, you can create filter templates and make them available to end users via the Delegated Administrator for Messaging interface. End users use the templates to build personal server filters to manipulate the delivery of mail messages to their mailboxes; that is, to reject unwanted messages, redirect mail, filter messages into mailbox folders, and so on.

A filter template generalizes a Sieve script by replacing “hard-coded” elements of the Sieve script with prompts and input fields. A Java servlet is used to parse the sieve templates and generate the UI pages in the browser. When an end user supplies values in the input fields, the servlet takes those values and saves them in a sieve script in the user’s directory profile entry. The prompts and input fields are presented to the end user through the Delegated Administrator interface.

A set of sample templates is provided and installed with Delegated Administrator. The template files are located in the following directory:

`nda-path/nda/nda/default/`*`lang`*`/templates/enduser/ssr/*.txt`

You can modify these filter templates or create new ones using the Sieve language. If you create new filter templates, you must save the filter template in a text file in the ssr directory described above. You must ensure that the file is word readable and you must add an LDAP entry for the filter template, as shown in the following example:

```
dn: cn=Subject Discard,cn=ssrconf,cn=en,
              cn=domainConfiguration,ou=config,o=isp
objectclass: top
objectclass: nsValueItem
cn: Subject Discard
nsvaluetype: nsValueCIS
nsvaluecis: ../templates/enduser/ssr/subject-discard.txt
```

Figure 9-7 shows a sample template.

Figure 9-7 Sample Sieve Template

```
#RULE: $Template="File To Folder"
require "fileinto";
if header :contains # Q1
    # Q2
{
    fileinto # Q3
    ;
}

#PRE: "This rule files messages into a folder."
#PRE: "Choose the header line to search on"
#PRE: "And specify the phrase you wish to search for"
#Q1: header "If the header line"
#Q2: value "Contains the phrase"
#Q3: folder "Then file into the folder"
```

In the above example, Q1, Q2, and Q3 serve as place holders for input values, where the UI can locate the position to substitute the value. Each token will map to a question and a data type for the input value.

The data type and associated question are defined in the comment lines for each token. They are defined as the form *token* : *data-type-variable*, and followed by a quoted string which contains the actual question. In the above example, `header value`, and `folder` are all data types that will either present a drop-down list, edit box, or otherwise. These data type variables tell the UI what type of information to get from the user.

When the template is parsed a dialog is generated and presented to the end user as shown in Figure 9-8. In the example, brackets indicate a drop-down list.

Figure 9-8 Sample Template Output

```
+-------------------------------------------------------------+
| Template: File To Folder Name: ____________________         |
+-------------------------------------------------------------+
|           This rule files messages to a folder              |
|           Choose the header line to search on               |
|       And specify the phrase you wish to search for         |
|                                                             |
|   If the header line: [From       ]                         |
|   Contains the phrase: ___________________________          |
|   Then file into the folder: ____________________           |
+-------------------------------------------------------------+
```

After the user enters the data, the rule is stored in the user's `mailSieveRuleSource` attribute.

The syntax of the template has the following restrictions:

- The `#RULE` line needs to appear before any other line, with `$Template` specified.
- Any comment line starting with `#PRE` is displayed before the input fields in the GUI page.

  `#PRE` statements need to be enclosed in double quote strings.
- Any comment line starting with `#POST` is displayed at the end of the GUI page.

  `#POST` statements mus be enclosed in double quote strings.
- Other comment lines are not be displayed in the GUI page.

- Tokens are ASCII strings and are case-insensitive; tokens cannot contain white spaces.
- Data-type variables follow the token string in the comment lines; these are also case insensitive.
- The actual question is defined in a comment line right after a data-type variable, and is enclosed by double quotes.

The following data-type variables are supported in the Sieve templates:

- `header` - When represented in GUI, a list box is used, and the following values are available: `Subject`, `To`, `From`.

  When the sieve rule is saved to the user entry, the `Subject` value is expanded to Subject, Comments, Keywords; the `From` value is expanded to From, Sender, Resent-from, Resent-sender, Return-path; and the `To` value is expanded to To, Cc, Bcc, Resent-to, Resent-cc, Reset-bcc.
- `value` - A text field is used to represent this.
- `address` - A text field is used to represent this. Addresses' syntax will be checked against RFC 822 mail addresses format.
- `folder` - A text field is used to represent this.
- `size` - Users can choose from Kilobyte, Megabyte, or specify any number.
- `message` - A text area is used to represent this.

Creating Channel-Level Filters

Channel-level filters apply to each message enqueued to a channel. A typical use for this type of filter is to block messages going through a specific channel.

To create a channel-level filter:

1. Write the filter using Sieve.
2. Store the filter in a file in the following directory:

   `msg-`*instance*`/imta/config/`*file*`.filter`

   The file must be world readable and owned by the MTA's uid.
3. Include the following in the channel configuration:

   `destinationfilter file:IMTA_TABLE:`*file*`.filter`

4. Recompile the configuration and restart the Dispatcher.

   Note that changes to the filter file do not require a recompile or restart of the Dispatcher.

The `destinationfilter` channel keyword enables message filtering on messages enqueued *to* the channel to which it is applied. The `sourcefilter` channel keyword enables message filtering on messages enqueued *by* (from) the channel to which it is applied. These keywords each have one required parameter which specifies the path to the corresponding channel filter file associated with the channel.

The syntax for the `destinationfilter` channel keyword is:

`destinationfilter` *URL-pattern*

The syntax for the `sourcefilter` channel keyword is:

`sourcefilter` *URL-pattern*

where *URL-pattern* is a URL specifying the path to the filter file for the channel in question. In the following example, *channel-name* is the name of the channel.

`destinationfilter file:///imta/config/`*channel-name*`.filter`

The `filter` channel keyword enables message filtering on the channels to which it is applied. The keyword has one required parameter which specifies the path to the filter files associated with each envelope recipient who receives mail via the channel.

The syntax for the `filter` channel keyword is

`filter` *URL-pattern*

URL-pattern is a URL that, after processing special substitution sequences, yields the path to the filter file for a given recipient address. *URL-pattern* can contain special substitution sequences that, when encountered, are replaced with strings derived from the recipient address, `local-part@host.domain` in question. These substitution sequences are shown in Table 9-4.

The `fileinto` keyword specifies how to alter an address when a mailbox filter `fileinto` operator is applied. The following example specifies that the folder name should be inserted as a subaddress into the original address, replacing any originally present subaddress:

```
fileinto $U+$S@$D
```

Table 9-4 Substitution Sequences

Sequence	Substitution String
$$	Substitute in the $ character
$A, $a	Substitute in the address, local-part@ host.domain
$D, $d	Substitute in host.domain
$H, $h	Substitute in host
$L, $l	Substitute in local-part
$U, $u	Substitute in local-part less any underscore or tilde prefixes and less any subaddress postfix
$~	Substitute in the file path for the home directory associated with the local part of the address

Creating MTA-Wide Filters

MTA-wide filters apply to all messages enqueued to the MTA. A typical use for this type of filter is to block unsolicited bulk email or other unwanted messages regardless of the messages' destinations. To create an MTA-wide filter:

1. Write the filter using Sieve
2. Store the filter in the following file:

   `msg-`*instance*`/imta/config/imta.filter`

   This filter file must be world readable. It is used automatically, if it exists.
3. Recompile the configuration and restart the Dispatcher

When using a compiled configuration, the MTA-wide filter file is incorporated into the compiled configuration.

Routing Discarded Messages out The FILTER_DISCARD Channel

By default, messages discarded via a mailbox filter are immediately discarded (deleted) from the system. However, when users are first setting up mailbox filters (and perhaps making mistakes), or for debugging purposes, it can be useful to have the deletion operation delayed for a period.

To have mailbox filter discarded messages temporarily retained on the system for later deletion, first add a `filter_discard` channel to your MTA configuration with the `notices` channel keyword specifying the length of time (normally number of days) to retain the messages before deleting them, as shown in the following example:

```
filter_discard notices 7
FILTER-DISCARD
```

Then set the option `FILTER_DISCARD=2` in the MTA option file. Messages in the `filter_discard` queue area should be considered to be in an extension of users' personal wastebasket folders. As such, note that warning messages are never sent for messages in the `filter_discard` queue area, nor are such messages returned to their senders when a bounce or return is requested. Rather, the only action taken for such messages is to eventually silently delete them, either when the final notices value expires, or if a manual bounce is requested using a utility such as `imsimta return`.

Debugging User Filters

The following information will help you if you are having problems with the user filters on your system.

The dirsync process updates the MTA's SSR database with information about the users' filters. Short filters are stored in the database. For long filters, the database stores an LDAP dn. Note that the MTA doesn't see changes to a user's filters until the dirsync process has updated the database.

To facilitate debugging problems with filters, follow these steps:

- In the file `imta.cnf`, make sure that the `ims-ms` channel is marked as follows:

  ```
  filter ssrd:$a fileinto $u+$s@$d
  ```

- Ensure that the `dirsync` process knows to synchronize filter information by using the `configutil` command as follows:

```
configutil -l -o service.imta.ssrenabled -v true
OK SET
configutil | fgrep ssr
service.imta.ssrenabled = true
```

- To test filters, use the `imsimta test` command as follow:

  ```
  imsimta test -rewrite -debug -filter user@domain
  ```

  In the output, look for the following:

  ```
  mmc_open_url called to open ssrd:user@ims-ms
     URL with quotes stripped: ssrd:user@ims-ms
  Determined to be an SSRD URL.
     Identifier: user@ims-ms-daemon
  Filter successfully obtained.
  ```

- If there's a syntax problem with the filter, look for the following:

  ```
  Error parsing filter expression:...
  ```

- If the filter is good, the `test` command displays the filter at the end of the output.

- If there are problems with the filter, the `test` command displays the following at the end of the output:

  ```
  Address list error -- 4.7.1 Filter syntax error: user@siroe.com
  ```

  Also, the `SMTP RCPT TO` command will return a temporary error response code, such as:

  ```
  RCPT TO:<user@siroe.com>
  452 4.7.1 Filter syntax error
  ```

- If you know the final rewritten form of the user's address, you can use the `imsimta test -url` command to see what the MTA is using as filters for the user:

  ```
  imsimta test -url ssrd:user@ims-ms-daemon
  ```

  You can use the `imsimta test -rewrite` command to find the final rewritten form of the user's address.

Chapter 10

Managing the Message Store

This chapter describes the message store and the message store administration interface. This chapter contains the following sections:

- Overview
- Message Store Directory Layout
- How the Store Erases Message
- Specifying Administrator Access to the Store
- About Message Store Quotas
- Configuring Message Store Quotas
- Specifying Aging Policies
- Configuring Message Store Partitions
- Performing Maintenance and Recovery Procedures
- Backing Up and Restoring the Message Store

Overview

The message store contains the user mailboxes for a particular Messaging Server instance. The size of the message store increases as the number of mailboxes, folders, and log files increase. You can control the size of the store by specifying limits on the size of mailboxes (disk quotas), by specifying limits on the total number of messages allowed, and by setting aging policies for messages in the store.

As you add more users to your system, your disk storage requirements increase. Depending on the number of users your server supports, the message store might require one physical disk or multiple physical disks. If you have a very large user base, you might have multiple Messaging Server instances, each responsible for a particular message store. Likewise, if you are supporting multiple hosted domains, you might want to dedicate a server instance to a single, large domain. With this configuration, you can designate a store administrator for a particular domain.

To manage the message store, iPlanet Messaging Server provides a set of command-line utilities in addition to the iPlanet Console interface. Table 10-1 describes these command-line utilities. For information about using these utilities, see “Performing Maintenance and Recovery Procedures” on page 257 and the *Messaging Server Reference Manual*.

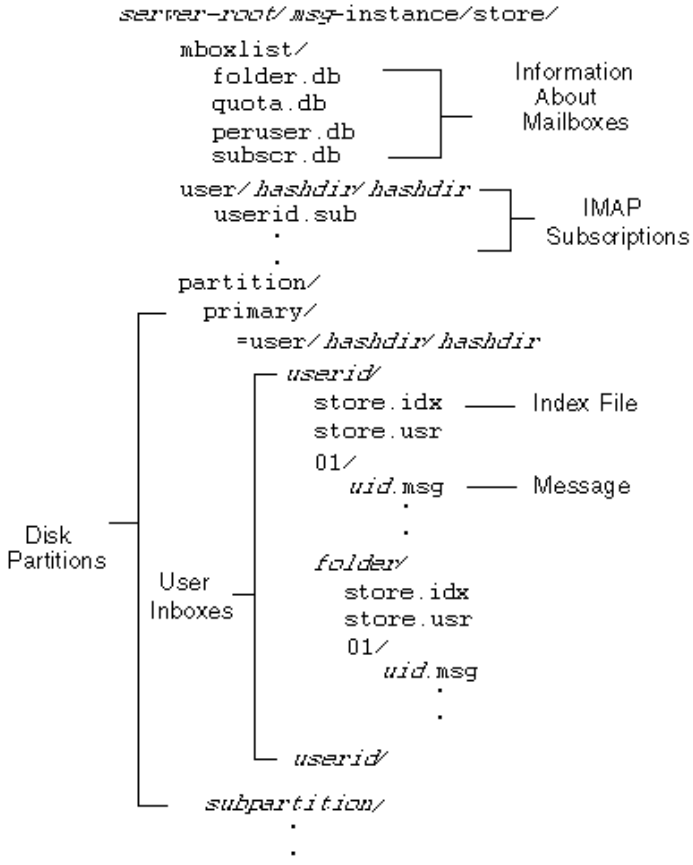
Table 10-1 Message Store Command-line Utilities

Utility	Description
`configutil`	Sets and modifies configuration parameters for the store.
`hashdir`	Identifies the directory that contains the message store for a particular user.
`mboxutil`	Lists, creates, deletes, renames, or moves mailboxes; reports quota usage.
`MoveUser`	Moves a user’s account from one messaging server to another.
`readership`	Collects readership information on shared IMAP folders.
`reconstruct`	Reconstructs mailboxes that have been damaged or corrupted.
`stored`	Performs background and daily tasks, expunges, and erases messages stored on disk.
`imsbackup`	Performs backups of the messages stored on disk.
`imsrestore`	Restores messages that have been backed up.

Message Store Directory Layout

Figure 10-1 shows the message store directory layout for a server instance. The message store is designed to provide fast access to mailbox contents. The store directories are described in Table 10-2.

Figure 10-1 Message Store Directory Layout



For example, a sample directory path might be:

server-root/msg-*instance*/store/partition/primary/=user/53/53/=mack1

Table 10-2 Message Store Directory Description

Location	Content/Description
server-root/msg-*instance*/store/	Top-level directory of the message store. Contains the mboxlist, user, and partition subdirectories.
.../store/mboxlist/	Contains a database (Berkley DB) that stores information about the mailboxes on the server and stores quota information about the mailboxes. The file folder.db contains information about mailboxes, including the name of the partition where the mailbox is stored, the ACL, and a copy of some of the information in store.idx. There is one entry in folder.db per mailbox The file quota.db contains information about quotas and quota usage. There is one entry in quota.db2 per user. The file peruser.db contains information about per-user flags. The flags indicate whether a particular user has seen or deleted a message. The file subscr.db contains information about user subscriptions.
.../store/user/	Contains information about the IMAP folders to which each user subscribes. Information for each user is stored in a file called *userid*.sub. These files are stored in a hash structure for fast searching. To find the directory that contains a particular user's files, use the hashdir utility.
.../store/partition/	Contains the default primary partition. You can also place any other subpartitions you define in this directory.
/*subpartition*/=user/	Contains all the user mailboxes in the subdirectory of the partition. The mailboxes are stored in a hash structure for fast searching. To find the directory that contains a particular user's mailbox, use the hashdir utility.

Table 10-2 Message Store Directory Description

Location	Content/Description
/=user/*hashdir*/*hashdir*/*userid*/	The top-level mail folder for the user whose ID is *userid*. For the default domain, *userid* is *uid*. For hosted domains, *userid* is *uid@domain*. Messages are delivered to this mail folder.
/*userid*/*folder*	A user-defined folder.
/*userid*/store.idx	An index that provides the following information about mail stored in the /*userid*/ directory: number of messages, disk quota used by this mailbox, the time the mailbox was last appended, message flags, variable-length information for each message including the headers and the MIME structure, and the size of each message. The index also includes a backup copy of mboxlist information for each user and a backup copy of quota information for each user.
/*userid*/store.usr	Contains a list of users who have accessed the folder. For each user listed, contains information about the last time the user accessed the folder, the list of messages the user has seen, and the list of messages the user has deleted.
/*userid*/store.exp	(Not shown in Figure 10-2.) Contains a list of message files that have been expunged, but not removed from disk. This file appears only if there are expunged messages.
/*userid*/store.sub	(Not shown in Figure 10-2.) Contains information about user subscriptions.
/*userid*/*nn*/	A hash directory that contains messages in the format *msgid*.msg; *nn* can be a number from 00 to 99. For example, messages 1 through 99 are stored in the 00 directory; messages 100 through 199 are stored in the 01 directory; messages 9990 through 9999 are stored in the 99 directory; messages 10000 through 10099 are in the 00 directory, and so on.

How the Store Erases Message

Messages are erased from the store in three stages:

1. **Delete**. A client marks the message to be deleted. At this point, the client can restore the message by removing the “deleted” marking.
2. **Expunge**. A client, or the aging policies you have specified, expunges messages that have been marked deleted from the mailbox. Once messages are expunged, the client can no longer restore them, but they are still stored on disk. (A second client with an existing connection to the same mailbox may still be able to fetch the messages.)
3. **Cleanup**. The `stored` utility erases from the disk any messages that have been expunged for at least one hour.

Specifying Administrator Access to the Store

Message store administrators can view and monitor user mailboxes and specify access control for the message store. Store administrators have proxy authentication privileges to any service (POP, IMAP, HTTP, or SMTP), which means they can authenticate to any service using the privileges of any user. These privileges allow store administrators to run certain utilities for managing the store. For example, using `MoveUser`, store administrators can move user accounts and mailboxes from one system to another.

This section discusses how to grant store privileges to the message store for your Messaging Server installation.

NOTE Other users might also have administrator privileges to the store. For example, if your site uses the Delegated Administration (DA) product, top-level DA administrators by default have store privileges for all messaging servers in the mail system. DA domain administrators by default have store privileges for their domain. For more information about the DA administrators, see the *Messaging Server Provisioning Guide* and the DA documentation.

You can perform tasks as described in the following subsections:

- Adding an Administrator
- Modifying an Administrator Entry
- Deleting an Administrator Entry

You can specify administrator access to the store by using the `configutil` command or by using Console.

If you want to use Console:

1. From Console, open the Messaging Server you want to configure.
2. Click the Configuration tab and select Message Store in the left pane.
3. Click the Administrator tab in the right pane.

Adding an Administrator

Console. To add an administrator entry at the Console:

1. Click the Administrator tab.

   The tab contains a list of existing administrator IDs.
2. Click the Add button beside the Administrator UID window.
3. In the Administrator UID field, type the user ID of the administrator you want to add.

   The user ID you type must be known to the iPlanet Directory Server.
4. Click OK to add the administrator ID to the list displayed in the Administrator tab.
5. Click Save in the Administrator tab to save the newly modified Administrator list.

Command Line. To add an administrator entry at the command line:

`configutil -o store.admins -v "`*`adminlist`*`"`

where *adminlist* is a comma-separated list of administrator IDs. If you specify more than one administrator, you must enclose the list in quotes.

Modifying an Administrator Entry

Console. To modify an existing entry in the message store Administrator UID list at the Console:

1. Click the Administrator tab.
2. Click the Edit button beside the Administrator UID window.
3. Enter your changes to the Administrator UID field.
4. Click OK to submit your changes and dismiss the Edit Administrator window.
5. Click Save in the Administrator tab to submit and preserve the modified Administrator list.

Command Line. To modify an existing entry in the message store Administrator UID list at the command line:

```
configutil -o store.admins -v "adminlist"
```

Deleting an Administrator Entry

Console. To delete an entry from the message store Administrator UID list by using the Console:

1. Click the Administrator tab.
2. Select an item in the Administrator UID list.
3. Click Delete to delete the item.
4. Click Save to submit and preserve your changes to the Administrator list.

Command Line. To delete store administrators at the command line, you can edit the administrator list as follows:

```
configutil -o store.admins -v "adminlist"
```

About Message Store Quotas

This section contains information about the following:

- User Quotas
- Domain Quotas and Family Group Quotas

- Exceptions for Telephony Application Servers

User Quotas

You can limit the size of the message store by specifying limits on the size of user mailboxes. You can specify the following types of quotas.

- Disk quotas allow you to limit the amount of disk space allotted to each user. Disk quotas apply to the total size of all the user's messages, regardless of how many mail folders the user has or to the total number of user messages. If disk space is limited, you might want to set user disk quotas.
- Message quotas allow you to limit the number of messages stored in a user's mailbox.

Quota information is stored as LDAP attributes and configuration variables. If quota enforcement is enabled, Messaging Server checks the quota cache and configuration file to ensure quotas have not been exceeded before inserting messages into the message store. If quota notification is enabled, users are sent an error message when they have reached their disk quota. You can also enable the server to send a warning message when users are nearing their quota limit.

You can set default quotas for all users or set quotas for individual users. To determine if a user is over quota, Messaging Server first checks to see if a quota has been set for the individual user. If no quota has been set, Messaging Server then looks at the default quota set for all users.

If the total size or the total number of all the user's messages exceeds the specified limits set, messages destined for the user remain in the message queue until one of the following occurs: (1) The size or number of all the user's messages no longer exceeds the limit, at which time the server delivers the message to the user. (2) The undelivered message has been in the queue longer than the specified grace period and the user is still over quota, at which time the server bounces the message.

NOTE The server does not consider the size of the message when it is attempting to deliver to an account that is still under quota. If the message causes the user to go over quota, the message is still delivered, but the next message will be held in the queue.

Disk space becomes available when a user deletes and expunges messages or when the server deletes messages according to the maintenance policies you have established (aging policies, for example).

Domain Quotas and Family Group Quotas

You can also set quotas for a particular domain and for family groups within a domain. These quotas are not enforced, but they are useful for reporting purposes. For more information about domain and family group quotas, see the *Delegated Administrator User's Guide*.

Exceptions for Telephony Application Servers

To support unified messaging requirements, Messaging Server provides the ability to override quota limitations imposed by the message store, thus guaranteeing the delivery of messages that have been accepted by certain agents, namely telephony application servers (TAS). Messages accepted by a TAS can be routed through a special MTA channel that will ensure the message is delivered to the store regardless of quota limits. For more information about configuring the TAS channel, see Chapter 8, "Configuring Channel Definitions."

Configuring Message Store Quotas

You set default quotas for all users by using iPlanet Console or by using the `configutil` command. You can also set quotas for individual users, family groups, and hosted domains.

This document describes how to set default quotas. For more information about setting quotas for individual users, family groups, and domains, see the *Delegated Administrator's User Guide*.

This section describes the following tasks:

- Specifying a Default User Quota
- Enabling Quota Enforcement and Notification
- Setting a Grace Period

If you want to use iPlanet Console:

1. From iPlanet Console, open the Messaging Server you want to configure.
2. Click the Configuration tab and select Message Store in the left pane.
3. Click the Quota tab in the right pane.

Specifying a Default User Quota

The default quota applies to users who do not already have individual quotas set for them. A quota set for an individual user overrides the default quota.

Console. To specify a default quota at the Console:

1. Click the Quota tab.
2. To specify a default user disk quota, for the “Default user disk quota” field, select one of the following options:

   Unlimited. Select this option if you do not want to set a default disk quota.

   Size specification. Select this option if you want to restrict the default user disk quota to a specific size. In the field beside the button, type a number, and from the drop-down list, choose Mbytes or Kbytes.
3. To specify a message number quota, in the “Default user message quota” box, type a number.
4. Click Save.

Command Line. To specify a default user disk quota for total message size:

```
configutil -o store.defaultmailboxquota -v [ -1 | number ]
```

where `-1` indicates no quota; *number* indicates a number in bytes.

To specify a default user quota for total message number:

```
configutil -o store.defaultmessagequota -v [ -1 | number ]
```

where `-1` indicates no quota; *number* indicates a number in bytes.

Enabling Quota Enforcement and Notification

You can enable or disable quota enforcement and quota notification. The action the server takes depends on how these configuration variables are set, as shown in Table 10-3.

Table 10-3 Quota Enforcement and Notification

	Enforcement On	Enforcement Off
Notification On	Messages are deferred for specified grace period; rejected if grace period expires. Messages cannot be appended to mailbox. IMAP SELECT, IMAP APPEND, SMTP sendmail mechanism and deliver command will display error message.	Messages are delivered to the store. Messages can be appended to mailbox. IMAP SELECT, IMAP APPEND, SMTP sendmail mechanism and deliver command do not display error messages.
Notification Off	Messages are deferred for specified grace period; rejected if grace period expires. Messages cannot be appended to mailbox. IMAP SELECT command, deliver command, and SMTP sendmail mechanism do not display error message. IMAP APPEND command will display error message.	Messages are delivered to the store. Messages can be appended to mailbox. IMAP SELECT, IMAP APPEND, SMTP sendmail mechanism and deliver command do not display error message.

Enabling Quota Enforcement

Console. To enable quota enforcement at the Console:

1. Click the Quota tab.
2. Check the “Enable quota enforcement” box.

   This box acts as a toggle. To disable quota enforcement, uncheck this box.
3. Click Save.

Command Line. To enable quota enforcement at the command line:

```
configutil -o store.quotaenforcement -v [ yes | no]
```

If you specify no, quotas are not enforced.

Enabling Quota Notification

Console. To enable quota notification at the Console:

1. Click the Quota tab.
2. Check the “Enable quota notification” box.

   This box acts as a toggle. To disable quota enforcement, uncheck this box.
3. Click Save.

Command Line. To enable quota notification at the command line:

```
configutil -o store.quotanotification -v [ yes | no ]
configutil -o store.quotaexceededmsg -v message
```

If the message is not set, then no quota warning message will be sent to the user.

Defining a Quota Warning Message

You can define the message that will be sent to users who have exceeded their disk quota as follows. Messages are sent to the user’s mailbox.

Console. To define a quota warning message at the Console:

1. Click the Quota tab.
2. From the drop-down list, choose the language you want to use.
3. Type the message you want to send in the message text field below the drop-down list.
4. Click Save.

Command Line. To define a quota warning message at the command line:

```
configutil -o store.quotaexceededmsg -v message
```

The message must be in RFC 822 format.

To define how often the warning message is sent:

```
configutil -o store.quotaexceedmsginterval -v number
```

where *number* indicates a number of days. For example, 3 would mean the message is sent every 3 days.

Specifying a Quota Threshold

You can send a warning message to IMAP users before they reach their disk quota by specifying a quota threshold. When a user's disk usage exceeds the specified threshold, the server sends a warning message to the user.

For IMAP users whose clients support the IMAP ALERT mechanism, the message is displayed on the user's screen each time the user selects a mailbox (a message is also written to the IMAP log).

Console. To specify a quota threshold at the Console:

1. Click the Quota tab.
2. In the "Quota warning threshold" field, enter a number for the warning threshold.

   This number represents a percentage of the allowed quota. For example, if you specify 90%, the user is warned after using 90% of the allowed disk quota. The default is 90%. To turn off this feature, enter 100%.
3. Click Save.

Command Line. To specify a quota threshold at the command line:

```
configutil -o store.quotawarn -v number
```

where *number* indicates a percentage of the allowed quota.

Setting a Grace Period

If a user mailbox exceeds the quota for allotted disk space or total number of messages, the grace period you specify determines how long messages will be held in the message queue before the server starts bouncing the messages. Messages will remain in the queue until one of the following occurs:

- The mailbox no longer exceeds the quota, at which time the server will deliver the message to the mailbox.
- The user has remained over quota longer than the specified grace period, at which time the server will bounce the message.
- The message has remained in the queue longer than the maximum message queue time.

Console. To set a grace period for how long messages are held in the queue at the Console:

1. Click the Quota tab.
2. In the “Over quota grace period” field, enter a number.
3. From the drop-down list, specify `Day(s)` or `Hour(s)`.
4. Click Save.

Command Line. To specify a quota grace period at the command line:

`configutil -o store.quotagraceperiod -v` *number*

where *number* indicates number of hours.

Specifying Aging Policies

Aging policies are another way to control disk usage on your server. You can control how long messages are stored in one or more mailboxes. If you have limited disk space, you might want to set aging policies to remove messages from the store. If you set aging policies, you should educate your users about these policies because the server will not send warning messages before it deletes messages from the store.

You can create aging rules based on the following criteria:

- Number of messages in the mailbox
- Total size of the mailbox
- Number of days that messages remain in the mailbox
- Number of days that messages exceeding a given size remain in the mailbox

If you specify more than one rule for a mailbox, all expiration rules will apply, but the most restrictive rule takes precedence. For example, assume two rules apply to a single mailbox. The first rule allows 1000 messages; the second rule allows 500 messages. When expiration occurs, the server will delete messages from the mailbox until 500 remain. For another example, if the first rule allows a message size of 100,000 bytes for 3 days and the second rule allows a message size of 1000 bytes for 12 days, the resulting union of rules allows a message size of 100,000 bytes for 3 days. The server will delete messages over 100,000 bytes that have been in the mailbox over 3 days. If you want to ensure that a specific rule is the only rule for a particular mailbox or set of mailboxes, use the Exclusive parameter.

Console. To create a new rule by using Console:

1. From iPlanet Console, open the Messaging Server you want to configure.
2. Click the Configuration tab and select Message Store in the left pane.
3. Click the Aging tab in the right pane.
4. Click Add to go to the Add Rule window.
5. Enter a name for the new rule.
6. Specify the target folders for which this rule applies.

   You can enter a path name, filename, or partial string. You can use IMAP wildcards as follows:

   * - Match any series of characters.
   % - Match any series of characters except slash characters.

   The new rule applies only to folders matching the pattern you specify.
7. If this rule is to be the only rule applied to the target folders, click the Exclusive selection box.
8. If you want to create a rule based on folder size, do the following:
   - In the “Message count” field, specify the maximum number of messages that will be retained in a folder before the oldest messages are removed.
   - In the “Folder size” field, specify a number for the folder size; from the associated drop-down list, choose Mbyte(s) or KByte(s).

   When the specified folder size is exceeded, the server removes the oldest messages until this size is no longer exceeded.
9. If you want to create a rule based on message age, in the “Number of days” field, specify a number to indicate how long messages should remain in the folder.
10. If you want to create a rule based on message size:
    - In the “Message size limit” field, enter a number to indicate the maximum size message allowed in the folder; from the associated drop-down list, choose Mbytes or Kbytes.
    - In the “Grace period” field, enter a number to indicate how long over-sized messages should remain in the folder.

    After the grace period, the server deletes messages that exceed the maximum size.

11. Click OK to add the new rule to the Aging Rule list and dismiss the Add window.

12. Click Save to submit and preserve the current Aging Rule list.

Command Line. To create a new rule at the command line, use the following commands where *name* represents the name you give the rule:

To specify the target folders for which this rule applies:

```
configutil -o store.expirerule.name.folderpattern -v pattern
```

For example, the pattern `user/*` matches everything; the patter `user/%@siroe.com/*` matches all folders for all users in the domain `siroe.com`; and the pattern `user/%/Trash` matches the Trash folder for all users.

To specify that this rule is to be the only rule applied to the target folders:

```
configutil -o store.expirerule.name.exclusive -v [ yes | no ]
```

To specify the maximum number of messages that will be retained in a folder before the oldest messages are removed:

```
configutil -o store.expirerule.name.messagecount -v number
```

To specify the folder size:

```
configutil -o store.expirerule.name.foldersizebytes -v number
```

where *number* is a size in bytes.

To specify message age:

```
configutil -o store.expirerule.name.messagedays -v number
```

where *number* indicates the number of days.

To specify message size:

```
configutil -o store.expirerule.name.messagesize -v number
```

where *number* is a size in bytes.

To indicate how long over-sized messages should remain in the folder:

```
configutil -o store.expirerule.name.messagesizedays -v number
```

where *number* indicates number of days.

Configuring Message Store Partitions

All user mailboxes are stored by default in the `msg-`*instance*`/store/partition/` directory. The `partition` directory is a logical directory that might contain a single subpartition or multiple subpartitions. The subpartitions might map to a single physical drive or to multiple physical drives. At start-up time, the `partition` directory contains one subpartition called the `primary` partition.

You can add partitions to the `partition` directory as necessary. For example, you might want to partition a single disk to organize your users as follows:

`msg-`*instance*`/store/partition/mkting/`
`msg-`*instance*`/store/partition/eng/`
`msg-`*instance*`/store/partition/sales/`

As disk storage requirements increase, you might want to map these partitions to different physical disk drives.

You should limit the number of mailboxes on any one disk. Distributing mailboxes across disks improves message delivery time (although it does not necessarily change the SMTP accept rate). The number of mailboxes you allocate per disk depends on the disk capacity and the amount of disk space allocated to each user. For example, you can allocate more mailboxes per disk if you allocate less disk space per user.

If your message store requires multiple disks, you can use RAID (Redundant Array of Inexpensive Disks) technology to ease management of multiple disks. With RAID technology, you can spread data across a series of disks but the disks appear as one logical volume so disk management is simplified. You might also want to use RAID technology for redundancy purposes; that is, to duplicate the store for failure recovery purposes.

NOTE To improve disk access, the message store and the message queue should reside on separate disks.

Adding a Partition

When adding a partition, you specify both an absolute physical path where the partition is stored on disk and a logical name, called the partition nickname.

The partition nickname allows you to map users to a logical partition name regardless of the physical path. When setting up user accounts and specifying the message store for a user, you can use the partition nickname. The name you enter must be an alphanumeric name and must use lowercase letters.

To create and manage the partition, the user ID used to run the server must have permission to write to the location specified in the physical path.

NOTE After adding a partition, you must stop then restart the server to refresh the configuration information.

Console. To add a partition to the store by using the Console:

1. From iPlanet Console, open the Messaging Server you want to configure.
2. Click the Configuration tab and select Message Store in the left pane.
3. Click the Partition tab in the right pane.
4. Click the Add button.
5. Enter the Partition nickname.

   This is the logical name for the specified partition.
6. Enter the Partition path.

   This is the absolute path name for the specified partition.
7. To specify this as the default partition, click the selection box labeled Make This the Default Partition.
8. Click OK to submit this partition configuration entry and dismiss the window.
9. Click Save to submit and preserve the current Partition list.

Command Line. To add a partition to the store at the command line:

```
configutil -o store.partition.nickname.path -v path
```

where *nickname* is the logical name of the partition and *path* indicates the absolute path name where the partition is stored.

To specify the path of the default primary partition:

```
configutil -o store.partition.primary.path -v path
```

Moving Mailboxes to a Different Disk Partition

By default, mailboxes are created in the `primary` partition. If the partition gets full, additional messages cannot be stored. There are several ways to address the problem:

- Reduce the size of user mailboxes
- If you are using volume management software, add additional disks
- Create additional partitions ("Adding a Partition," on page 254) and move mailboxes to the new partitions

If possible, we recommend adding additional diskspace to a system using volume management software since this procedure is the most transparent for the user. However, you may also move mailboxes to a different partition by doing the following:

1. Make sure user is disconnected from their mailbox during the migration process. This can be done by informing the user to log off and stay off during mailbox move, or, by setting the `mailAllowedServiceAccess` attribute so that POP, IMAP and HTTP services are disallowed after they are logged off. (See http://docs.iplanet.com/docs/manuals/messaging/ims50/pg/users.htm#19110.)

NOTE Setting `mailAllowedServiceAccess` to disallow POP, IMAP,HTTP access does not disconnect any open connections to the mailbox. You must make sure that all connections are closed prior to the moving mailboxes.

---

2. Move the user mailbox with the following command:

   `mboxutil -r user/<`*`userid`*`>/INBOX user/<`*`userid`*`>/INBOX <`*`partition_name`*`>`

   Example:

   `mboxutil -r user/ofanning/INBOX user/ofanning/INBOX secondary`

3. Set the `mailMessageStore` attribute in the moved user's LDAP entry to the name of the new partition.

   Example: `mailMessageStore: secondary`

4. Inform the user that message store connection is now allowed. If applicable, change the `mailAllowedServiceAccess` attribute to allow POP, IMAP and HTTP services.

Performing Maintenance and Recovery Procedures

This section provides information about the utilities you use to perform maintenance and recovery tasks for the message store. You should always read your postmaster mail for warnings and alerts that the server might send. You should also monitor the log files for information about how the server is performing. For more information about log files, see Chapter 12, “Logging and Log Analysis.”

This section contains the following:

- Managing Mailboxes
- Monitoring Quota Limits
- Monitoring Disk Space
- Using the stored Utility
- Repairing Mailboxes and the Mailboxes Database
- Moving a User’s Account

Managing Mailboxes

This section describes the following utilities for managing and monitoring mailboxes: `mboxutil`, `hashdir`, `readership`.

The mboxutil Utility

You use the `mboxutil` command to perform typical maintenance tasks on mailboxes. These tasks include the following:

- List mailboxes
- Create mailboxes
- Delete mailboxes
- Rename mailboxes
- Move mailboxes from one partition to another

You can also use the `mboxutil` command to view information about quotas. For more information, see “Monitoring Quota Limits” on page 260.

Table 10-4 lists the mboxutil commands. For detailed syntax and usage requirements, see the *Messaging Server Reference Manual*.

Table 10-4 mboxutil Options

Option	Description
`-a`	Lists all user quota information.
`-c` *mailbox*	Creates the specified mailbox.
`-d` *mailbox*	Deletes the specified mailbox.
`-g` *group*	Lists quota information for the specified group.
`-k` *mailbox cmd*	Locks the specified mailbox at the folder level; runs the specified command; after command completes, unlocks the mailbox.
`-l`	Lists all of the mailboxes on a server.
`-p` *pattern*	When used in conjunction with the `-l` option, lists only those mailboxes with names that match *pattern*. You can use IMAP wildcards.
`-q` *domain*	Lists quota information for the specified domain.
`-r` *oldname newname* [*partition*]	Renames the mailbox from *oldname* to *newname*. To move a folder from one partition to another, specify the new partition with the *partition* option.
	This option can be used to rename a user. For example, `mboxutil -r user/user1/INBOX user/user2/INBOX` moves all mail and mailboxes from user1 to user2, and new messages will appear in the new INBOX. . \(If user2 already exists, this operation will fail.)
`-u` *user*	Lists user information such as current size of mail store, quota (if one has been set), and percentage of quota currently in use.
`-x`	When used in conjunction with the -l option, shows the path and access control for a mailbox.

Mailbox Naming Conventions

You must specify mailbox names in the following format: `user/`*userid*`/`*mailbox*, where *userid* is the user that owns the mailbox and *mailbox* is the name of the mailbox. For hosted domains, *userid* is *uid@domain*.

For example, the following command creates the mailbox named INBOX for the user whose user ID is crowe. INBOX is the default mailbox for mail delivered to the user crowe.

```
mboxutil -c user/crowe/INBOX
```

Important: The name INBOX is reserved for each user's default mailbox. INBOX is the only folder name that is case-insensitive. All other folder names are case-sensitive.

Examples

To list all mailboxes for all users:

```
mboxutil -l
```

To list all mailboxes and also include path and ACL information:

```
mboxutil -l -x
```

To create the default mailbox named INBOX for the user daphne:

```
mboxutil -c user/daphne/INBOX
```

To delete a mail folder named projx for the user delilah:

```
mboxutil -d user/delilah/projx
```

To delete the default mailbox named INBOX and *all mail folders* for the user druscilla:

```
mboxutil -d user/druscilla/INBOX
```

To rename the mail folder memos to memos-april for the user desdemona:

```
mboxutil -r user/desdemona/memos user/desdemona/memos-april
```

To lock a mail folder named legal for the user dulcinea:

```
mboxutil -k user/dulcinea/legal cmd
```

where *cmd* is the command you wish to run on while the folder is locked.

To move the mail account for the user dimitria to a new partition:

```
mboxutil -r user/dimitria/INBOX user/dimitria/INBOX partition
```

where *partition* specifies the name of the new partition.

To move the mail folder named personal for the user dimitria to a new partition:

```
mboxutil -r user/dimitria/personal user/dimitria/personal partition
```

The hashdir Utility

The mailboxes in the message store are stored in a hash structure for fast searching. Consequently, to find the directory that contains a particular user's mailbox, use the `hashdir` utility.

This utility identifies the directory that contains the message store for a particular account. This utility reports the relative path to the message store, such as `d1/a7/`. The path is relative to the directory level just before the one based on the user ID. The utility sends the path information to the standard output.

For example, to find the relative path to the mailbox for user `crowe`:

```
hashdir crowe
```

The readership Utility

The `readership` utility reports on how many users other than the mailbox owner have read messages in a shared IMAP folder.

An owner of a IMAP folder may grant permission for others to read mail in the folder. A folder that others are allowed to access is called a *shared folder*. Administrators can use the `readership` utility to see how many users other than the owner are accessing a shared folder.

This utility scans all mailboxes and produces one line of output per shared folder, reporting the number of readers followed by a space and the name of the mailbox.

Each reader is a distinct authentication identity that has selected the shared folder within the past specified number of days. Users are not counted as reading their own personal mailboxes. Personal mailboxes are not reported unless there is at least one reader other than the folder's owner.

For example, the following command counts as a reader any identity that has selected the shared IMAP folder within the last 15 days:

```
readership -d 15
```

Monitoring Quota Limits

You can monitor quota usage and limits by using the `mboxutil` utility. The `mboxutil` utility generates a report that lists defined quotas and limits, and provides information on quota usage. Quotas and usage figures are reported in kilobytes.

For example, the following command lists all user quota information:

```
mboxutil -a
```

The next example lists quota information for the user crowe:

```
mboxutil -u crowe
```

The next example lists quota information for a the domain `siroe.com`:

```
mboxutil -q siroe.com
```

Monitoring Disk Space

You can specify how often the system should monitor disk space and under what circumstances the system should send a warning. To configure disk space monitoring and notification, you use the `configutil` command to set the alarm space attributes, which are described in Table 10-5.

Table 10-5 Disk Space Alarm Attributes

Disk Space Attributes	Default Value
`alarm.diskavail.msgalarmstatinterval`	3600 seconds
`alarm.diskavail.msgalarmthreshold`	10%
`alarm.diskavail.msgalarmwarninginterval`	24 hours

For example, if you want the system to monitor disk space every 600 seconds, specify the following command:

```
configutil -o alarm.diskavail.msgalarmstatinterval -v 600
```

If you want to receive a warning whenever available disk space falls below 20%, specify the following command:

```
configutil -o alarm.diskavail.msgalarmthreshold -v 20
```

For more information about setting alarm attributes, see the *Messaging Server Reference Manual*.

Using the stored Utility

The `stored` utility performs the following monitoring and maintenance tasks for the server:

- Background and daily messaging tasks
- Deadlock detection and rollback of deadlocked database transactions
- Cleanup of temporary files on startup
- Implementation of aging policies
- Periodic monitoring of server state, disk space, service response times, and so on
- Issuing of alarms if necessary

The `stored` utility automatically performs cleanup and expiration operations once a day at 11 PM. You can choose to run additional cleanup and expiration operations.

You can also use the `stored` utility to create a backup of the mailboxes database and log files. If the database becomes corrupt, you can use the backup copy to replace the database without having to reconstruct the database. To create a backup of the database, you use the `configutil` command to specify values for the following parameters:

`configutil -o local.store.snapshotinterval -v` *`number`*

where *number* specifies how often `stored` will back up the database; *number* indicates a time interval in minutes.

`configutil -o local.store.snapshotpath -v` *`path`*

where *path* indicates the location of the backup copy.

Table 10-6 lists the `stored` options. Some common usage examples follow the table. For detailed syntax and usage requirements, see the *Messaging Server Reference Manual*.

Table 10-6 stored Options

Option	Description
`-c`	Performs one cleanup pass to erase expunged messages. Runs once, then exits. The `-c` option is a one-time operation, so you do not need to specify the `-1` option.

Table 10-6 stored Options

Option	Description
`-d`	Run as daemon. Performs system checks and activates alarms, deadlock detection, and database repair.
`-1`	Run once, then exit.
`-n`	Run in trial mode only. Does not actually age or cleanup messages. Runs once, then exits.
`-v`	Verbose output.
`-v -v`	More verbose output.

To test expiration policies:

```
stored -n
```

To perform a single aging and cleanup pass:

```
stored -l -v
```

If you want to change the time of the automatic cleanup and expiration operations, use the `configutil` utility as follows:

```
configutil -o store.expirestart -v 21
```

Occasionally, you might need to restart the `stored` utility; for example, if the mailbox list database becomes corrupted. To restart `stored` on UNIX, use the following commands at the command line:

server-root`/msg-`*instance*`/stop-msg store`
server-root`/msg-`*instance*`/start-msg store`

If any server daemon crashes, you must stop all daemons and restart all daemons including `stored`.

Repairing Mailboxes and the Mailboxes Database

If one or more mailboxes becomes corrupt, you can use the `reconstruct` utility to rebuild the mailboxes or the mailboxes database, and repair any inconsistencies.

The `reconstruct` utility rebuilds one or more mailboxes, or the master mailbox file, and repairs any inconsistencies. You can use this utility to recover from almost any form of data corruption in the mail store. Note that low-level database repair, such as completing transactions and rolling back incomplete transactions is performed with `stored -d`.

Table 10-7 lists the `reconstruct` options. For detailed syntax and usage requirements, see the *Messaging Server Reference Manual*.

Table 10-7 reconstruct Options

Option	Description
`-f`	Forces a reconstruct without performing a consistency check. You can use this option to force a reconstruct even if the consistency check passes.
`-m`	Performs a high-level consistency check and repair of the mailboxes database. Examines every mailbox in the spool area, adding or removing entries from the mailboxes database as appropriate. Prints a message to the standard output file whenever it adds or removes an entry from the database. This option should be run with `stored -d` to ensure that the database is checkpointed as it is reconstructed.
`-n`	Performs a consistency check, but makes no repairs even if a problem is found. This option is used primarily for debugging, but it can also be used to check the store.
`-o`	Checks for orphaned accounts. Searches for inboxes in the current messaging server host which do not have corresponding entries in LDAP. For example, the `-o` option would find inboxes of owners who have been deleted from LDAP or moved to a different server host. For each orphaned account it finds, `reconstruct` writes the following command to the standard output: `mboxutil -d user/`*`userid`*`/INBOX`
`-o -d` *filename*	If `-d` *filename* is specified with the `-o` option, `reconstruct` opens the specified file and writes the `mboxutil -d` commands into that file. The file may then be turned into a script file to delete the orphaned accounts.
`-p` *partition*	Specifies a partition name. You can use this option on the first usage of `reconstruct`.

Table 10-7 reconstruct Options

Option	Description
`-q`	Fixes any inconsistencies in the quota subsystem, such as mailboxes with the wrong quota root or quota roots with the wrong quota usage reported. The `-q` option can be run while other server processes are running.
`-r [`*mailbox*`]`	Performs a consistency check and repairs the partition area of the specified mailbox or mailboxes if necessary. The `-r` option also repairs the sub-mailboxes within the specified mailbox if necessary. If you specify `-r` with no mailbox argument, the utility repairs the spool areas of all mailboxes within the database if necessary.

Rebuilding Mailboxes

To rebuild mailboxes, use the `-r` option. You should use this option when:

- Accessing a mailbox returns one of the following errors: “System I/O error” or “Mailbox has an invalid format”.
- Accessing a mailbox causes the server to crash.
- Files have been added to or removed from the spool directory.

With the 5.0 release, `reconstruct -r` first runs a consistency check. It reports any consistencies and rebuilds only if it detects any problems. Consequently, performance of the reconstruct utility is improved with this release.

You can use `reconstruct` as described in the following examples:

To rebuild the spool area for the mailboxes belonging to the user `daphne`, use the following command:

```
reconstruct -r user/daphne
```

To rebuild the spool area for all mailboxes listed in the mailbox database:

```
reconstruct -r
```

You must use this option with caution, however, because rebuilding the spool area for all mailboxes listed in the mailbox database can take a very long time for large message stores. (See “reconstruct Performance” on page 267.) A better method for failure recovery might be to use multiple disks for the store. If one disk goes down, the entire store does not. If a disk becomes corrupt, you need only rebuild a portion of the store by using the `-p` option as follows:

```
reconstruct -r -p subpartition
```

To rebuild mailboxes listed in the command-line argument only if they are in the `primary` partition:

```
reconstruct -p primary mbox1 mbox2 mbox3
```

If you do need to rebuild all mailboxes in the primary partition:

```
reconstruct -r -p primary
```

If you want to force reconstruct to rebuild a folder without performing a consistency check, use the `-f` option. For example, the following command forces a reconstruct of the user folder `daphne`:

```
reconstruct -f -r user/daphne
```

To check all mailboxes without fixing them, use the `-n` option as follows:

```
reconstruct -r -n
```

Checking and Repairing Mailboxes

To perform a high-level consistency check and repair of the mailboxes database:

```
reconstruct -m
```

You should use the `-m` option when:

- One or more directories were removed from the store spool area, so the mailbox database entries also need to be removed.
- One or more directories were restored to the store spool area, so the mailbox database entries also need to be added.
- The `stored -d` option is unable to make the database consistent.

  If the `stored -d` option is unable to make the database consistent, you should perform the following steps in the order indicated:

  - Shut down all servers.
  - Remove all files in *server-root*`/msg-`*instance*`/store/mboxlist`.
  - Restart the server processes.
  - Run `reconstruct -m` to build a new mailboxes database from the contents of the spool area.

Removing Orphaned Accounts

To search for orphaned accounts (orphaned accounts are mailboxes that do not have corresponding entries in LDAP):

```
reconstruct -o
```

Command output follows:

```
reconstruct: Start checking for orphaned mailboxes
mboxutil -d user/test/annie/INBOX
mboxutil -d user/test/oliver/INBOX
reconstruct: Found 2 orphaned mailbox(es)
reconstruct: Done checking for orphaned mailboxes
```

To create a file listing orphaned mailboxes that can be turned into a script file that deletes the orphaned mailboxes, where the file is to be named `orphans.cmd`:

```
reconstruct -o -d orphans.cmd
```

Command output follows:

```
reconstruct: Start checking for orphaned mailboxes
reconstruct: Found 2 orphaned mailbox(es)
reconstruct: Done checking for orphaned mailboxes
```

reconstruct Performance

The time it takes `reconstruct` to perform an operation depends on a number of factors including:

- The kind of operation being performed and the options chosen
- Disk performance
- The number of folders when running `reconstruct -m`
- The number of messages when running `reconstruct -r`
- The overall size of the message store
- What other processes the system is running and how busy the system is
- Whether or not there is ongoing POP, IMAP, HTTP, or SMTP activity

The `reconstruct -r` option performs an initial consistency check; this check improves `reconstruct` performance depending on how many folders must be rebuilt.

In one example with approximately 2400 users, a message store of 85GB, and concurrent POP, IMAP, or SMTP activity on the server:

- `reconstruct -m` took about 1 hour
- `reconstruct -r -f` took about 18 hours

NOTE A `reconstruct` operation may take significantly less time if the server is not performing ongoing POP, IMAP, HTTP, or SMTP activity.

Moving a User's Account

The `MoveUser` utility moves a user's account from one messaging server to another. When user accounts are moved from one messaging server to another, it's also necessary to move the user's mailboxes and the messages they contain from one server to the other. In addition to moving mailboxes from one server to another, `MoveUser` updates entries in the Directory Server to reflect the user's new `mailhost` name and message store path.

To use the `MoveUser` utility, the user must authenticate by including the `-a` option in the `MoveUser` command. Any valid message store administrator can run the `MoveUser` command. Users are granted store administrator privileges as follows:

- You can grant message store administration privileges for a specific Messaging Server by using iPlanet console. For more information, see "Specifying Administrator Access to the Store" on page 242.
- DA Top-level administrators are automatically granted message store administration privileges for the entire mail system.
- DA domain administrators are automatically granted message store administration privileges for the domain.

Table 10-8 lists the `MoveUser` options. Usage examples follow the table. For detailed syntax and usage requirements, see the *Messaging Server Reference Manual*.

Table 10-8 MoveUser Options

Option	Description
`-a` *destproxyuser*	ProxyAuth user for destination messaging server.
`-A`	Do not add an alternate email address to the LDAP entry.

Table 10-8 MoveUser Options

Option	Description
`-d` *destmailhost*	Destination messaging server.
`-D` *binddn*	Binding *dn* to the given *ldapURL*.
`-F`	Delete messages in source messaging server after successful move of mailbox. (If not specified, messages will be left in source messaging server.)
`-h`	Display help for this command.
`-l` *ldapURL*	URL to establish a connection with the Directory Server:
`-L`	Add a license for Messaging Server if not already set.
`-m` *destmaildrop*	Message store path for destination messaging server. (If not specified, the default is used.)
`-n` *msgcount*	Number of messages to be moved at once.
`-o` *srcmaildrop*	Message store path for source messaging server. (If not specified, the default is used.)
`-p` *srcproxypasswd*	ProxyAuth password for source messaging server.
`-s` *srcmailhost*	Source messaging server.
`-S`	Do not set new message store path for each user.
`-u` *uid*	User ID for the user mailbox that is to be moved. Cannot be used with `-l` option.
`-U` *newuid*	New (renamed) user ID that the mailbox is to be moved to.
`-v` *destproxypwd*	ProxyAuth password for destination messaging server.
`-w` *bindpasswd*	Binding password for the *binddn* given in the `-D` option.
`-x` *srcproxyuser*	ProxyAuth user for source messaging server.

To move all users from `host1` to `host2`, based on account information in the Directory Server `siroe.com`:

```
MoveUser -l \
  "ldap://siroe.com:389/o=Siroe.com???\
  (mailhost=host1.domain.com)" \
  -D "cn=Directory Manager" -w password -s host1 -x admin \
  -p password -d host2 -a admin -v password
```

To move one user from `host1` which uses port 150 to `host2`, based on account information in the Directory Server `siroe.com`:

```
MoveUser -l \
  "ldap://siroe.com:389/o=Siroe.com???(uid=userid)" \
  -D "cn=Directory Manager" -w password -s host1:150 -x admin \
  -p password -d host2 -a admin -v password
```

To move a group of users whose uid starts with letter ‘s’ from `host1` to `host2`, based on account information in the Directory Server `server1.siroe.com`:

```
MoveUser -l \
  "ldap://server1.siroe.com:389/o=Siroe.com???(uid=s*)" \
  -D "cn=Directory Manager" -w password -s host1 -x admin \
  -p password -d host2 -a admin -v password
```

To move a user’s mailboxes from `host1` to `host2` when the user ID of `admin` is specified in the command line:

```
MoveUser -u uid \
  -s host1 -x admin -p password \
  -d host2 -a admin -v password
```

To move a user named `aldonza` from `host1` to a new user ID named `dulcinea` on `host2`:

```
MoveUser -u aldonza -U dulcinea \
  -s host1 -x admin -p password \
  -d host2 -a admin -v password
```

Backing Up and Restoring the Message Store

Backup and restore is one of the most common and important administrative tasks. You must implement a backup and restore policy for your message store to ensure that data is not lost if problems such as the following occur:

- System crashes
- Hardware failure
- Accidental deletion of messages or mailboxes
- Problems when reinstalling or upgrading a system
- Natural disasters (for example, earthquakes, fire, hurricanes)

You also need to back up data when migrating users.

Messaging Server provides command-line utilities that allow you to back up and restore the message store. Messaging Server also provides an integrated solution with Legato Networker.

Messaging Server provides a single-copy backup procedure. Regardless of how many user folders contain a particular message, during backup, the message file is backed up only once using the first message file found. The second message copy is backed up as a link to the name of the first message file, and so on. The backup utility maintains a hash table of all messages using the device and inode of the message files as the index. This method does have implications when restoring data, however. For more information, see “Considerations for Partial Restore” on page 275.

This section contains the following subsections:

- “Creating a Backup Policy” on page 272
- “Creating Backup Groups” on page 273
- “Messaging Server Backup and Restore Utilities” on page 274
- “Considerations for Partial Restore” on page 275
- “Using Legato Networker” on page 277

Creating a Backup Policy

Your backup policy will depend on several factors, such as:

- Peak Business Loads
- Full and Incremental Backups
- Parallel or Serial Backups

Peak Business Loads

You need to take into account peak business loads when scheduling backups for your system. For example, backups are probably best scheduled for early morning hours such as 2:00 a.m.

Full and Incremental Backups

Incremental backups will scan the store for changed data and back up only what has changed. Full backups will back up the entire message store. You need to determine how often the system should perform full as opposed to incremental backups. You'll probably want to perform incremental backups as a daily maintenance procedure. Full backups are more appropriate when you need to move or migrate data.

Parallel or Serial Backups

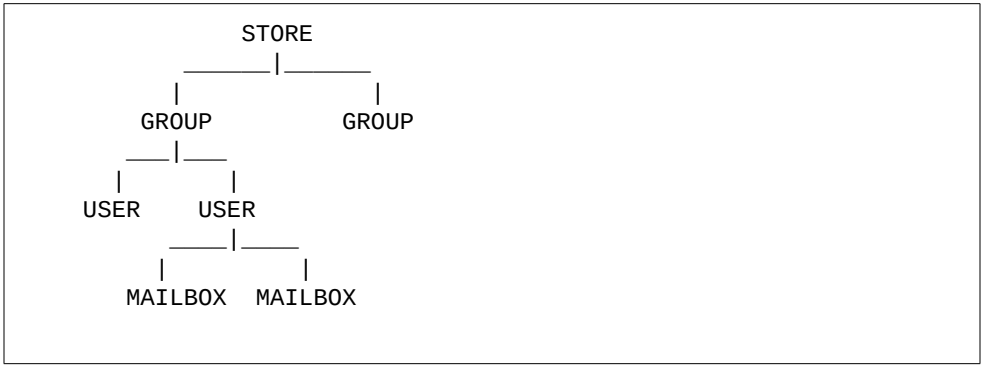
When user data is stored on multiple disks, you can back up user groups in parallel if you wish. Depending on system resources, parallel backups can speed up the overall backup procedure. However, you might want to use serial backups, for example, if you do not want to impact the server's performance. Whether to use parallel or serial backups can depend on many factors, including system load, hardware configuration, how many tape drives are available, and so on.

Creating Backup Groups

By organizing users into groups, you can improve backup management. For example, you can specify separate backup sessions for each group. Or you can choose to back up several groups in parallel.

Assuming user messages are stored according to user last name, users whose names begin with A through F would represent a backup group while users whose last names begin with G through M would represent another backup group.

The logical view of the message store looks like the following:



By cataloging users into groups, you can improve backup management. For example, you can specify separate backup sessions for each group. Or you can choose to back up several groups in parallel. For more information about creating backup groups, see “Creating Backup Groups” on page 273.

If you want to create backup groups, you need to create a configuration file in which to store your group definitions. This file must be named `backup-groups.conf` and it must be stored in the following directory:

serverRoot`/msg-`*instance*`/config/backup-groups.conf`

The format of this file is:

```
groups=definitions
groups=definitions
.
.
.
```

For example, if you want to group users by the first letter of their user IDs, use the following definitions:

```
groupA=a*
groupB=b*
groupC=c*
```

Backup object naming uses the logical structure of the message store, as follows:

/server/group/user/mailbox

Where *server* is the message store instance name. For example: `siroe`

Messaging Server includes one predefined backup group that is available without creating the `backup-groups` configuration file. This group is called `ALL`; it includes all users.

Messaging Server Backup and Restore Utilities

To back up and restore your data, Messaging Server provides the `imsbackup` and `imsrestore` utilities.

Please note that the `imsbackup` and `imsrestore` utilities are not intended to provide a general-purpose backup facility. These utilities do not have the advanced features found in general purpose tools like Legato Networker. For example, the utilities have only very limited support for tape auto-changers. They cannot write a single store to multiple concurrent devices. Comprehensive backup will be achieved via plug-ins to generalized tools like Legato Networker. For more information about using Legato Networker, see “Using Legato Networker” on page 277.

The imsbackup Utility

With `imsbackup`, you can write selected contents of the Message Store to any serial device, including magnetic tape, a UNIX pipe, or a plain file. The backup or selected parts of the backup may later be recovered by using the `imsrestore` utility. The output of `imsbackup` can be piped to `imsrestore`.

To perform a back up, issue the `imsbackup` command as shown in the following example, which backs up `user1` to `backupfile`:

```
imsbackup -f backupfile /mystore/ALL/user1
```

This command uses the default blocking factor of 20. For a complete syntax description of the `imsbackup` command, see the *Messaging Server Reference Manual*.

The imsrestore Utility

To restore messages from the backup device, use the `imsrestore` command. For example, the following command restores messages for `user1` from the file `backupfile`.

```
imsrestore -f backupfile /mystore/ALL/user1
```

For a complete syntax description of the `imsbackup` command, see the *Messaging Server Reference Manual*.

Considerations for Partial Restore

This single-copy backup procedure has implications when restoring messages as follows:

- **Full Restore.** During a full restore, linked messages will still point to the same inode as the message file to which they are linked.
- **Partial Backup/Restore.** During a partial backup and partial restore, however, the single-copy characteristic of the message store might not be preserved.

Assume there are three messages belonging to three users A, B, and C, as follows:

A/INBOX/1
B/INBOX/1
C/INBOX/1

Example 1. In the first example, the system performs a partial backup and full restore procedure as follows:

1. Back up users B and C.
2. Delete users B and C.
3. Restore the backup data from step 1.

In this example, B/INBOX/1 and C/INBOX/1 are assigned a new inode number and the message data is written to a new place on the disk. Only one message is restored; the second message is a hard link to the first message.

Example 2. In this example, the system performs a full backup and a partial restore as follows:

1. Perform full backup.
2. Delete user A.
3. Restore user A.

A/INBOX/1 is assigned a new inode number.

Example 3. In this example, partial restore might require more than one attempt:

1. Perform full backup.

   B/INBOX/1 AND C/INBOX/1 are backed up as links to A/INBOX/1.
2. Delete users A and B.
3. Restore user B.

   The restore utilities ask the administrator to restore A/INBOX first.
4. Restore users A and B.
5. Delete user A (optional).

NOTE If you want to ensure that all messages are restored for a partial restore, you can run the `imsbackup` command with the `-i` option. The `-i` option backs up every message multiple times if necessary. This option is most useful in POP environments.

Using Legato Networker

Messaging Server includes a backup API that provides an interface with third-party backup tools, such as Legato Networker. The physical message store structure and data format are encapsulated within the backup API. The backup API interacts directly with the message store. It presents a logical view of the message store to the backup service. The backup service uses the conceptual representation of the message store to store and retrieve the backup objects.

Messaging Server provides an Application Specific Module (ASM) that can be invoked by the Legato Networker's `save` and `recover` commands to back up and restore the message store data. The ASM then invokes the Messaging Server `imsbackup` and `imsrestore` utilities.

NOTE This section provides information about how to use Legato Networker with the Messaging Server message store. To understand the Legato Networker interface, see your Legato documentation.

Backing Up Data Using Legato Networker

To perform backups of the Messaging Server message store using Legato Networker, you must perform the following preparatory steps before invoking the Legato interface:

1. Create a symbolic link from `/usr/lib/nsr/imsasm` to *serverRoot*`/msg-`*instance*`/bin/imsasm`
2. From Sun or Legato, obtain a copy of the `nsrfile` binary and copy it to the following directory:

   `/usr/lib/nsr/nsrfile`
3. If you want to back up users by groups, perform the following steps:
   a. Create a backup group file as described in "Creating Backup Groups" on page 273.
   b. To verify your configuration, run `mkbackupdir.sh`.

      Look at the directory structure in *serverRoot*`/msg-`*instance*`/backup`. The structure should look similar to that shown in Figure 10-2.

   Note that if you do not specify a `backup-groups.conf` file, the backup process will use the default backup group `ALL` for all users.

4. In the directory `/nsr/res/`, create a `res` file for your savegroup to invoke the `mkbackupdir.sh` script before the backup. See Figure 10-3 for an example.

NOTE Legato Networker has a limitation of 64 characters for the saveset name. By default `mkbackupdir.sh` will create the store image under the *serverRoot*`/msg-`*instance*`/backup` directory. If the name of this directory plus the logical name of the mailbox (for example, `siroe/groupA/fred`) is greater than 64 characters, then you must run `mkbackupdir.sh -p`. For example the following command will create the backup image under the directory "`/`":

```
mkbackupdir.sh -p /
```

Figure 10-2 shows a sample backup groups directory structure.

Figure 10-2 Backup Group Directory Structure

```
siroe-groupA-a1
            -a2
     -groupB-b1
            -b2
     -groupC-c1
            -c2
```

Figure 10-3 shows a sample `res` file named `IMS.res` in the `nsr` directory:

Figure 10-3 Sample res File

```
type: savepnpc
precmd: "echo mkbackupdir started",
   "/usr/siroe/server5/msg-siroe/bin/mkbackupdir.sh"
pstcmd: "echo imsbackup Completed";
timeout: "12:00 pm";
```

You are now ready to run the Legato Networker interface as follows:

1. Create the Messaging Server savegroup if necessary.

a. Run `nwadmin`.

b. Select Customize | Group | Create.

2. Create a backup client using `savepnpc` as the backup command:

a. Set the saveset to the directory created by `mkbackupdir`.

For a single session backup, use *serverRoot*`/msg-`*instance*`/backup`

For parallel backups, use *serverRoot*`/msg-`*instance*`/backup/`*server*`/`*group*

Be sure you've already created *group* as defined in "Creating Backup Groups" on page 273.

You must also set the parallelism to the number of backup sessions.

See "Example. Creating A Backup Client in Networker" on page 279.

3. Select Group Control | Start to test your backup configuration.

Example. Creating A Backup Client in Networker. To create a backup client in Networker. From `nwadmin`, select Client | Client Setup | Create:

```
Name: siroe
Group: IMS
Savesets:/usr/siroe/server5/msg-siroe/backup/siroe/groupA
   /usr/siroe/server5/msg-siroe/backup/siroe/groupB
   /usr/siroe/server5/msg-gotmail/backup/gotmail/groupC
         .
         .
Backup Command:savepnpc
Parallelism: 4
```

Restoring Data Using Legato Networker

To recover data, you can use the Legato Networker `nwrecover` interface or the `recover` command-line utility. The following example recovers user a1's INBOX:

```
recover -a -f -s siroe
/usr/siroe/server5/msg-siroe/backup/siroe/groupA/a1/INBOX
```

The next example recovers the entire message store:

```
recover -a -f -s siroe /usr/siroe/server5/msg-siroe/backup/siroe
```

Chapter 11

Configuring Security and Access Control

iPlanet Messaging Server supports a full range of flexible security features that allow you to keep messages from being intercepted, prevent intruders from impersonating your users or administrators, and permit only specific people access to specific parts of your messaging system.

The Messaging Server security architecture is part of the security architecture of iPlanet servers as a whole. It is built on industry standards and public protocols for maximum interoperability and consistency. To implement Messaging Server security policies, therefore, you will need not only this chapter but several other documents as well. In particular, information in *Managing Servers with Netscape Console* is required for setting up Messaging Server security.

This chapter contains the following sections:

- About Server Security
- About HTTP Security
- Configuring Authentication Mechanisms
- User Password Login
- Configuring Encryption and Certificate-Based Authentication
- Configuring Administrator Access to Messaging Server
- Configuring Client Access to POP, IMAP, and HTTP Services
- Configuring Client Access to SMTP Services

About Server Security

Server security encompasses a broad set of topics. In most enterprises, ensuring that only authorized people have access to the servers, that passwords or identities are not compromised, that people do not misrepresent themselves as others when communicating, and that communications can be held confidential when necessary are all important requirements for a messaging system.

Perhaps because the security of server communication can be compromised in many ways, there are many approaches to enhancing it. This chapter focuses on setting up encryption, authentication, and access control. It discusses the following security-related Messaging Server topics:

- **User ID and password login:** requiring users to enter their user IDs and passwords to log in to IMAP, POP, HTTP, or SMTP, and the use of SMTP password login to transmit sender authentication to message recipients.
- **Encryption and authentication:** setting up your server to use the TLS and SSL protocols to encrypt communication and authenticate clients.
- **Administrator access control:** using the access-control facilities of Netscape Console to delegate access to a Messaging Server and some of its individual tasks.
- **TCP client access control:** using filtering techniques to control which clients can connect to your server's POP, IMAP, HTTP, and SMTP services.

Not all security and access issues related to Messaging Server are treated in this chapter. Security topics that are discussed elsewhere include the following:

- **Physical security:** Without provisions for keeping server machines physically secure, software security can be meaningless.
- **Encrypted messages (S/MIME):** With Secure Multipurpose Internet Mail Extensions, senders can encrypt messages prior to sending them, and recipients can store the encrypted messages after receipt, decrypting them only to read them. Using S/MIME requires no special Messaging Server configuration or tasks; it is strictly a client action. See your client documentation for information on setting it up. Note that the Messenger Express client interface does not support encryption of email messages.
- **Message-store access:** You can define a set of message-store administrators for the Messaging Server. These administrators can view and monitor mailboxes and can control access to them. For details, see Chapter 10, "Managing the Message Store."

- **End-user account configuration:** End-user account information is primarily maintained by using the Delegated Administrator product. For more information, see the Delegated Administrator documentation. You can also manage end-user accounts by using the Console interface. For more information, see Chapter 3, “Managing Mail Users and Mailing Lists.”
- **Filtering unsolicited bulk email (UBE):** See Chapter 9, “Mail Filtering and Access Control.”

iPlanet has produced a large number of documents that cover a variety of security topics. For additional background on the topics mentioned here and for other security-related information, see the iPlanet documentation web site at http://docs.iplanet.com.

About HTTP Security

Messaging Server supports the same security features for the HTTP protocol as it does for the IMAP protocol—both user ID/password authentication and client certificate authentication are supported. There are some differences, however, in how the protocols handle network connections between client and server.

When a POP, IMAP, or SMTP client logs in to Messaging Server, a connection is made and a session is established. The connection lasts for the duration of the session; that is, from login to logout. When establishing a new connection, the client must reauthenticate to the server.

When an HTTP client logs in to Messaging Server, the server provides a unique session ID to the client. The client uses the session ID to establish multiple connections during a session. The HTTP client need not reauthenticate for each connection; the client need only reauthenticate if the session is dropped and the client wants to establish a new session. (If an HTTP session remains idle for a specified time period, the server will automatically drop the HTTP session and the client is logged out; the default time period is 2 hours.)

HTTP session connections remain secure because:

- The session IDs are bound to a specific IP address, so they cannot be used by another host.
- Each session ID has a timeout value associated with it; if the session ID is not used for a specified time period, the session ID becomes invalid.
- The server keeps a database of all open session IDs, so a client cannot forge an ID.

- The session ID is stored in the URL, but not in any cookie files.

For information about specifying configuration parameters for improved connection performance, see Chapter 4, “Configuring POP, IMAP, and HTTP Services.”

Configuring Authentication Mechanisms

An authentication mechanism is a particular method for a client to prove its identity to a server. Messaging Server supports authentication methods defined by the Simple Authentication and Security Layer (SASL) protocol and it supports certificate-based authentication. The SASL mechanisms are described in this section. For more information about certificate-based authentication, see “Configuring Encryption and Certificate-Based Authentication,” on page 288.

Messaging Server supports the following SASL authentication methods for password-based authentication.

- **PLAIN** - This mechanism passes the user’s plaintext password over the network, where it is susceptible to eavesdropping.

  Note that SSL can be used to alleviate the eavesdropping problem. For more information, see “Configuring Encryption and Certificate-Based Authentication,” on page 288.

- **DIGEST-MD5** - A challenge / response authentication mechanism based upon the HTTP-digest authentication defined in RFC 2831. (DIGEST-MD5 is not yet supported by Messaging Multiplexor.)

- **CRAM-MD5** - A challenge / response authentication mechanism similar to APOP, but suitable for use with other protocols as well. Defined in RFC 2195.

- **APOP** - A challenge / response authentication mechanism that can be used only with the POP3 protocol. Defined in RFC 1939.

With a challenge / response authentication mechanism, the server sends a challenge string to the client. The client responds with a hash of that challenge and the user's password. If the client's response matches the server's own hash, the user is authenticated. The hash isn't reversible, so the user's password isn't compromised when sent over the network.

NOTE	The POP, IMAP, and SMTP services support all SASL mechanisms. The HTTP service supports only the plaintext password mechanism.

Configuring Access to Plaintext Passwords

To work, the CRAM-MD5, DIGEST-MD5, or APOP SASL authentication methods require access to the users' plaintext passwords. You need to perform the following steps:

1. Configure Directory Server to store passwords in cleartext.
2. Configure Messaging Server so that it knows Directory Server is using cleartext passwords.

Configure Directory Server

To enable CRAM-MD5, DIGEST-MD5, or APOP mechanisms, you must configure the Directory Server to store passwords in cleartext as follows:

1. In Console, open the Directory Server you want to configure.
2. Click the Configuration tab.
3. Open Database in the left pane.
4. Click Passwords in the right pane.
5. From the Password encryption drop-down list, choose "cleartext".

Configure Messaging Server

You can now configure Messaging Server so that it knows the Directory Server is able to retrieve cleartext passwords. This makes it safe for Messaging Server to advertise APOP, CRAM-MD5, and DIGEST-MD5:

```
configutil -o sasl.default.ldap.has_plain_passwords -v 1
```

You can disable SASL mechanisms by setting the value to 0 or null ("").

NOTE Existing users cannot use APOP, CRAM-MD5, or DIGESt-MD5 until their password is reset or migrated (see Transitioning Users).

Transitioning Users

You can use `configutil` to specify information about transitioning users. For example, if a user password changes or if a client attempts to authenticate with a mechanism for which they do not have a proper entry.

```
configutil -o sasl.default.transition_criteria -v value
```

For value, you can specify one of the following:

- `CHANGE` - If a user password changes, the server transitions to plaintext. This is the default.
- `CLIENT` - If a client attempts to use a mechanism for which they do not have a proper entry, the server asks the client to authenticate using a plaintext password. The server then creates the desired mechanism entry using the same password value.
- `PLAIN` - If a client uses a plaintext password, the server transitions to plaintext.

To successfully transition users, you must set up ACIs in the Directory Server that allow Messaging Server write access to the user password attribute. To do this, perform the following steps:

1. In Console, open the Directory Server you want to configure.
2. Click the Directory tab.
3. Select the base suffix for the user/group tree.
4. From the Object menu, select Access Permissions.
5. Select (double click) the ACI for "Messaging Server End User Administrator Write Access Rights".
6. Click ACI Attributes.
7. Add the `userpassword` attribute to the list of existing attributes.
8. Click OK.

User Password Login

Requiring password submission on the part of users logging into Messaging Server to send or receive mail is a first line of defense against unauthorized access. Messaging Server supports password-based login for its IMAP, POP, HTTP, and SMTP services.

IMAP, POP, and HTTP Password Login

By default, users must submit a password to retrieve their messages from Messaging Server. You enable or disable password login separately for POP, IMAP, and HTTP services. For more information about password login for POP IMAP, and HTTP Services, see “Password-Based Login,” on page 68.

User passwords can be transmitted from the user’s client software to your server as cleartext or in encrypted form (with the exception of POP). If both the client and your server are configured to enable SSL and both support encryption of the required strength (as explained in “Enabling SSL and Selecting Ciphers,” on page 294), encryption occurs.

User IDs and passwords are stored in your installation’s LDAP user directory. Password security criteria, such as minimum length, are determined by directory policy requirements; they are not part of Messaging Server administration.

Certificate-based login is an alternative to password-based login. It is discussed in this chapter along with the rest of SSL; see “Setting Up Certificate-Based Login,” on page 296.

Challenge / response SASL mechanisms are another alternative to plaintext password login.

SMTP Password Login

By default, users need not submit a password when they connect to the SMTP service of Messaging Server to send a message. You can, however, enable password login to SMTP in order to enable authenticated SMTP.

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 travelling (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.

For instructions on enabling SMTP password login (and thus Authenticated SMTP), see “SMTP Authentication and SASL” on page 174.

You can use Authenticated SMTP with or without SSL encryption.

Configuring Encryption and Certificate-Based Authentication

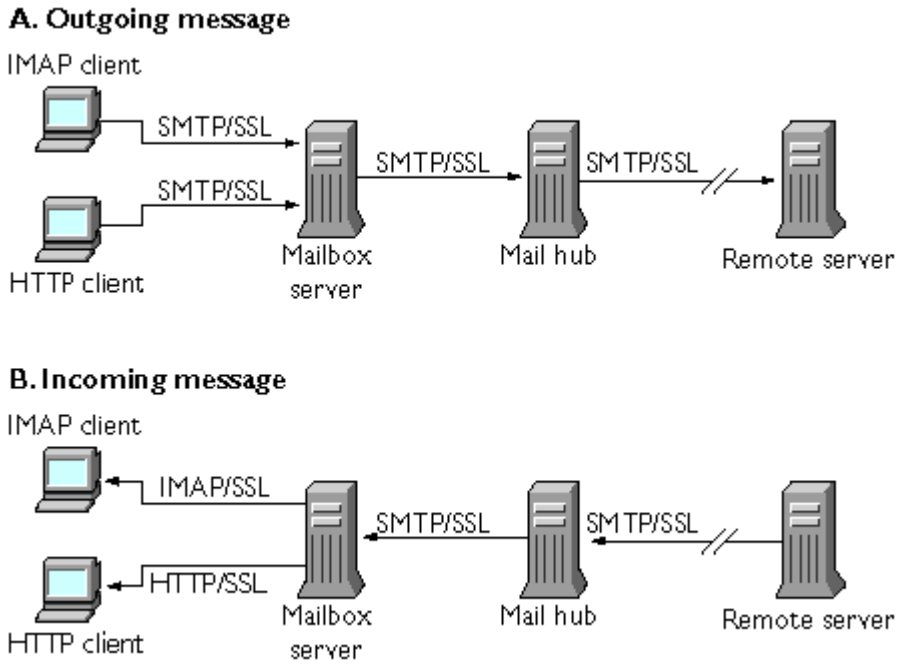
iPlanet Messaging Server uses the Transport Layer Security (TLS) protocol, otherwise known as the Secure Sockets Layer (SSL) protocol, for encrypted communications and for certificate-based authentication of clients and servers. iPlanet Messaging Server supports SSL versions 3.0 and 3.1. TLS is fully compatible with SSL and includes all necessary SSL functionality.

For background information on SSL, see the *Introduction to SSL* (reproduced as an appendix to *Managing Servers with Netscape Console*). SSL is based on the concepts of public-key cryptography, described in *Introduction to Public-Key Cryptography* (also reproduced as an appendix to *Managing Servers with Netscape Console*).

If transmission of messages between a Messaging Server and its clients and between the server and other servers is encrypted, there is little chance for eavesdropping on the communications. If connecting clients are authenticated, there is little chance for intruders to impersonate (spoof) them.

SSL functions as a protocol layer beneath the application layers of IMAP4, HTTP, and SMTP. SMTP and SMTP / SSL use the same port; HTTP and HTTP / SSL require different ports; IMAP and IMAP / SSL can use the same port or different ports. SSL acts at a specific stage of message communication, as shown in Figure 11-1, for both outgoing and incoming messages.

Figure 11-1 Encrypted Communications with Messaging Server



SSL provides hop-to-hop encryption, but the message is not encrypted on each intermediate server. Client support for S/MIME is required to get end-to-end encryption.

NOTE To enable encryption for outgoing messages, you must modify the channel definition to include the `tls` channel keywords, such as `maytls`, `musttls`, and so on. For more information, see the *Messaging Server Reference Manual*.

Keep in mind that the extra overhead in setting up an SSL connection can put a performance burden on the server. In designing your messaging installation and in analyzing performance, you may need to balance security needs against server capacity.

NOTE Because all iPlanet servers support SSL, and the interface for enabling and configuring SSL through Console is nearly identical across all servers, several of the tasks described in this section are documented more completely in the SSL chapter of *Managing Servers with Netscape Console*. For those tasks, this chapter gives summary information only.

Obtaining Certificates

Whether you use SSL for encryption or for authentication, you need to obtain a server certificate for your Messaging Server. The certificate identifies your server to clients and to other servers.

Managing Internal and External Modules

A server certificate establishes the ownership and validity of a key pair, the numbers used to encrypt and decrypt data. Your server's certificate and key pair represent your server's identity. They are stored in a certificate database that can be either internal to the server or on an external, removable hardware card (smartcard).

iPlanet servers access a key and certificate database using a module conforming to the Public-Key Cryptography System (PKCS) #11 API. The PKCS #11 module for a given hardware device is usually obtained from its supplier and must be installed into the Messaging Server before the Messaging Server can use that device. The preinstalled "Netscape Internal PKCS # 11 Module" supports a single internal software token that uses the certificate database that is internal to the server.

Setting up the server for a certificate involves creating a database for the certificate and its keys and installing a PKCS #11 module. If you do not use an external hardware token, you create an internal database on your server, and you use the internal, default module that is part of Messaging Server. If you do use an external token, you connect a hardware smartcard reader and install its PKCS #11 module.

You can manage PKCS #11 modules, whether internal or external, through Console. To install a PKCS #11 module:

1. Connect a hardware card reader to the Messaging Server host machine and install drivers.
2. Use the PKCS #11 Management interface in Console to install the PKCS #11 module for the installed driver.

(For more complete instructions, see the chapter on SSL in *Managing Servers with Netscape Console*.)

Installing Hardware Encryption Accelerators. If you use SSL for encryption, you may be able to improve server performance in encrypting and decrypting messages by installing a hardware encryption accelerator. An encryption accelerator typically consists of a hardware board, installed permanently in your server machine, plus a software driver. iPlanet Messaging Server supports accelerator modules that follow the PKCS #11 API. (They are essentially hardware tokens that do not store their own keys; they use the internal database for that.) You install an accelerator by first installing the hardware and drivers as specified by the manufacturer, and then completing the installation—as with hardware certificate tokens—by installing the PKCS #11 module.

Requesting a Server Certificate

You request a server certificate by opening your server in iPlanet Console and running the Certificate Setup Wizard. You can access the Wizard from the Console menu or from the Messaging Server Encryption tab. Using the Wizard, you perform the following tasks:

1. Generate a certificate request.
2. Send the request by email to the certificate authority (CA) that is to issue the certificate.

When the email response from the CA arrives, you save it as a text file and install it using the Certificate Setup Wizard.

(For more complete instructions, see the chapter on SSL in *Managing Servers with Netscape Console*.)

Installing the Certificate

Installing is a separate process from requesting. Once the email response to your request for a certificate has arrived from the CA and been saved as a text file, run the Certificate Setup Wizard once more to install the file as a certificate:

1. Specify that you are installing a certificate that you have already obtained.
2. Paste the text of your certificate into a field when prompted to do so.

(For more complete instructions, see the chapter on SSL in *Managing Servers with Netscape Console*.)

NOTE This is also the process you follow to install a CA certificate (described next), which your server uses to determine whether to trust the certificates presented by clients.

Installing Certificates of Trusted CAs

You also use the Certificate Setup Wizard to install the certificates of certificate authorities. A CA certificate validates the identity of the CA itself. Your server uses these CA certificates in the process of authenticating clients and other servers.

If, for example, you set up your enterprise for certificate-based client authentication in addition to password-based authentication (see "Setting Up Certificate-Based Login" on page 157), you need to install the CA certificates of all CAs that are trusted to issue the certificates that your clients may present. These CAs may be internal to your organization or they may be external, representing commercial or governmental authorities or other enterprises. (For more details on the use of CA certificates for authentication, see *Introduction to Cryptography* in *Managing Servers with Netscape Console*.)

When installed, Messaging Server initially contains CA certificates for several commercial CAs. If you need to add other commercial CAs or if your enterprise is developing its own CA for internal use (using iPlanet Certificate Server), you need to obtain and install additional CA certificates.

NOTE The CA certificates automatically provided with Messaging Server are not initially marked as trusted for client certificates. You need to edit the trust settings if you want to trust client certificates issued by these CAs. For instructions, see "Managing Certificates and Trusted CAs" on page 153.

To request and install a new CA certificate, you:

1. Contact the certificate authority (possibly through the Web or by email) and download its CA certificate.
2. Save the received text of the certificate as a text file.
3. Use the Certificate Setup Wizard, as described in the previous section, to install the certificate.

For more complete instructions, see the chapter on SSL in *Managing Servers with Netscape Console*.

Managing Certificates and Trusted CAs

Your server can have any number of certificates of trusted CAs that it uses for authentication of clients.

You can view, edit the trust settings of, or delete any of the certificates installed in your Messaging Server by opening your server in Console and choosing the Certificate Management Command in the Console menu. For instructions, see the chapter on SSL in *Managing Servers with Netscape Console*.

Creating a Password File

On any iPlanet server, when you use the Certificate Setup Wizard to request a certificate, the wizard creates a key pair to be stored in either the internal module's database or in an external database (on a smartcard). The wizard then prompts you for a password, which it uses to encrypt the private key. Only that same password can later be used to decrypt the key. The wizard does not retain the password nor store it anywhere.

On most iPlanet servers for which SSL is enabled, the administrator is prompted at startup to supply the password required to decrypt the key pair. On Messaging Server, however, to alleviate the inconvenience of having to enter the password multiple times (it is needed by at least three server processes), and to facilitate unattended server restarts, the password is read from a password file.

The password file is named `sslpassword.conf` and is in the directory *server-instance*`/config/`. Entries in the file are individual lines with the format

`moduleName:password`

where *moduleName* is the name of the (internal or external) PKCS #11 module to be used, and *password* is the password that decrypts that module's key pair. The password is stored as clear (unencrypted) text.

Messaging Server provides a default version of the password file, with the following single entry (for the internal module and default password):

```
Internal (Software) Token:netscape!
```

If you specify anything but the default password when you install an internal certificate, you need to edit the above line of the password file to reflect the password you specified. If you install an external module, you need to add a new line to the file, containing the module name and the password you specified for it.

CAUTION Because the administrator is not prompted for the module password at server startup, it is especially important that you ensure proper administrator access control to the server and proper physical security of the server host machine and its backups.

Enabling SSL and Selecting Ciphers

You can use Console to enable SSL and to select the set of encryption ciphers that Messaging Server can use in its encrypted communications with clients.

About Ciphers

A *cipher* is the algorithm used to encrypt and decrypt data in the encryption process. Some ciphers are stronger than others, meaning that a message they have scrambled is more difficult for an unauthorized person to unscramble.

A cipher operates on data by applying a key—a long number—to the data. Generally, the longer the key the cipher uses during encryption, the harder it is to decrypt the data without the proper decryption key.

When a client initiates an SSL connection with a Messaging Server, the client lets the server know what ciphers and key lengths it prefers to use for encryption. In any encrypted communication, both parties must use the same ciphers. Because there are a number of cipher-and-key combinations in common use, a server should be flexible in its support for encryption. iPlanet Messaging Server can support up to 6 combinations of cipher and key length.

Table 6.1 lists the ciphers that Messaging Server supports for use with SSL 3.0. The table summarizes information that is available in more detail in the *Introduction to SSL* section of *Managing Servers with Netscape Console*.

Table 11-1 SSL Ciphers for Messaging Server

Cipher	Description
RC4 with 128-bit encryption and MD5 message authentication	The fastest encryption cipher (by RSA) and a very high-strength combination of cipher and encryption key.
Triple DES with 168-bit encryption and SHA message authentication	A slower encryption cipher (a U.S. government-standard) but the highest-strength combination of cipher and encryption key.

Table 11-1 SSL Ciphers for Messaging Server

Cipher	Description
DES with 56-bit encryption and SHA message authentication	A slower encryption cipher (a U.S. government-standard) and a moderate-strength combination of cipher and encryption key.
RC4 with 40-bit encryption and MD5 message authentication	The fastest encryption cipher (by RSA) and a lower-strength combination of cipher and encryption key.
RC2 with 40-bit encryption and MD5 message authentication	A slower encryption cipher (by RSA) and a lower-strength combination of cipher and encryption key.
No encryption, only MD5 message authentication	No encryption; use of a message digest for authentication alone.

Unless you have a compelling reason for not using a specific cipher, you should support them all. However, note that export laws restrict the use of certain encryption ciphers in certain countries. Also, some client software produced before the relaxation of United States Export Control laws cannot use the higher strength encryption. Be aware that while the 40-bit ciphers might hinder the casual eavesdropper, they are not secure and therefore will not stop a motivated attack.

Console. To enable SSL and select encryption ciphers by using the Console, follow these steps:

1. In Console, open the Messaging Server whose cipher settings you want to modify.
2. Click the Configuration tab in the left pane and select the Services folder.
3. Click the Encryption tab in the right pane.
4. Check the Enable SSL box to enable SSL on your server.
5. Check the RSA box if you want to enable RSA ciphers.
6. From the Token drop-down list, choose the token you want to use.
7. From the Certificate drop-down list, choose the certificate you want to use.
8. Click Cipher Preferences to open the list of available ciphers.
9. Click the boxes to select the encryption cipher or ciphers that you want your server to support.

To disable SSL completely, deselect the Enable SSL box.

NOTE	To enable SSL encryption for outgoing messages, you must modify the channel definition to include the `tls` channel keywords, such as `maytls`, `musttls`, and so on. For more information, see the *Messaging Server Reference Manual*.

Command Line. You can also enable SSL and select ciphers at the command line as follows:

To enable or disable SSL:

```
configutil -o nsserversecurity -v [ on | off ]
```

To enable or disable RSA ciphers:

```
configutil -o encryption.rsa.nssslactivation -v [ on | off ]
```

To specify a token:

```
configutil -o encryption.rsa.nsssltoken -v tokenname
```

To specify a certificate:

```
configutil -o encryption.rsa.nssslpersonalityssl -v certname
```

Note that if you enable RSA ciphers, you must also specify a token and a certificate.

To choose a cipher preference:

```
configutil -o encryption.nsssl3ciphers -v cipherlist
```

where *cipherlist* is a comma-separated list of ciphers.

Setting Up Certificate-Based Login

In addition to password-based authentication, iPlanet servers support authentication of users through examination of their digital certificates. In certificate-based authentication, the client establishes an SSL session with the server and submits the user's certificate to the server. The server then evaluates whether the submitted certificate is genuine. If the certificate is validated, the user is considered authenticated.

To set up your Messaging Server for certificate-based login:

1. Obtain a server certificate for your server. (For details, see “Obtaining Certificates,” on page 290.)
2. Run the Certificate Setup Wizard to install the certificates of any trusted certificate authorities that will issue certificates to the users your server will authenticate. (For details, see “Installing Certificates of Trusted CAs,” on page 292.)

   Note that as long as there is at least one trusted CA in the server’s database, the server requests a client certificate from each connecting client.
3. Turn on SSL. (For details, see “Enabling SSL and Selecting Ciphers,” on page 294.)
4. (Optional) Edit your server’s `certmap.conf` file so that the server appropriately searches the LDAP user directory based on information in the submitted certificates.

   Editing the `certmap.conf` file is not necessary if the email address in your users’ certificates matches the email address in your users’ directory entries, and you do not need to optimize searches or validate the submitted certificate against a certificate in the user entry.

   For details of the format of `certmap.conf` and the changes you can make, see the SSL chapter of *Managing Servers with Netscape Console*.

Once you have taken these steps, when a client establishes an SSL session so that the user can log in to IMAP or HTTP, the Messaging Server requests the user’s certificate from the client. If the certificate submitted by the client has been issued by a CA that the server has established as trusted, and if the identity in the certificate matches an entry in the user directory, the user is authenticated and access is granted (depending on access-control rules governing that user).

There is no need to disallow password-based login to enable certificate-based login. If password-based login is allowed (which is the default state), and if you have performed the tasks described in this section, both password-based and certificate-based login are supported. In that case, if the client establishes an SSL session and supplies a certificate, certificate-based login is used. If the client does not use SSL or does not supply a certificate, the server requests a password.

For more details on setting up your entire installation of iPlanet servers and clients to use certificate-based authentication, see the *Single Sign-On Deployment Guide*.

Configuring Administrator Access to Messaging Server

This section describes how to control the ways in which server administrators can gain access to Messaging Server. Administrative access to a given Messaging Server and to specific Messaging Server tasks occurs within the context of delegated server administration.

Delegated server administration is a feature of most iPlanet servers; it refers to the capability of an administrator to provide other administrators with selective access to individual servers and server features. This chapter briefly summarizes delegated server tasks. For more detailed information, see the chapter on delegating server administration in *Managing Servers with Netscape Console.* You should also read the section, “Provisioning Messaging Server Administrators” in the *Messaging Server Provisioning Guide*. The Provisioning Guide describes server Administrators, administrators who can configure the messaging server, and iDA Administrators, administrators who can add, modify and delete users and groups in the system

Hierarchy of Delegated Administration

When you install the first iPlanet server on your network, the installation program automatically creates a group in the LDAP user directory called the Configuration Administrators group. By default, the members of the Configuration Administrators group have unrestricted access to all hosts and servers on your network.

The Configuration Administrators group is at the top of an access hierarchy, such as the following, that you can create to implement delegated administration for Messaging Server:

1. **Configuration administrator.** The “super user” for the network of iPlanet servers. Has complete access to all resources.
2. **Server administrator.** A domain administrator might create groups to administer each type of server. For example, a Messaging Administrators group might be created to administer all Messaging Servers in an administrative domain or across the whole network. Members of that group have access to all Messaging Servers (but no other servers) in that administrative domain.

3. **Task administrator.** Finally, any of the above administrators might create a group, or designate an individual user, with restricted access to a single Messaging Server or a set of Messaging Servers. Such a task administrator is permitted to perform only specific, limited server tasks (such as starting or stopping the server only, or accessing logs of a given service).

Console provides convenient interfaces that allow an administrator to perform the following tasks:

- Grant a group or an individual access to a specific Messaging Server, as described in "Providing Access to the Server as a Whole" (next).
- Restrict that access to specific tasks on a specific Messaging Server, as described in "Restricting Access to Specific Tasks," on page 299.

Providing Access to the Server as a Whole

To give a user or group permission to access a given instance of Messaging Server, you:

1. Log in to Console as an administrator with access to the Messaging Server you want to provide access to.
2. Select that server in the Console window.

   From the Console menu, choose Object, then choose Set Access Permissions.
3. Add or edit the list of users and groups with access to the server.

(For more complete instructions, see the chapter on delegating server administration in *Managing Servers with Netscape Console*.)

Once you have set up the list of individuals and groups that have access to the particular Messaging Server, you can then use ACIs, as described next, to delegate specific server tasks to specific people or groups on that list.

Restricting Access to Specific Tasks

An administrator typically connects to a server to perform one or more administrative tasks. Common administrative tasks are listed in the Messaging Server Tasks form in Console.

By default, access to a particular Messaging Server means access to all of its tasks. However, each task in the Task form can have an attached set of access-control instructions (ACIs). The server consults those ACIs before giving a connected user (who must already be a user with access permissions to the server as a whole) access to any of the tasks. In fact, the server displays in the Tasks form only those tasks to which the user has permission.

If you have access to a Messaging Server, you can create or edit ACIs on any of the tasks (that is, on any of the tasks to which you have access), and thus restrict the access that other users or groups can have to them.

To restrict the task access that a connected user or group can have, you:

1. Log in to the Console as an administrator with access to the Messaging Server you want to provide restricted access to.
2. Open the server and select a task in the server's Tasks form by clicking on the Task text.
3. From the Edit menu, choose Set Access Permissions, and add or edit the list of access rules to give a user or group the kind of access you want them to have.
4. Repeat the process for other tasks, as appropriate.

   (For more complete instructions, see the chapter on delegating server administration in *Managing Servers with Netscape Console.*)

ACIs and how to create them are described more fully in the chapter on delegating server administration in *Managing Servers with Netscape Console.*

Configuring Client Access to POP, IMAP, and HTTP Services

Messaging Server supports sophisticated access control on a service-by-service basis for its IMAP, POP, and HTTP services so that you can exercise far-ranging and fine-grained control over which clients can gain access to your server.

If you are managing messaging services for a large enterprise or an Internet service provider, these capabilities can help you to exclude spammers and DNS spoofers from your system and improve the general security of your network. For control of unsolicited bulk email specifically, see also Chapter 9, “Mail Filtering and Access Control.”

NOTE If controlling access by IP address is *not* an important issue for your enterprise, you do not have to create any of the filters described in this section. If minimal access control is all you need, see the section “Mostly Allowing,” on page 307 for instructions on setting it up.

How Client Access Filters Work

The Messaging Server access-control facility is a program that listens at the same port as the TCP daemon it serves; it uses access filters to verify client identity, and it gives the client access to the daemon if the client passes the filtering process.

As part of its processing, the Messaging Server TCP client access-control system performs (when necessary) the following analyses of the socket end-point addresses:

- Reverse DNS lookups of both end points (to perform name-based access control)
- Forward DNS lookups of both end points (to detect DNS spoofing)
- `Identd` callback (to check that the user on the client end is known to the client host)

The system compares this information against access-control statements called *filters* to decide whether to grant or deny access. For each service, separate sets of Allow filters and Deny filters control access. Allow filters explicitly grant access; Deny filters explicitly forbid access.

When a client requests access to a service, the access-control system compares the client's address or name information to each of that service's filters—in order—using these criteria:

- The search stops at the first match. Because Allow filters are processed before Deny filters, Allow filters take precedence.
- Access is granted if the client information matches an Allow filter for that service.
- Access is denied if the client information matches a Deny filter for that service.
- If no match with any Allow or Deny filter occurs, access is granted—except in the case where there are Allow filters but no Deny filters, in which case lack of a match means that access is denied.

The filter syntax described here is flexible enough that you should be able to implement many different kinds of access-control policies in a simple and straightforward manner. You can use both Allow filters and Deny filters in any combination, even though you can probably implement most policies by using almost exclusively Allows or almost exclusively Denies.

The following sections describe filter syntax in detail and give usage examples. The section "Creating Access Filters for Services," on page 309 gives the procedure for creating access filters.

Filter Syntax

Filter statements contain both service information and client information. The service information can include the name of the service, names of hosts, and addresses of hosts. The client information can include host names, host addresses, and user names. Both the server and client information can include wildcard names or patterns.

The very simplest form of a filter is:

```
service: hostSpec
```

where *service* is the name of the service (such as `smtp`, `pop`, `imap`, or `http`) and *hostSpec* is the host name, IP address, or wildcard name or pattern that represents the client requesting access. When a filter is processed, if the client seeking access matches *client*, access is either allowed or denied (depending on which type of filter this is) to the service specified by *service*. Here are some examples:

```
imap: roberts.newyork.siroe.com
```

```
pop: ALL
```

```
http: ALL
```

If these are Allow filters, the first one grants the host `roberts.newyork.siroe.com` access to the IMAP service, and the second and third grant all clients access to the POP and HTTP services, respectively. If they are Deny filters, they deny those clients access to those services. (For descriptions of wildcard names such as `ALL`, see “Wildcard Names,” on page 304.)

Either the server or the client information in a filter can be somewhat more complex than this, in which case the filter has the more general form of:

`serviceSpec``: `*`clientSpec`*

where *serviceSpec* can be either *service* or *service@hostSpec*, and *clientSpec* can be either *hostSpec* or *user@hostSpec*. *user* is the user name (or a wildcard name) associated with the client host seeking access. Here are two examples:

```
pop@mailServer1.siroe.com: ALL
```

```
imap: srashad@xyz.europe.siroe.com
```

If these are Deny filters, the first filter denies all clients access to the SMTP service on the host `mailServer1.siroe.com`. The second filter denies the user `srashad` at the host `xyz.europe.siroe.com` access to the IMAP service. (For more information on when to use these expanded server and client specifications, see “Server-Host Specification” on page 305 and “Client User-Name Specification” on page 306.)

Finally, at its most general, a filter has the form:

`serviceList``: `*`clientList`*

where *serviceList* consists of one or more *serviceSpec* entries, and *clientList* consists of one or more *clientSpec* entries. Individual entries within *serviceList* and *clientList* are separated by blanks and / or commas.

In this case, when a filter is processed, if the client seeking access matches any of the *clientSpec* entries in *clientList*, then access is either allowed or denied (depending on which type of filter this is) to all the services specified in *serviceList*. Here is an example:

```
pop, imap, http: .europe.siroe.com .newyork.siroe.com
```

If this is an Allow filter, it grants access to POP, IMAP, and HTTP services to all clients in either of the domains `europe.siroe.com` and `newyork.siroe.com`. For information on using a leading dot or other pattern to specify domains or subnet, see “Wildcard Patterns,” on page 304.

Wildcard Names

You can use the following wildcard names to represent service names, host names or addresses, or user names:

Table 11-2 Wildcard Names

Wildcard Name	Explanation
ALL	The universal wildcard. Matches all names.
LOCAL	Matches any local host (one whose name does not contain a dot character). However, if your installation uses only canonical names, even local host names will contain dots and thus will not match this wildcard.
UNKNOWN	Matches any user whose name is unknown, or any host whose name or address is unknown. Use this wildcard name carefully: Host names may be unavailable due to temporary DNS server problems—in which case all filters that use `UNKNOWN` will match all client hosts. A network address is unavailable when the software cannot identify the type of network it is communicating with—in which case all filters that use `UNKNOWN` will match all client hosts on that network.
KNOWN	Matches any user whose name is known, or any host whose name *and* address are known. Use this wildcard name carefully: Host names may be unavailable due to temporary DNS server problems—in which case all filters that use `KNOWN` will fail for all client hosts. A network address is unavailable when the software cannot identify the type of network it is communicating with—in which case all filters that use `KNOWN` will fail for all client hosts on that network.
DNSSPOOFER	Matches any host whose DNS name does not match its own IP address.

Wildcard Patterns

You can use the following patterns in service or client addresses:

- A string that begins with a dot character (.). A host name is matched if the last components of its name match the specified pattern. For example, the wildcard pattern `.siroe.com` matches all hosts in the domain `siroe.com`.
- A string that ends with a dot character (.). A host address is matched if its first numeric fields match the specified pattern. For example, the wildcard pattern `123.45.` matches the address of any host in the subnet `123.45.0.0`.
- A string of the form `n.n.n.n/m.m.m.m`. This wildcard pattern is interpreted as a *net/mask* pair. A host address is matched if *net* is equal to the bitwise AND of the address and *mask*. For example, the pattern `123.45.67.0/255.255.255.128` matches every address in the range `123.45.67.0` through `123.45.67.127`.

EXCEPT Operator

The access-control system supports a single operator. You can use the `EXCEPT` operator to create exceptions to matching names or patterns when you have multiple entries in either *serviceList* or *clientList*. For example, the expression:

list1 `EXCEPT` *list2*

means that anything that matches *list1* is matched, *unless* it also matches *list2*.

Here is an example:

```
ALL: ALL EXCEPT isserver.siroe.com
```

If this were a Deny filter, it would deny access to all services to all clients except those on the host machine `isserver.siroe.com`.

`EXCEPT` clauses can be nested. The expression:

list1 `EXCEPT` *list2* `EXCEPT` *list3*

is evaluated as if it were:

list1 `EXCEPT (`*list2* `EXCEPT` *list3*`)`

Server-Host Specification

You can further identify the specific service being requested in a filter by including server host name or address information in the *serviceSpec* entry. In that case the entry has the form:

service@hostSpec

You might want to use this feature when your Messaging Server host machine is set up for multiple internet addresses with different internet host names. If you are a service provider, you can use this facility to host multiple domains, with different access-control rules, on a single server instance.

Client User-Name Specification

For client host machines that support the `identd` service as described in RFC 1413, you can further identify the specific client requesting service by including the client's user name in the *clientSpec* entry in a filter. In that case the entry has the form:

user@hostSpec

where *user* is the user name as returned by the client's `identd` service (or a wildcard name).

Specifying client user names in a filter can be useful, but keep these caveats in mind:

- The `identd` service is not authentication; the client user name it returns cannot be trusted if the client system has been compromised. In general, do not use specific user names; use only the wildcard names `ALL`, `KNOWN`, or `UNKNOWN`.
- User-name lookups take time; performing lookups on all users may slow access by clients that do not support `identd`. Selective user-name lookups can alleviate this problem. For example, a rule like:

  serviceList`: @xyzcorp.com ALL@ALL`

  would match users in the domain `xyzcorp.com` without doing user-name lookups, but it would perform user-name lookups with all other systems.

The user-name lookup capability can in some cases help you guard against attack from unauthorized users on the client's host. It is possible in some TCP/IP implementations, for example, for intruders to use `rsh` (remote shell service) to impersonate trusted client hosts. If the client host supports the `ident` service, you can use user-name lookups to detect such attacks. For an example and discussion, see "Allowing Only Identified Users," on page 307.

Filter Examples

The examples in this section show a variety of approaches to controlling access. In studying the examples, keep in mind that Allow filters are processed before Deny filters, the search terminates when a match is found, and access is granted when no match is found at all.

The examples listed here use host and domain names rather than IP addresses. Remember that you can include address and netmask information in filters, which can improve reliability in the case of name-service failure.

Mostly Denying

In this case, access is denied by default. Only explicitly authorized hosts are permitted access.

The default policy (no access) is implemented with a single, trivial deny file:

```
ALL: ALL
```

This filter denies all service to all clients that have not been explicitly granted access by an Allow filter. The Allow filters, then, might be something like these:

```
ALL: LOCAL @netgroup1
```

```
ALL: .siroe.com EXCEPT externalserver.siroe.com
```

The first rule permits access from all hosts in the local domain (that is, all hosts with no dot in their host name) and from members of the group `netgroup1`. The second rule uses a leading-dot wildcard pattern to permit access from all hosts in the `siroe.com` domain, with the exception of the host `externalserver.siroe.com`.

Mostly Allowing

In this case, access is granted by default. Only explicitly specified hosts are denied access.

The default policy (access granted) makes Allow filters unnecessary. The unwanted clients are listed explicitly in Deny filters such as these:

```
ALL: externalserver.siroe1.com, .siroe.asia.com
```

```
ALL EXCEPT pop: contractor.siroe1.com, .siroe.com
```

The first filter denies all services to a particular host and to a specific domain. The second filter permits nothing but POP access from a particular host and from a specific domain.

Allowing Only Identified Users

You can use the following Deny filter to exclude all users but those known to a client host's `identd` service:

```
ALL: UNKWOWN@ALL
```

The filter denies all services to all unknown users from all domains.

You could write a more specific Deny filter, with a *clientSpec* entry of UNKNOWN@*host*. When it receives a request from *host*, the access-control system then uses the ident service on *host* to find out whether that host actually sent the request and what the user name of the requestor is. If the host responds that the sending user is unknown, that may be evidence of an attack. (However, note also that, if the client's host does not support the identd service, all requestors will match the UNKNOWN@*host* filter.)

Employing user-name lookups in Allow filters is less trustworthy. Suppose you write an Allow filter with a *clientSpec* entry of KNOWN@*host*. Because an intruder can spoof both the client connection and the ident lookup, a match with the KNOWN@*host* filter is not strong evidence of absence of spoofing. Furthermore, if the client system has been compromised (as noted earlier), ident may return false information.

For more information, see "Client User-Name Specification" on page 306.

Denying Access to Spoofed Domains

You can use the DNSSPOOFER wildcard name in a filter to detect host-name spoofing. When you specify DNSSPOOFER, the access-control system performs forward or reverse DNS lookups to verify that the client's presented host name matches its actual IP address. Here is an example for a Deny filter:

```
ALL: DNSSPOOFER
```

This filter denies all services to all remote hosts whose IP addresses don't match their DNS host names.

Controlling Access to Virtual Domains

If your messaging installation uses virtual domains, in which a single server instance is associated with multiple IP addresses and domain names, you can control access to each virtual domain through a combination of Allow and Deny filters. For example, you can use Allow filters like:

```
ALL@msgServer.siroe1.com: @.siroe1.com
ALL@msgServer.siroe2.com: @.siroe2.com
...
```

coupled with a Deny filter like:

```
ALL: ALL
```

Each Allow filter permits only hosts within domain*N* to connect to the service whose IP address corresponds to msgServer.siroe*N*.com. All other connections are denied.

Denying an Individual User

If you must deny access to an especially notorious individual user, the most general Deny filter you can apply is the following:

```
ALL: badUser@ALL
```

This filter cannot, of course, guard against the same person attempting to gain access under a different user name.

Creating Access Filters for Services

You can create Allow and Deny filters for the IMAP, POP, or HTTP services.

Console. To create filters by using Console, follow these steps:

1. In Console, open the Messaging Server that you want to create access filters for.
2. Click the Configuration tab.
3. Open the Services folder in the left pane and select IMAP, POP, or HTTP beneath the Services folder.
4. Click the Access tab in the right pane.

   The Allow and Deny fields in the tab show the existing Allow and Deny filters for that service. Each line in the field represents one filter. For either of the fields, you can specify the following actions:

   - Click Add to create a new filter. An Allow Filter window or Deny filter window opens; enter the text of the new filter into the window, and click OK.
   - Select a filter and click Edit to modify the filter. An Allow Filter window or Deny filter window opens; edit the text of the filter displayed in the window, and click OK.
   - Select a filter and click Delete to remove the filter.

Note that if you need to rearrange the order of Allow or Deny filters, you can do so by performing a series of Delete and Add actions.

For a specification of filter syntax and a variety of examples, see “Filter Syntax” on page 302. For additional examples, see “Filter Examples” on page 306.

Command Line. You can also specify access and deny filters at the command line as follows:

To create or edit access filters for services:

```
configutil -o service.service.domainallowed -v filter
```

where *service* is `pop`, `imap`, or `http` and *filter* follows the syntax rules described in “Filter Syntax” on page 302.

To create or edit deny filters for services:

```
configutil -o service.service.domainnotallowed -v filter
```

where *service* is `pop`, `imap`, or `http` and *filter* follows the syntax rules described in “Filter Syntax” on page 302.

Creating Access Filters for HTTP Proxy Authentication

Any store administrator can proxy authenticate to any service. (For more information about store administrators, see “Specifying Administrator Access to the Store” on page 242.) For the HTTP service only, any user can proxy authenticate to the service if their client host is granted access via a proxy authentication access filter.

Proxy authentication allows other services, such as a portal site, to authenticate users and pass the authentication credentials to the HTTP login service. For example, assume a portal site offers several services, one of which is Messenger Express web-based email. By using the HTTP proxy authentication feature, end users need only authenticate once to the portal service; they need not authenticate again to access their email. The portal site must configure a login server that acts as the interface between the client and the service. To help configure the login server for Messenger Express authentication, iPlanet offers an authentication SDK for Messenger Express.

This section describes how to create allow filters to permit HTTP proxy authentication by IP address. This section does not describe how to set up your login server or how to use the Messenger Express authentication SDK. For more information about setting up your login server for Messenger Express and using the authentication SDK, contact your iPlanet representative.

Console. To create access filters for proxy authentication to the HTTP service:

1. In Console, open the Messaging Server that you want to create access filters for.

2. Click the Configuration tab.
3. Open the Services folder in the left pane and select HTTP beneath the Services folder.
4. Click the Proxy tab in the right pane.

The Allow field in the tab shows the existing Allow filters for proxy authentication.

5. To create a new filter, click Add.

   An Allow filter window opens. Enter the text of the new filter into the window and click OK.

6. To edit an existing filter, select the filter and click Edit.

   An Allow filter window opens. Edit the text of the filter display in the window, and click OK.

7. To delete an existing filter, select a field from the Allow field, and click Delete.
8. When you are finished making changes to the Proxy tab, click Save.

For more information about allow filter syntax, see “Filter Syntax” on page 302.

Command Line. You can also specify access filters for proxy authentication to the HTTP service at the command line as follows:

```
configutil -o service.service.proxydomainallowed -v filter
```

where *filter* follows the syntax rules described in “Filter Syntax” on page 302.

Configuring Client Access to SMTP Services

For information about configuring client access to SMTP services, see Chapter 9, “Mail Filtering and Access Control.”

Chapter 12

Logging and Log Analysis

iPlanet Messaging Server can create log files that record events related to its administration, to communications using any of the protocols (SMTP, POP, IMAP, and HTTP) that the server supports, and to other processes employed by the server. By examining the log files, you can monitor many aspects of the server's operation.

Because the MTA uses a separate logging facility than the other services, you cannot use iPlanet Console to configure logging services and view logs. Instead, you configure MTA logging by specifying information in configuration files. Consequently this chapter is divided into three parts. The first part describes general introductory information; the second part describes logging for the message store and administration services; the third part describes logging for the MTA service.

PART 1: Introduction

PART 2: Service Logs (Message Store and Administration Server)

PART 3: Service Logs (MTA)

PART 1: Introduction

You can customize the policies for creating and managing the Messaging Server log files. This chapter describes the types and structure of log files, and discusses how to administer and how to view the log files.

Logged Services

Messaging Server creates a separate set of log files for each of the major protocols, or services, it supports. You can customize and view each type of log file individually. Table 12-1 lists the services that can be logged, and describes the log files for each service.

Table 12-1 Logged Services

Service	Log-file description
Admin	Contains logged events related to communication between iPlanet Console and Messaging Server (mostly through several CGI processes), by way of its Administration Server
SMTP	Contains logged events related to SMTP activity of this server
IMAP	Contains logged events related to IMAP4 activity of this server
POP	Contains logged events related to POP3 activity of this server
HTTP	Contains logged events related to HTTP activity of this server
Default	Contains logged events related to other activity of this server, such as command-line utilities and other processes

Analyzing Logs with Third-Party Tools

For log analyses and report generation beyond the capabilities of iPlanet Messaging Server, you need to use other tools. You can manipulate log files on your own with text editors or standard system tools.

With a scriptable text editor supporting regular-expression parsing, you can potentially search for and extract log entries based on any of the criteria discussed in this chapter, and possibly sort the results or even generate sums or other statistics.

In UNIX environments you might also be able to modify and use existing report-generation tools that were developed to manipulate UNIX `syslog` files. If you wish to use a public-domain `syslog` manipulation tool, remember that you may need to modify it to account for the different date / time format and for the two extra components (*facility* and *logLevel*) that appear in Messaging Server log entries but not in `syslog` entries.

PART 2: Service Logs (Message Store and Administration Server)

This section describes logging for the following services: POP, IMAP, HTTP, Admin, and Default (see Table 12-1).

For these services, you can use iPlanet Console to specify log settings and to view logs. The settings you specify affect which and how many events are logged. You can use those settings and other characteristics to refine searches for logged events when you are analyzing log files.

Part 2 contains the following sections:

- Log Characteristics
- Log File Format
- Defining and Setting Logging Options
- Searching and Viewing Logs

Log Characteristics

This section describes the following log characteristics for the message store and administration services: logging levels, categories of logged events, filename conventions for logs, and log-file directories.

Logging Levels

The level, or priority, of logging defines how detailed, or verbose, the logging activity is to be. A higher priority level means less detail; it means that only events of high priority (high severity) are logged. A lower level means greater detail; it means that more events are recorded in the log file.

You can set the logging level separately for each service—POP, IMAP, HTTP, Admin, and Default—and you can use logging levels to filter searches for log events. Table 12-2 describes the available levels. These logging levels are a subset of those defined by the UNIX `syslog` facility

Table 12-2 Logging Levels for Store and Administration Services

Level	Description
Critical	The minimum logging detail. An event is written to the log whenever a severe problem or critical condition occurs—such as when the server cannot access a mailbox or a library needed for it to run.
Error	An event is written to the log whenever an error condition occurs—such as when a connection attempt to a client or another server fails.
Warning	An event is written to the log whenever a warning condition occurs—such as when the server cannot understand a communication sent to it by a client.
Notice	An event is written to the log whenever a notice (a normal but significant condition) occurs—such as when a user login fails or when a session closes.
Informational	An event is written to the log with every significant action that takes place—such as when a user successfully logs on or off or creates or renames a mailbox.
Debugging	The most verbose logging. Useful only for debugging purposes. Events are written to the log at individual steps within each process or task, to pinpoint problems.

When you select a particular logging level, events corresponding to that level and to all higher (less verbose) levels are logged. The default level of logging is `Notice`.

NOTE The more verbose the logging you specify, the more disk space your log files will occupy; for guidelines, see “Defining and Setting Logging Options” on page 320.

Categories of Logged Events

Within each supported service or protocol, Messaging Server further categorizes logged events by the facility, or functional area, in which they occur. Every logged event contains the name of the facility that generated it. These categories aid in filtering events during searches. Table 12-3 lists the categories that Messaging Server recognizes for logging purposes.

Table 12-3 Categories in Which Log Events Occur

Facility	Description
General	Undifferentiated actions related to this protocol or service
LDAP	Actions related to Messaging Server accessing the LDAP directory database
Network	Actions related to network connections (socket errors fall into this category)
Account	Actions related to user accounts (user logins fall into this category)
Protocol	Protocol-level actions related to protocol-specific commands (errors returned by POP, IMAP, or HTTP functions fall into this category)
Stats	Actions related to the gathering of server statistics
Store	Low-level actions related to accessing the message store (read / write errors fall into this category)

For examples of using categories as filters in log searches, see “Searching and Viewing Logs” on page 324.

Filename Conventions for Message Store and Administration Logs

Log files for the POP, IMAP, HTTP, Admin, and Default service use identical naming conventions. Each log file has a filename of the form:

service.*sequenceNum*.*timeStamp*

Table 12-4 lists the message store log filename conventions.

Table 12-4 Filename Conventions for Store and Administration Logs

Component	Definition
service	The service being logged: POP, IMAP, HTTP, Admin, Default.
sequenceNum	An integer that specifies the order of creation of this log file compared to others in the log-file directory. Log files with higher sequence numbers are more recent than those with lower numbers. Sequence numbers do not roll over; they increase monotonically for the life of the server (beginning at server installation).
timeStamp	A large integer that specifies the date and time of file creation. (Its value is expressed in standard UNIX time: the number of seconds since midnight January 1, 1970.)

For example, a log file named `imap.63.915107696` would be the 63rd log file created in the directory of IMAP log files, created at 12:34:56 PM on December 31, 1998.

The combination of open-ended sequence numbering with a timestamp gives you more flexibility in rotating, expiring, and selecting files for analyzing. For more specific suggestions, see “Defining and Setting Logging Options” on page 320.

Log-File Directories

Every logged service is assigned a single directory, in which its log files are stored. All IMAP log files are stored together, as are all POP log files, and log files of any other service. You define the location of each directory, and you also define how many log files of what maximum size are permitted to exist in the directory.

Make sure that your storage capacity is sufficient for all your log files. Log data can be voluminous, especially at lower (more verbose) logging levels.

It is important also to define your logging level, log rotation, log expiration, and server-backup policies appropriately so that all of your log-file directories are backed up and none of them become overloaded; otherwise, you may lose information. See “Defining and Setting Logging Options” on page 320.

Log File Format

All message store and administration service log files created by Messaging Server have identical content formats. Log files are multiline text files, in which each line describes one logged event. All event descriptions, for each of the supported services, have the general format:

dateTime hostName processName[pid]: category logLevel: eventMessage

Table 12-5 lists the log file components. Note that this format of event descriptions is identical to that defined by the UNIX `syslog` facility, except that the date/time format is different and the format includes two additional components (*category* and *logLevel*).

Table 12-5 Store and Administration Log File Components

Component	Definition
dateTime	The date and time at which the event was logged, expressed in *dd/mm/yyyy hh:mm:ss* format, with a time-zone field expressed as *+/-hhmm* from GMT. For example: `02/Jan/1999:13:08:21 -0700`
hostName	The name of the host machine on which the server is running: for example, `showshoe`. **Note:** If there is more than one instance of Messaging Server on the host, you can use the process ID (pid) to separate logged events of one instance from another.
processName	The name of the process that generated the event: for example, `cgi_store`.
pid	The process ID of the process that generated the event: for example, `18753`.
category	The category that the event belongs to: for example, `General` (see Table 12-3 on page 317).
logLevel	The level of logging that the event represents: for example, `Notice` (see Table 12-2 on page 316).
eventMessage	An event-specific explanatory message that may be of any length: for example, `Log created (894305624)`.

Here are three examples of logged events as viewed using iPlanet Console:

```
02/May/1998:17:37:32 -0700 showshoe cgi_store[18753]:
General Notice:
  Log created (894155852)
```

```
04/May/1998:11:07:44 -0400 xyzmail cgi_service[343]: General Error:
  function=getserverhello|port=2500|error=failed to connect
```

```
03/Dec/1998:06:54:32 +0200 SiroePost imapd[232]: Account Notice:
  close [127.0.0.1] [unauthenticated] 1998/12/3 6:54:32
  0:00:00 0 115 0
```

IMAP and POP event entries may end with three numbers. The example above has 0 115 0. The first number is bytes sent by client, the second number is the bytes sent by the server, and third number is mailboxes selected (always 1 for POP).

When viewing a log file in the Log Viewer window, you can limit the events displayed by searching for any specific component in an event, such as a specific logging level or category, or a specific process ID. For more information, see “Searching and Viewing Logs” on page 324.

The event message of each log entry is in a format specific to the type of event being logged: that is, each service defines what content appears in any of its event messages. Many event messages are simple and self-evident; others are more complex.

Defining and Setting Logging Options

You can define the message store and administration service logging configurations that best serve your administration needs. This section discusses issues that may help you decide on the best configurations and policies, and it explains how to implement them.

Flexible Logging Architecture

The naming scheme for log files (*service.sequenceNum.timeStamp*) helps you to design a flexible log-rotation and backup policy. The fact that events for different services are written to different files makes it easier for you to isolate problems quickly. Also, because the sequence number in a filename is ever-increasing and the timestamp is always unique, later log files do not simply overwrite earlier ones after a limited set of sequence numbers is exhausted. Instead, older log files are overwritten or deleted only when the more flexible limits of age, number of files, or total storage are reached.

Messaging Server supports automatic rotation of log files, which simplifies administration and facilitates backups. You are not required to manually retire the current log file and create a new one to hold subsequent logged events. You can back up all but the current log file in a directory at any time, without stopping the server or manually notifying the server to start a new log file.

In setting up your logging policies, you can set options (for each service) that control limits on total log storage, maximum number of log files, individual file size, maximum file age, and rate of log-file rotation.

Planning the Options You Want

Keep in mind that you must set several limits, more than one of which might cause the rotation or deletion of a log file. Whichever limit is reached first is the controlling one. For example, if your maximum log-file size is 3.5 MB, and you specify that a new log be created every day, you may actually get log files created faster than one per day if log data builds up faster than 3.5 MB every 24 hours. Then, if your maximum number of log files is 10 and your maximum age is 8 days, you may never reach the age limit on log files because the faster log rotation may mean that 10 files will have been created in less than 8 days.

The following default values, provided for Messaging Server administration logs, may be a reasonable starting point for planning:

Maximum number of log files in a directory: 10
Maximum log-file size: 2 MB
Total maximum size permitted for all log files: 20 MB
Minimum free disk space permitted: 5 MB
Log rollover time: 1 day
Maximum age before expiration: 7 days
Level of logging: Notice

You can see that this configuration assumes that server-administration log data is predicted to accumulate at about 2 MB per day, backups are weekly, and the total space allotted for storage of admin logs is at least 25 MB. (These settings may be insufficient if the logging level is more verbose.)

For POP, IMAP or HTTP logs, the same values might be a reasonable start. If all services have approximately the same log-storage requirements as the defaults shown here, you might expect to initially plan for about 150 MB of total log-storage capacity. (Note that this is meant only as a general indication of storage requirements; your actual requirements may be significantly different.)

Setting Logging Options

You can set options that control the message store logging configuration by using iPlanet Console or the command line.

The optimal settings for these options depend on the rate at which log data accumulates. It may take between 4,000 and 10,000 log entries to occupy 1 MB of storage. At the more verbose levels of logging (such as Notice), a moderately busy server may generate hundreds of megabytes of log data per week. Here is one approach you can follow:

- Set a level of logging that is consistent with your storage limits—that is, a level that you estimate will cause log-data accumulation at approximately the rate you used to estimate the storage limit.
- Define the log file size so that searching performance is not impacted. Also, coordinate it with your rotation schedule and your total storage limit. Given the rate at which log entries accumulate, you might set a maximum that is slightly larger than what you expect to accumulate by the time a rotation automatically occurs. And your maximum file size times your maximum number of files might be roughly equivalent to your total storage limit.

  For example, if your IMAP log rotation is daily, your expected accumulation of IMAP log data is 3 MB per day, and your total storage limit for IMAP logs is 25 MB, you might set a maximum IMAP log-file size of 3.5 MB. (In this example, you could still lose some log data if it accumulated so rapidly that all log files hit maximum size and the maximum number of log files were reached.)
- If server backups are weekly and you rotate IMAP log files daily, you might specify a maximum number of IMAP log files of about 10 (to account for faster rotation if the individual log-size limit is exceeded), and a maximum age of 7 or 8 days.
- Pick a total storage limit that is within your hardware capacity and that coordinates with the backup schedule you have planned for the server. Estimate the rate at which you anticipate that log data will accumulate, add a factor of safety, and define your total storage limit so that it is not exceeded over the period between server backups.

  For example, if you expect to accumulate an average of 3 MB of IMAP log-file data per day, and server backups are weekly, you might specify on the order of 25 - 30 MB as the storage limit for IMAP logs (assuming that your disk storage capacity is sufficient).

- For safety, pick a minimum amount of free disk space that you will permit on the volume that holds the log files. That way, if factors other than log-file size cause the volume to fill up, old log files will be deleted before a failure occurs from attempting to write log data to a full disk.

Note that you can choose to send log information to the syslog facility instead of to the server-supplied log files. You can send log information to syslog by setting the `syslogfacility` option as follows:

`configutil -o logfile.`*service*`.syslogfacility -v` *value*

where *service* is `admin`, `pop`, `imap`, or `http` and *value* is `user`, `mail`, `daemon`, `local0` to `local7`, or `none`.

If the value is set, Messages are logged to the syslog facility corresponding to the set value and all the other log file service options are ignored. When the option is not set or the value is `none`, logging uses the Messaging Server log files.

Console. To set logging options using iPlanet Console:

1. Open the Messaging Server whose log file options you want to set.
2. Click the Configuration tab, open the Log Files folder in the left pane, and select the log files of a service (such as IMAP, HTTP, or Admin).
3. From the “Levels of detail” drop-down list, choose a logging level.
4. In the “Directory path for log files” field, enter the name of the directory to hold your log files.
5. In the “File size for each log” field, enter your maximum log-file size.
6. In the “Create new log every” field, enter a number for the log-rotation schedule.
7. In the “Number of logs per directory” and the “When a log is older than” fields, enter the maximum number of log files and a maximum age to coordinate with your backup schedule.
8. In the “When total log size exceeds” field, enter the total storage limit you want.
9. In the “When free disk space is less than” field, enter the minimum amount of free disk space you want to reserve.

Command Line. To set logging options at the command line, use the `configutil` command as shown in the following examples.

To set the logging level:

```
configutil -o logfile.service.loglevel -v level
```

where *service* is `admin`, `pop`, `imap`, or `http` and *loglevel* is `Nolog`, `Critical`, `Error`, `Warning`, `Notice`, `Information`, or `Debug`.

To specify a directory path for log files:

```
configutil -o logfile.service.logdir -v dirpath
```

To specify a maximum file size for each log:

```
configutil -o logfile.service.maxlogfilesize -v size
```

where *size* specifies a number of bytes.

To specify a log rotation schedule:

```
configutil -o logfile.service.rollovertime -v number
```

where *number* specifies a number of seconds.

To specify a maximum number of log files per directory:

```
configutil -o logfile.service.maxlogfiles -v number
```

To specify a storage limit:

```
configutil -o logfile.service.maxlogsize -v number
```

where *number* specifies a number in bytes.

To specify the a minimum amount of free disk space you want to reserve:

```
configutil -o logfile.service.minfreediskspace -v number
```

where *number* specifies a number in bytes.

To specify an age for logs at which they will expire:

```
configutil -o logfile.service.expirytime -v number
```

where *number* specifies a number in seconds.

Searching and Viewing Logs

iPlanet Console provides a basic interface for viewing message store and administration log data. It allows for selecting individual log files and for performing flexible filtered searches of log entries within those files.

For a given service, log files are listed in chronological order. Once you have chosen a log file to search, you can narrow the search for individual events by specifying search parameters.

Search Parameters

These are the search parameters you can specify for viewing log data:

- **A time period**. You can specify the beginning and end of a specific time period to retrieve events from, or you can specify a number of days (before the present) to search. You might typically specify a range to look at logged events leading up to a server crash or other occurrence whose time you know of. Alternatively, you might specify a day range to look at only today's events in the current log file.
- **A level of logging**. You can specify the logging level (see "Logging Levels" on page 315). You might select a specific level to uncover a specific problem; for example, Critical to see why the server went down, or Error to locate failed protocol calls.
- **A facility.** You can specify the facility (see "Categories of Logged Events" on page 317). You might select a specific facility if you know the functional area that contains the problem; for example, Store if you believe a server crash involved a disk error, or Protocol if the problem lies in an IMAP protocol command error.
- **A text search pattern.** You can provide a text search pattern to further narrow the search. You can include any component of the event (see "Log File Format" on page 319) that can be expressed in a wildcard-type search, such as event time, process name, process ID, and any part of the event message (such as remote host name, function name, error number, and so on) that you know defines the event or events you want to retrieve.

  Your search pattern can include the following special and wildcard characters:

  `*` Any set of characters (example: `*.com`)
  `?` Any single character (example: `199?`)
  `[`*nnn*`]` Any character in the set *nnn* (example: `[aeiou]`)
  `[^`*nnn*`]` Any character not in the set *nnn* (example: `[^aeiou]`)
  `[`*n*`-`*m*`]` Any character in the range *n*-*m* (example: `[A-Z]`)
  `[^`*n*`-`*m*`]` Any character not in the range *n*-*m* (example: `[^0-9]`)
  \ Escape character: place before `*`, `?`, `[`, or `]` to use them as literals

Note: Searches are case-sensitive.

Examples of combining logging level and facility in viewing logs might include the following:

- Specifying Account facility (and Notice level) to display failed logins, which may be useful when investigating potential security breaches

- Specifying Network facility (and all logging levels) to investigate connection problems
- Specifying all facilities (and Critical logging level) to look for basic problems in the functioning of the server

Specifying a Search and Viewing Results

Follow these steps to search for logged events with specific characteristics belonging to a given service:

1. In iPlanet Console, open the Messaging Server whose log files you want to inspect.
2. Follow either of these steps to display the Log Files Content tab for a given logged service:
   - Click the Tasks tab, then click "View *service* logs", where *service* is the name of the logged service (such as "IMAP service" or "administration").
   - Click the Configuration tab, then open the Log Files folder in the left pane and select the log files of a service (such as IMAP or Admin). Then click the Content tab in the right pane.
3. The Content tab for that logged service is displayed.
4. In the Log filename field, select the log file you want to examine.
5. Click the View selected log button to open the Log Viewer window.
6. In the Log Viewer window, specify your desired search parameters (described in the previous section, "Search Parameters").
7. Click Update to perform the search and display the results in the Log entry field.

PART 3: Service Logs (MTA)

The MTA provides facilities for logging each message as it is enqueued and dequeued. You can control logging on a per-channel basis or you can specify that message activity on all channels be logged. In the initial configuration, logging is disabled on all channels.

Enabling logging causes the MTA to write an entry to a `mail.log*` file each time a message passes through an MTA channel. Such log entries can be useful if you wish to get statistics on how many messages are passing through the MTA (or through particular channels), or when investigating other questions such as whether and when a message was sent or delivered.

If you are only interested in gathering statistics on the number of messages passing through a few particular MTA channels, then you may wish to enable the logging channel keyword on just those MTA channels of main interest. Many sites prefer to enable logging on all MTA channels. In particular, if you are trying to track down problems, the first step in diagnosing some problems is to notice that messages are not going to the channel you expected or intended, and having logging enabled for all channels can help you investigate such problems.

CAUTION If logging is enabled, `mail.log` steadily grows and, if left unchecked, consumes all available disk space. Monitor the size of this file and periodically delete unnecessary contents. You can also delete the entire file as another version will be created as needed.

Enabling MTA Logging

To enable logging for a particular channel, you add the `logging` keyword to the channel definition in the MTA configuration file, as shown in the following example:

channel-name keyword1 keyword2 `logging`

If you wish to have all of your channels log message activity to the logging file, then simply add a defaults channel block to the start of the channel block section of your MTA configuration file. For example:

```
defaults logging

l defragment charset7 us-ascii charset8 iso-8859-01
siroe.com
```

The `defaults` channel would appear immediately after the first blank line in the MTA configuration file. It is important that a blank line appear before and after the line `defaults logging`.

Each message is logged as it is enqueued and dequeued. All log entries are made to the file `mail.log_current` in the MTA log directory: `msg-`*`instance`*`/log/imta/mail.log_current`.

The message return job, which runs every night around midnight, appends any existing `mail.log_yesterday` to the cumulative log file, `mail.log`, renames the current `mail.log_current` file to `mail.log_yesterday`, and then begins a new `mail.log_current` file. It also performs the analogous operations for any `connection.log*` files.

You can send MTA log messages to syslog by setting the `LOG_MESSAGES_SYSLOG` option.

Specifying Additional MTA Logging Options

In addition to the basic information always provided when logging is enabled, you can specify that additional, optional information fields be included by setting various `LOG_*` MTA options in the MTA Option file. For complete details about the Option file, see the *Messaging Server Reference Manual*.

- `LOG_MESSAGE_ID`. This option allows correlation of which entries relate to which message.
- `LOG_FILENAME`. This option makes it easier to immediately spot how many times delivery of a particular message file has been retried, and can be useful in understanding when the MTA does or does not split a message to multiple recipients into separate message file copies on disk.
- `LOG_CONNECTION`. This option causes the MTA to log TCP/IP connections, as well as message traffic. The connection log entries are written to the `mail.log*` files by default, or may optionally be written to `connection.log*` files; see `SEPARATE_CONNECTION_LOG` option.
- `SEPARATE_CONNECTION_LOG`. This option may be used to specify that connection log entries instead be written to `connection.log` files.
- `LOG_PROCESS`. When used in conjunction with `LOG_CONNECTION`, this option allows correlation by process id of which connection entries correspond to which message entries.
- `LOG_USERNAME`. This option controls whether or not the user name associated with a process that enqueues mail is saved in the `mail.log` file. For SMTP submissions where SASL (SMTP AUTH) is used, the user name field will be the authenticated user name (prefixed with an asterisk character).

MTA Log Entry Format

The MTA log file is written as ASCII text. By default, each log file entry contains eight or nine fields as shown in Figure 12-1.

Figure 12-1 MTA Log Entry Format

```
19-Jan-1998 19:16:57.64 l tcp_local E 1 adam@sesta.com
 rfc822;marlowe@siroe.com marlowe@siroe.com
```

The log entry shows:

1. The date and time the entry was made.
2. The channel name for the source channel (in the example, `l`).
3. The channel name for the destination channel (in the example, `tcp_local`). (For SMTP channels, when `LOG_CONNECTION` is enabled, a plus, +, indicates inbound to the SMTP server; a minus, -, indicates outbound via the SMTP client.)
4. The type of entry (E); see Table 12-6.
5. The size of the message (1). This is expressed in kilobytes by default, although this default can be changed by using the `BLOCK_SIZE` keyword in the MTA option file.
6. The envelope From: address (`adam@sesta.com`). Note that for messages with an empty envelope From: address, such as notification messages, this field will be blank.
7. The original form of the envelope To: address (`marlowe@siroe.com`).
8. The active (current) form of the envelope To: address (`marlowe@siroe.com`).
9. The delivery status (SMTP channels only).

Table 12-6 describes the logging entry codes.

Table 12-6 Logging Entry Codes

Entry	Description
D	Successful dequeue
DA	Successful dequeue with SASL (authentication)

Table 12-6 Logging Entry Codes

Entry	Description
DS	Successful dequeue with TLS (security)
DSA	Successful dequeue with TLS and SASL (security and authentication)
E	Enqueue
EA	Successful enqueue with SASL (authentication)
ES	Successful enqueue with TLS (security)
ESA	Successful enqueue with TLS and SASL (security and authentication)
J	Rejection of attempted enqueue (rejection by slave channel program)
Q	Temporary failure to dequeue
R	Recipient address rejected on attempted dequeue (rejection by master channel program), or generation of a failure / bounce message
W	Warning message generated regarding a not-yet-delivered message
Z	Some successful recipients, but this recipient was temporarily unsuccessful; the original message file of all recipients was dequeued, and in its place a new message file for this and other unsuccessful recipients will be immediately enqueued
SMTP channels' LOG_CONNECTION + or - entries	
C	Connection closed
O	Connection opened
X	Connection rejected
Y	Connection attempt failed before being established
I	ETRN command received

With `LOG_CONNECTION`, `LOG_FILENAME`, `LOG_MESSAGE_ID`, `LOG_NOTARY`, `LOG_PROCESS`, and `LOG_USERNAME` all enabled, the format becomes as shown in Figure 12-2. (The sample log entry line has been wrapped for typographic reasons; the actual log entry would appear on one physical line.)

Figure 12-2 Log Format with Additional Fields

```
19-Jan-1998 13:13:27.10 HOSTA   2e2d.2.1 tcp_local   l
 E 1 service@siroe.com rfc822;adam@sesta.com
 adam 276 /imta/queue/l/ZZ01IWFY9ELGWM00094D.00
 <01IWFVYLGTS499EC9Y@siroe.com> inetmail
 siroe.com (siroe.com [192.160.253.66])
```

Where the additional fields, beyond those already discussed above, are:

1. The name of the node on which the channel process is running (in the example, `HOSTA`).
2. The process id (expressed in hexadecimal), followed by a period (dot) character and a count. If this had been a multithreaded channel entry (e.g., a `tcp_*` channel entry), there would also be a thread id present between the process id and the count. In the example, the process id is `2e2d.2.1`.
3. The NOTARY (delivery receipt request) flags for the message, expressed as an integer (in the example, 276).
4. The file name in the MTA queue area (in the example, `/imta/queue/l/ZZ01IWFY9ELGWM00094D.00`).
5. The message id (in the example, `<01IWFVYLGTS499EC9Y@siroe.com>`).
6. The name of the executing process (in the example, `inetmail`). On UNIX, for dispatcher processes such as the SMTP server, this will usually be `inetmail` (unless SASL was used).
7. The connection information (in the example, `siroe.com (siroe.com [192.160.253.66]`). The connection information consists of the sending system or channel name, such as the name presented by the sending system on the HELO/EHLO line (for incoming SMTP messages), or the enqueuing channel's official host name (for other sorts of channels). In the case of TCP/IP channels, the sending system's “real” name, that is, the symbolic name as reported by a DNS reverse lookup and/or the IP address, can also be reported within parentheses as controlled by the `ident*` channel keywords; see “IDENT Lookups” on page 170. This sample assumes use of one of these keywords, for instance us of the default `identnone` keyword, that selects display of both the name found from the DNS and IP address.

Managing the MTA Log Files

The message return job, which runs every night around midnight, appends any existing `mail.log_yesterday` to the cumulative log file, `mail.log`, renames the current `mail.log_current` file to `mail.log_yesterday`, and then begins a new `mail.log_current` file. It also performs the analogous operations for any `connection.log*` files.

The MTA performs automatic rollovers to maintain the current file, but you must manage the cumulative `mail.log` file by determining policies for tasks such as backing up the file, truncating the file, deleting the file, and so on.

When considering how to manage the log files, note that the MTA periodic return job will execute a site-supplied *server-instance*`/imta/bin/daily_cleanup` procedure, if one exists. Thus some sites might choose to supply their own cleanup procedure that, for instance, renames the old `mail.log` file once a week (or once a month), and so on.

Examples of MTA Message Logging

The exact field format and list of fields logged in the MTA message files will vary according to exactly what logging options you set. This section will show a few examples of interpreting typical sorts of log entries. For a description of additional, optional fields, see “Specifying Additional MTA Logging Options” on page 328.

NOTE For typographic reasons, log file entries will be shown folded onto multiple lines—actual log file entries are one line per entry.

When reviewing a log file, keep in mind that on a typical system many messages are being handled at once. Typically, the entries relating to a particular message will be interspersed among entries relating to other messages being processed during that same time. The basic logging information is suitable for gathering a sense of the overall numbers of messages moving through the MTA.

If you wish to correlate particular entries relating to the same message to the same recipient(s), you will probably want to enable `LOG_MESSAGE_ID`. If you wish to correlate particular messages with particular files in the MTA queue area, or to see from the entries how many times a particular not-yet-successfully-dequeued message has had delivery attempted, you will probably want to enable `LOG_FILENAME`. For SMTP messages (handled via a TCP / IP channel), if you want to correlate TCP connections to and from remote systems with the messages sent, you will probably want to enable `LOG_PROCESS` and some level of `LOG_CONNECTION`.

Figure 12-3 show a fairly basic example of the sorts of log entries one might see if a local user sends a message out an outgoing TCP / IP channel, for example, to the Internet. In this example, `LOG_CONNECTION` is enabled. The lines marked with (1) and (2) are one entry—they would appear on one physical line in an actual log file. Similarly, the lines marked with (3) - (7) are one entry and would appear on one physical line.

Figure 12-3 Logging: A Local User Sends An Outgoing Message

```
19-Jan-1998 19:16:57.64 l            tcp_local    E 1 (1)
 adam@sesta.com rfc822;marlowe@siroe.com marlowe@siroe.com (2)

19-Jan-1998 19:17:01.16 tcp_local                 D 1 (3)
 adam@sesta.com rfc822;marlowe@siroe.com marlowe@siroe.com (4)
 dns;thor.siroe.com
 (TCP|206.184.139.12|2788|192.160.253.66|25) (5)
 (THOR.SIROE.COM -- Server ESMTP [iMS V5.0 #8694]) (6)
 smtp;250 2.1.5 marlowe@siroe.com and options OK. (7)
```

1. This line shows the date and time of an enqueue (E) from the `l` channel to the `tcp_local` channel of a one (1) block message.
2. This is part of the same physical line of the log file as (1), presented here as a separate line for typographical convenience. It shows the envelope From: address, in this case `adam@sesta.com`, and the original version and current version of the envelope To: address, in this case `marlowe@siroe.com`.
3. This shows the date and time of a dequeue (D) from the `tcp_local` channel of a one (1) block message—that is, a successful send by the `tcp_local` channel to some remote SMTP server.
4. This shows the envelope From: address, the original envelope To: address, and the current form of the envelope To: address.
5. This shows that the actual system to which the connection was made is named `thor.siroe.com` in the DNS, that the local sending system has IP address 206.184.139.12 and is sending from port 2788, that the remote destination system has IP address 192.160.253.66 and the connection port on the remote destination system is port 25.
6. This shows the SMTP banner line of the remote SMTP server.

7. This shows the SMTP status code returned for this address; 250 is the basic SMTP success code and in addition, this remote SMTP server responds with extended SMTP status codes and some additional text.

Figure 12-4 shows a logging entry similar to that shown in Figure 12-3, but with the additional information logged by setting `LOG_FILENAME=1` and `LOG_MESSAGE_ID=1` showing the filename and message-id; see (1) and (2). The message-id in particular can be used to correlate which entries relate to which message.

Figure 12-4 Logging: Including Optional Logging Fields

```
19-Jan-1998 19:16:57.64 l            tcp_local    E 1
 adam@sesta.com rfc822;marlowe@siroe.com marlowe@siroe.com
 /imta/queue/tcp_local/ZZ01ISKLSKLZLI90N15M.00
   <01ISKLSKC2QC90N15M@sesta.com> (1)

19-Jan-1998 19:17:01.16 tcp_local                 D 1
 adam@sesta.com rfc822;marlowe@siroe.com marlowe@siroe.com
 /imta/queue/tcp_local/Z01ISKLSKLZLI90N15M.00
   <01ISKLSKC2QC90N15M@sesta.com> (2)
 dns;thor.siroe.com (TCP|206.184.139.12|2788|192.160.253.66|25)
 (THOR.SIROE.COM -- Server ESMTP [iMS V5.0 #8694])
 smtp;250 2.1.5 marlowe@siroe.com and options OK.
```

Figure 12-5 illustrates sending to multiple recipients with `LOG_FILENAME=1`, `LOG_MESSAGE_ID=1`, and `LOG_CONNECTION=1` enabled. Here user `adam@sesta.com` has sent to the MTA mailing list `test-list@sesta.com`, which expanded to `bob@sesta.com`, `carol@varrius.com`, and `david@varrius.com`. Note that the original envelope To: address is `test-list@sesta.com` for each recipient, though the current envelope To: address is each respective address. Note how the message-id is the same throughout, though two separate files (one for the `l` channel and one going out the `tcp_local` channel) are involved.

Figure 12-5 Logging: Sending to a List

```
19-Jan-1998 20:01:44.10 l            l            E 1
 adam@sesta.com rfc822;test-list@sesta.com bob
 imta/queue/l/ZZ01ISKND3DE1K90N15M.00
 <01ISKND2H8MS90N15M@sesta.com>

19-Jan-1998 20:01:44.81 l            tcp_local    E 1
 adam@sesta.com rfc822;test-list@sesta.com carol@varrius.com
 imta/queue/tcp_local/ZZ01ISKND2WS1I90N15M.00
 <01ISKND2H8MS90N15M@sesta.com>

19-Jan-1998 20:01:44.81 l            tcp_local    E 1
 adam@sesta.com rfc822;test-list@sesta.com david@varrius.com
 imta/queue/tcp_local/ZZ01ISKND2WS1I90N15M.00
 <01ISKND2H8MS90N15M@sesta.com>

19-Jan-1998 20:01:50.69 l                         D 1
 adam@sesta.com rfc822;test-list@sesta.com bob
 imta/queue/l/ZZ01ISKND3DE1K90N15M.00
 <01ISKND2H8MS90N15M@sesta.com>

19-Jan-1998 20:01:57.36 tcp_local                 D 1
 adam@sesta.com rfc822;test-list@sesta.com carol@varrius.com
 imta/queue/tcp_local/ZZ01ISKND2WS1I90N15M.00
 <01ISKND2H8MS90N15M@sesta.com>
 dns;gw.varrius.com (TCP|206.184.139.12|2788|192.160.253.66|25)
 (gw.varrius.com -- SMTP Sendmail)
 smtp;250 OK.

19-Jan-1998 20:02:06.14 tcp_local                 D 1
 adam@sesta.com rfc822;test-list@sesta.com david@varrius.com
 imta/queue/tcp_local/ZZ01ISKND2WS1I90N15M.00
 <01ISKND2H8MS90N15M@sesta.com>
 dns;gw.varrius.com (TCP|206.184.139.12|2788|192.160.253.66|25)
 (gw.varrius.com -- SMTP Sendmail)
 smtp;250 OK.
```

Figure 12-6 illustrates an attempt to send to a non-existent domain (here `very.bogus.com`); that is, sending to a domain name that is not noticed as non-existent by the MTA's rewrite rules and that the MTA matches to an outgoing TCP/IP channel. This example assumes the MTA option settings of `LOG_FILENAME=1` and `LOG_MESSAGE_ID=1`.

When the TCP/IP channel runs and checks for the domain name in the DNS, the DNS returns an error that no such name exists. Note the “rejection” entry (R), as seen in (5), with the DNS returning an error that this is not a legal domain name, as seen in (6).

Because the address is rejected after the message has been submitted, the MTA generates a bounce message to the original sender. The MTA enqueues the new rejection message to the original sender (1), and sends a copy to the postmaster (4) before deleting the original outbound message (the R entry shown in (5)).

Notification messages, such as bounce messages, have an empty envelope From: address—as seen, for instance, in (2) and (8)—in which the envelope From: field is shown as an empty space. The initial enqueue of a bounce message generated by the MTA shows the message-id for the new notification message followed by the message-id for the original message (3). (Such information is not always available to the MTA, but when it is available to be logged, it allows correlation of the log entries corresponding to the outbound failed message with the log entries corresponding to the resulting notification message.) Such notification messages are enqueued to the process channel, which in turn enqueues them to an appropriate destination channel (7).

Figure 12-6 Logging: Sending to a Non-existent Domain

```
19-JAN-1998 20:49:04 l            tcp_local    E 1
 adam@sesta.com rfc822;user@very.bogus.com user@very.bogus.com
 imta/queue/tcp_local/ZZ01ISKP0S0LVQ94DU0K.00
 <01ISKP0RYMAS94DU0K@SESTA.COM>

19-JAN-1998 20:49:33 tcp_local    process      E 1 (1)
 rfc822;adam@sesta.com adam@sesta.com  (2)
 imta/queue/process/ZZ01ISKP0S0LVQ94DTZB.00
 <01ISKP22MW8894DTAS@SESTA.COM>,<01ISKP0RYMAS94DU0K@SESTA.COM>
(3)

19-JAN-1998 20:49:33 tcp_local    process      E 1 (4)
 rfc822;postmaster@sesta.com postmaster@sesta.com
 imta/queue/process/ZZ01ISKP0S0LVQ94DTZB.00
 <01ISKP22MW8894DTAS@SESTA.COM>,<01ISKP0RYMAS94DU0K@SESTA.COM>

19-JAN-1998 20:50:07 tcp_local                 R 1 (5)
 adam@sesta.com rfc822;user@very.bogus.com user@very.bogus.com
 imta/queue/tcp_local/ZZ01ISKP0S0LVQ94DU0K.00
 <01ISKP0RYMAS94DU0K@SESTA.COM>
 Illegal host/domain name found (6)

19-JAN-1998 20:50:08 process      l            E 3 (7)
 rfc822;adam@sesta.com adam (8)
 imta/queue/l/ZZ01ISKP23BUQS94DTYL.00
 <01ISKP22MW8894DTAS@SESTA.COM>

19-JAN-1998 20:50:08 process      l            E 3
 rfc822;postmaster@sesta.com postmaster
 imta/queue/l/ZZ01ISKP23BUQS94DTYL.00
 <01ISKP22MW8894DTAS@SESTA.COM>

19-JAN-1998 20:50:12 l                         D 3
 rfc822;adam@sesta.com adam
 imta/queue/l/ZZ01ISKP23BUQS94DTYL.00
 <01ISKP22MW8894DTAS@SESTA.COM>

19-JAN-1998 20:50:12 l                         D 3
 rfc822;postmaster@sesta.com postmaster
 imta/queue/l/ZZ01ISKP23BUQS94DTYL.00
 <01ISKP22MW8894DTAS@SIROE.COM>
```

Figure 12-7 illustrates an attempt to send to a bad address on a remote system. This example assumes MTA option settings of `LOG_FILENAME=1` and `LOG_MESSAGE_ID=1`, and channel option settings of `LOG_BANNER=1` and `LOG_TRANSPORTINFO=1`. Note the rejection entry (R), seen in (1). But in contrast to the rejection entry in Figure 12-6, note that the rejection entry here shows that a connection to a remote system was made, and shows the SMTP error code issued by the remote SMTP server, (2) and (3). The inclusion of the information shown in (2) is due to setting the channel options `LOG_BANNER=1` and `LOG_TRANSPORTINFO=1`.

Figure 12-7 Logging: Sending to a Non-existent Remote User

```
20-JAN-1998 13:11:05 l            tcp_local    E 1
 adam@sesta.com rfc822;nonesuch@siroe.com nonesuch@siroe.com
 imta/queue/tcp_local/ZZ01ISLNBB1JOE94DUWH.00
 <01ISLNBAWV3094DUWH@sesta.com>

20-JAN-1998 13:11:08 tcp_local    process      E 1
 rfc822;adam@sesta.com adam@sesta.com
 imta/queue/process/ZZ01ISLNBB1JOE94DSGB.00
 <01ISLNBFKIDS94DUJ8@sesta.com>,<01ISLNBAWV3094DUWH@sesta.com>

20-JAN-1998 13:11:08 tcp_local    process      E 1
 rfc822;postmaster@sesta.com postmaster@sesta.com
 imta/queue/process/ZZ01ISLNBB1JOE94DSGB.00
 <01ISLNBFKIDS94DUJ8@sesta.com>,<01ISLNBAWV3094DUWH@sesta.com>

20-JAN-1998 13:11:11 tcp_local                 R 1  (1)
 adam@sesta.com rfc822;nonesuch@siroe.com nonesuch@siroe.com
 imta/queue/tcp_local/ZZ01ISLNBB1JOE94DUWH.00
 <01ISLNBAWV3094DUWH@sesta.com>
 dns;thor.siroe.com
 (TCP|206.184.139.12|2788|192.160.253.66|25)        (2)
 (THOR.SIROE.COM -- Server ESMTP [iMS V5.0 #8694])
 smtp; 553 unknown or illegal user: nonesuch@siroe.com  (3)

20-JAN-1998 13:11:12 process      l            E 3
 rfc822;adam@sesta.com adam
 imta/queue/l/ZZ01ISLNBGND1094DQDP.00
 <01ISLNBFKIDS94DUJ8@sesta.com>

20-JAN-1998 13:11:12 process      l            E 3
 rfc822;postmaster@sesta.com postmaster
 imta/queue/l/ZZ01ISLNBGND1094DQDP.00
 <01ISLNBFKIDS94DUJ8@sesta.com>

20-JAN-1998 13:11:13 l                         D 3
 rfc822;adam@sesta.com adam@sesta.com
 imta/queue/l/ZZ01ISLNBGND1094DQDP.00
 <01ISLNBFKIDS94DUJ8@sesta.com>

20-JAN-1998 13:11:13 l                         D 3
 rfc822;postmaster@sesta.com postmaster@sesta.com
 imta/queue/l/ZZ01ISLNBGND1094DQDP.00
 <01ISLNBFKIDS94DUJ8@sesta.com>
```

Figure 12-8 illustrates the sort of log file entry resulting when the MTA rejects a remote side's attempt to submit a message. (This example assumes that no optional LOG_* options are enabled, so only the basic fields are logged in the entry. Note that enabling the LOG_CONNECTION option, in particular, would result in additional informative fields in such J entries.) In this case, the example is for an MTA that has set up SMTP relay blocking (see "Configuring SMTP Relay Blocking" on page 217) with an ORIG_SEND_ACCESS mapping including:

```
ORIG_SEND_ACCESS

! ...numerous entries omitted...
!
   tcp_local|*|tcp_local|*   $NRelaying$ not$ permitted
```

and where alan@very.bogus.com is not an internal address. Hence the attempt of the remote user harold@varrius.com to relay through the MTA system to the remote user alan@very.bogus.com is rejected.

Figure 12-8 Logging: Rejecting a Remote Side's Attempt to Submit a Message

```
28-May-1998 12:02:23 tcp_local                J 0     (1)
 harold@varrius.com rfc822; alan@very.bogus.com       (2)
 550 5.7.1 Relaying not permitted: alan@very.bogus.com (3)
```

1. This log shows the date and time the MTA rejects a remote side's attempt to submit a message. The rejection is indicated by a J record. (Cases where an MTA channel is attempting to send a message which is rejected is indicated by R records, as shown in Figure 12-6 and Figure 12-7).
2. The attempted envelope From: and To: addresses are shown. In this case, no original envelope To: information was available so that field is empty.
3. The entry includes the SMTP error message the MTA issued to the remote (attempted sender) side.

Figure 12-9 illustrates the sort of log file entries resulting when a message cannot be delivered upon the first attempt, so the MTA attempts to send the message several times. This example assumes option settings of LOG_FILENAME=1 and LOG_MESSAGE_ID=1.

Figure 12-9 Logging: Multiple Delivery Attempts

```
15-Jan-1998 10:31:05.18 tcp_internal   tcp_local  E 3 (1)
 adam@hosta.sesta.com rfc822;user@some.org user@some.org
 imta/queue/tcp_local/ZZ01IS3D2ZP7FQ9UN54R.00
 <01IRUD7SVA3Q9UN2D4@sesta.com>

15-Jan-1998 10:31:10.37 tcp_local                Q 3 (2)
 adam@hosta.sesta.com rfc822;user@some.org user@some.org
 imta/queue/tcp_local/ZZ01IS3D2ZP7FQ9UN54R.00        (3)
 <01IRUD7SVA3Q9UN2D4@sesta.com>
 TCP active open: Failed connect()    Error: no route to host (4)

 ...several hours worth of entries...

15-Jan-1998 12:45:39.48 tcp_local                Q 3 (5)
 adam@hosta.sesta.com rfc822;user@some.org user@some.org
 imta/queue/tcp_local/ZY01IS3D2ZP7FQ9UN54R.00        (6)
 <01IRUD7SVA3Q9UN2D4@sesta.com>
 TCP active open: Failed connect()    Error: no route to host

 ...several hours worth of entries...

15-Jan-1998 16:45:24.72 tcp_local                Q 3
 adam@hosta.sesta.com rfc822;user@some.org user@some.org
 imta/queue/tcp_local/ZX01IS67NY4RRK9UN7GP.00        (7)
 <01IRUD7SVA3Q9UN2D4@sesta.com>
 TCP active open: Failed connect() Error: connection refused (8)

 ...several hours worth of entries...

15-Jan-1998 20:45:51.55 tcp_local                D 3 (9)
 adam@hosta.sesta.com rfc822;user@some.org user@some.org
 imta/queue/tcp_local/ZX01IS67NY4RRK9UN7GP.00
 <01IRUD7SVA3Q9UN2D4@sesta.com>
 dns;host.some.org (TCP|206.184.139.12|2788|192.1.1.1|25)
 (All set, fire away)
 smtp; 250 Ok
```

1. The message comes in the `tcp_internal` channel—perhaps from a POP or IMAP client, or perhaps from another host within the organization using the MTA as an SMTP relay; the MTA enqueues it to the outgoing `tcp_local` channel.

2. The first delivery attempt fails, as indicated by the Q entry.
3. That this is a first delivery attempt can be seen from the ZZ* filename.
4. This delivery attempt failed when the TCP/IP package could not find a route to the remote side. As opposed to Figure 12-6, the DNS did not object to the destination domain name, some.org; rather, the "no route to host" error indicates that there is some network problem between the sending and receiving side.
5. The next time the MTA periodic job runs it reattempts delivery, again unsuccessfully.
6. The file name is now ZY*, indicating that this is a second attempt.
7. The file name is ZX* for this third unsuccessful attempt.
8. The next time the periodic job reattempts delivery the delivery fails, though this time the TCP/IP package is not complaining that it cannot get through to the remote SMTP server, but rather the remote SMTP server is not accepting connections. (Perhaps the remote side fixed their network problem, but has not yet brought their SMTP server back up—or their SMTP server is swamped handling other messages and hence was not accepting connections at the moment the MTA tried to connect.)
9. Finally the message is dequeued.

Figure 12-10 illustrates the case of a message routed through the conversion channel. The site is assumed to have a CONVERSIONS mapping table such as:

```
CONVERSIONS

  IN-CHAN=tcp_local;OUT-CHAN=l;CONVERT   Yes
```

This example assumes option settings of LOG_FILENAME=1 and LOG_MESSAGE_ID=1.

Figure 12-10 Logging: Incoming SMTP Message Routed Through the Conversion Channel

```
04-Feb-1998 00:06:26.72 tcp_local    conversion   E 9 (1)
 amy@siroe.edu rfc822;bert@sesta.com bert@sesta.com
 imta/queue/conversion/ZZ01IT5UAMZ4QW98518O.00
 <01IT5UALL14498518O@siroe.edu>

04-Feb-1998 00:06:29.06 conversion   l            E 9 (2)
 amy@siroe.edu rfc822;bert@sesta.com bert
 imta/queue/l/ZZ01IT5UAOXLDW98509E.00
 <01IT5STUMUFO984Z8L@siroe.edu>

04-Feb-1998 00:06:29.31 conversion                D 9 (3)
 amy@siroe.edu rfc822;bert@sesta.com bert
 imta/queue/conversion/ZZ01IT5UAMZ4QW98518O.00
 <01IT5UALL14498518O@siroe.edu>

04-Feb-1998 00:06:32.62 l                         D 9 (4)
 amy@siroe.edu rfc822;bert@siroe.com bert
 imta/queue/l/ZZ01IT5UAOXLDW98509E.00
 <01IT5STUMUFO984Z8L@siroe.edu>
```

1. The message from external user `amy@siroe.edu` comes in addressed to the `l` channel recipient `bert@sesta.com`. The `CONVERSIONS` mapping entry, however, causes the message to be initially enqueued to the conversion channel (rather than directly to the `l` channel).
2. The conversion channel runs and enqueues the message to the `l` channel.
3. Then the conversion channel can dequeue the message (delete the old message file).
4. And finally the `l` channel dequeues (delivers) the message.

Figure 12-11 illustrates log output for an outgoing message when connection logging is enabled, via `LOG_CONNECTION=3`. `LOG_PROCESS=1`, `LOG_MESSAGE_ID=1` and `LOG_FILENAME=1` are also assumed in this example. The example shows the case of user `adam@sesta.com` sending the same message (note that the message ID is the same for each message copy) to three recipients, `bobby@hosta.sesta.com`, `carl@hosta.sesta.com`, and `dave@hostb.sesta.com`. This example assumes that the message is going out a `tcp_local` channel marked (as such channels usually are) with the `single_sys` channel keyword. Therefore, a separate message file on

disk will be created for each set of recipients to a separate host name, as seen in (1), (2), and (3), where the bobby@hosta.sesta.com and carl@hosta.sesta.com recipients are stored in the same message file, but the dave@hostb.sesta.com recipient is stored in a different message file.

Figure 12-11 Logging: Outbound Connection Logging

```
19-Feb-1998 10:52:05.41 1e488.0 l            tcp_local    E 1
 adam@sesta.com rfc822;bobby@hosta.sesta.com bobby@hosta.sesta.com
 imta/queue/tcp_local/ZZ01ITRF7BO388000FCN.00  (1)
  <01ITRF7BDHS6000FCN@SESTA.COM>

19-Feb-1998 10:52:05.41 1e488.0 l            tcp_local    E 1
 adam@sesta.com rfc822;carl@hosta.sesta.com carl@hosta.sesta.com
 imta/queue/tcp_local/ZZ01ITRF7BO388000FCN.00  (2)
  <01ITRF7BDHS6000FCN@SESTA.COM>

19-Feb-1998 10:52:05.74 1e488.1 l            tcp_local    E 1
 adam@sesta.com rfc822;dave@hostb.sesta.com dave@hostb.sesta.com
 imta/queue/tcp_local/ZZ01ITRF7C11FU000FCN.00  (3)
  <01ITRF7BDHS6000FCN@SESTA.COM>

19-Feb-1998 10:52:10.79 1f625.2.0 tcp_local    -            O (4)
 TCP|206.184.139.12|5900|206.184.139.66|25
 SMTP/hostb.sesta.com/mailhub.sesta.com         (5)

19-Feb-1998 10:52:10.87 1f625.3.0 tcp_local    -            O (6)
 TCP|206.184.139.12|5901|206.184.139.70|25
 SMTP/hosta.sesta.com/hosta.sesta.com          (7)

19-Feb-1998 10:52:12.28 1f625.3.1 tcp_local                 D 1
 adam@sesta.com rfc822;bobby@hosta.sesta.com bobby@hosta.sesta.com
 imta/queue/tcp_local/ZZ01ITRF7BO388000FCN.00
  <01ITRF7BDHS6000FCN@SESTA.COM>
 hosta.sesta.com dns;hosta.sesta.com          (8)
 (TCP|206.184.139.12|5901|206.184.139.70|25)
 (hosta.sesta.com -- Server ESMTP [iMS V5.0 #8790])
 (TCP|206.184.139.12|5901|206.184.139.70|25)
 smtp;250 2.1.5 bobby@hosta.sesta.com and options OK.

19-Feb-1998 10:52:12.28 1f625.3.1 tcp_local                  D 1
 adam@sesta.com rfc822;carl@hosta.sesta.com carl@hosta.sesta.com
 imta/queue/tcp_local/ZZ01ITRF7BO388000FCN.00
  <01ITRF7BDHS6000FCN@SESTA.COM>
 hosta.sesta.com dns;hosta.sesta.com
 (TCP|206.184.139.12|5901|206.184.139.70|25)
 (hosta.sesta.com -- Server ESMTP [iMS V5.0 #8790])
 (TCP|206.184.139.12|5901|206.184.139.70|25)
 smtp;250 2.1.5 carl@hosta.sesta.com and options OK.

19-Feb-1998 10:52:12.40 1f625.3.2 tcp_local     -            C (9)
 TCP|206.184.139.12|5901|206.184.139.70|25
```

```
 SMTP/hosta.sesta.com/hosta.sesta.com

19-Feb-1998 10:52:13.01 1f625.2.1 tcp_local                  D 1
 adam@sesta.com rfc822;dave@hostb.sesta.com dave@hostb.sesta.com
 imta/queue/tcp_local/ZZ01ITRF7C11FU000FCN.00
  <01ITRF7BDHS6000FCN@SESTA.COM>
 mailhub.sesta.com dns;mailhub.sesta.com
 (TCP|206.184.139.12|5900|206.184.139.66|25)
 (MAILHUB.SESTA.COM -- Server ESMTP [iMS V5.0 #8694])
 (TCP|206.184.139.12|5900|206.184.139.66|25)
 smtp;250 2.1.5 dave@hostb.sesta.com and options OK.

19-Feb-1998 10:52:13.05 1f625.2.2 tcp_local     -            C (10)
 TCP|206.184.139.12|5900|206.184.139.66|25
 SMTP/hostb.sesta.com/mailhub.sesta.com
```

1. The message is enqueued to the first recipient...
2. ....and to the second recipient...
3. ....and to the third recipient.
4. Having `LOG_CONNECTION=3` set causes the MTA to write this entry. The minus, `-`, indicates that this entry refers to an outgoing connection. The `O` means that this entry corresponds to the opening of the connection. Also note that the process id here is the same, 1f625, since the same process is used for the multithreaded TCP/IP channel for these separate connection opens, though this open is being performed by thread 2 vs. thread 3.
5. As there are two separate remote systems to which to connect, the multithreaded SMTP client in separate threads opens up a connection to each—the first in this entry, and the second shown in 7. This part of the entry shows the sending and destination IP numbers and port numbers, and shows both the initial host name, and the host name found by doing a DNS lookup. In the `SMTP/`*initial-host*`/`*dns-host* clauses, note the display of both the initial host name, and that used after performing a DNS MX record lookup on the initial host name: `mailhub.sesta.com` is apparently an MX server for `hostb.sesta.com`.
6. The multithreaded SMTP client opens up a connection to the second system in a separate thread (though the same process).

7. As there are two separate remote systems to which to connect, the multithreaded SMTP client in separate threads opens up a connection to each—the second in this entry, and the first shown above in 5. This part of the entry shows the sending and destination IP numbers and port numbers, and shows both the initial host name, and the host name found by doing a DNS lookup. In this example, the system `hosta.sesta.com` apparently receives mail directly itself.
8. Besides resulting in specific connection entries, `LOG_CONNECTION=3` also causes inclusion of connection related information in the regular message entries, as seen here for instance.
9. Having `LOG_CONNECTION=3` causes the MTA to write this entry. After any messages are dequeued, (the bobby and carl messages in this example), the connection is closed, as indicated by the `C` in this entry.
10. Having `LOG_CONNECTION=3` causes the MTA to write this entry. After any messages are dequeued, (the dave message in this example), the connection is closed, as indicated by the `C` in this entry.

Figure 12-12 illustrates log output for an incoming SMTP message when connection logging is enabled, via `LOG_CONNECTION=3`.

Figure 12-12 Logging: Inbound Connection Logging

```
19-Feb-1998 17:02:08.70 tcp_local    +            O   (1)
 TCP|206.184.139.12|25|192.160.253.66|1244 SMTP       (2)

19-Feb-1998 17:02:26.65 tcp_local    l            E 1
 service@siroe.com rfc822;adam@sesta.com adam
 THOR.SIROE.COM (THOR.SIROE.COM [192.160.253.66]) (3)

19-Feb-1998 17:02:27.05 tcp_local    +            C   (4)
 TCP|206.184.139.12|25|192.160.253.66|1244 SMTP

19-Feb-1998 17:02:31.73 l                         D 1
 service@siroe.com rfc822;adam@sesta.com adam
```

1. The remote system opens a connection. The `O` character indicates that this entry regards the opening of a connection; the `+` character indicates that this entry regards an incoming connection.

2. The IP numbers and ports for the connection are shown. In this entry, the receiving system (the system making the log file entry) has IP address 206.184.139.12 and the connection is being made to port 25; the sending system has IP address 192.160.253.66 and is sending from port 1244.

3. In the entry for the enqueue of the message from the incoming TCP / IP channel (`tcp_local`) to the `l` channel recipient, note that information beyond the default is included since `LOG_CONNECTION=3` is enabled. Specifically, the name that the sending system claimed on its HELO or EHLO line, the sending system's name as found by a DNS reverse lookup on the connection IP number, and the sending system's IP address are all logged; see Chapter 8, "Configuring Channel Definitions," for a discussion of channel keywords affecting this behavior.

4. The inbound connection is closed. The `C` character indicates that this entry regards the closing of a connection; the + character indicates that this entry regards an incoming connection.

Appendix A

SNMP Support

The iPlanet Messaging Server supports system monitoring through the Simple Network Management Protocol (SNMP). Using an SNMP client (sometimes called a *network manager*) such as Sun Net Manager or HP OpenView (not provided with the this product), you can monitor certain parts of the iPlanet Messaging Server.

This chapter describes how to enable SNMP support for the Messaging Server. It also gives an overview of the type of information provided by SNMP. Note that it does not describe how to view this information from an SNMP client. Please refer to your SNMP client documentation for details on how to use it to view SNMP-based information. This document also describes some of the data available from the Messaging Server SNMP implementation, but complete MIB details are available from RFC 2788 and RFC 2789.

This chapter consists of the following sections:

- "SNMP Implementation," on page 349
- "Configuring SNMP Support for the iPlanet Messaging Server on Solaris 8," on page 351
- "Monitoring from an SNMP Client," on page 352
- "Co-existence with Other iPlanet Products on Unix Platforms," on page 353
- "SNMP Information from the Messaging Server," on page 353

SNMP Implementation

The iPlanet Messaging Server implements two standardized 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 for monitoring both 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, source IP addresses of open network connections), while the historical information provides cumulative totals (for example, total messages processed, total inbound connections).

NOTE For a complete listing of Messaging Server SNMP monitoring information, refer to RFC 2788 and RFC 2789.

SNMP is supported on Solaris 8 platforms only. Support on other platforms will appear in a later release. The SNMP support on Solaris makes use of the native Solaris SNMP technology, Solstice Enterprise Agents (SEA). Customers do not need to install SEA on Solaris 8 systems: the necessary run-time libraries are already present.

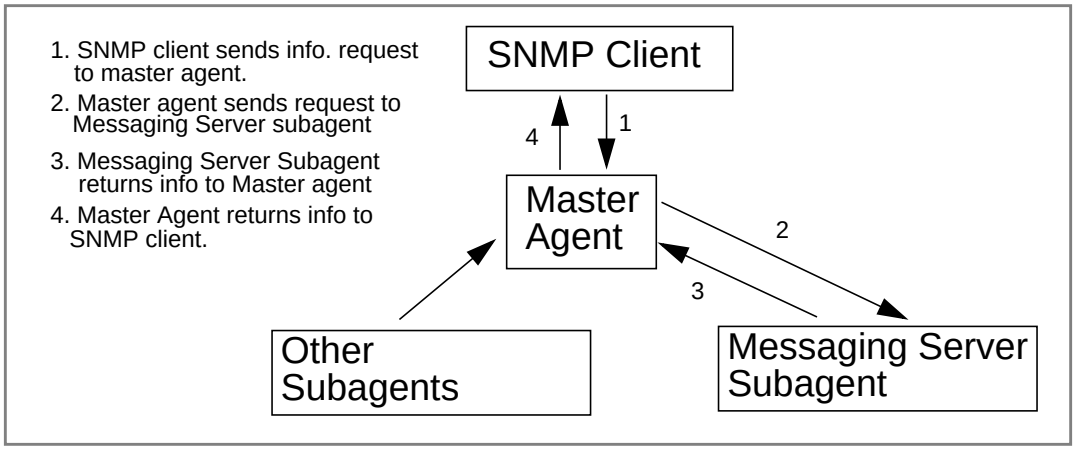
Limitations of the Messaging Server SNMP support are as follows:

- Only one instance of Messaging Server per host computer can be monitored via SNMP.
- The SNMP support is for monitoring only. No SNMP management is supported.
- No SNMP traps are implemented. (RFC 2788 provides similar functionality without using traps.)

SNMP Operation in the Messaging Server

On Solaris platforms, the Messaging Server SNMP process is an SNMP subagent which, upon startup, registers itself with the platform's native SNMP master agent. SNMP requests from clients go to the master agent. The master agent then forwards any requests destined for the Messaging Server to the Messaging Server subagent process. The Messaging Server subagent process then processes the request and relays the response back to the client via the master agent. This process is shown in Figure A-1.

Figure A-1 SNMP Information Flow



Configuring SNMP Support for the iPlanet Messaging Server on Solaris 8

Although the overhead imposed by SNMP monitoring is very small, the Messaging Server nonetheless ships with SNMP support disabled. To enable the SNMP support, run the following commands:

```
# su user-id-for-ims
# configutil -o local.snmp.enable -v 1
# start-msg snmp
```

Once you have enabled SNMP, the `start-msg` command, without any parameters specified, will automatically start the SNMP subagent process along with the other Messaging Server processes.

Note that the Solaris native SNMP master agent must be running in order for the Messaging Server SNMP subagent to operate. The Solaris native SNMP master agent is the `snmpdx` daemon which is normally started as part of the Solaris boot procedure.

The SNMP subagent will automatically select a UDP port on which to listen. Should you require, you can assign a fixed UDP port to the subagent with the following command:

```
# configutil -o local.snmp.port -v port-number
```

You may later undo this setting by specifying a value of zero for the port number. A value of zero, the default setting, tells Messaging Server to allow the subagent to automatically select any available UDP port.

Two SNMP subagent configuration files are placed in the `/etc/snmp/conf` directory: `ims.acl` which contains SNMP access control information, and `ims.reg` which contains SNMP MIB OID registration information.

Normally, there should be no reason to edit either of these files. The MIBs served out by Messaging Server are read-only and there's no need to specify a port number in the `ims.reg` file. If you do specify a port number, then it will be honored unless you also set a port number with the `configutil` utility. In that case, the port number set with `configutil` is the port number which will be used by the subagent. If you do edit the files, then you will need to stop and restart the SNMP subagent in order for your changes to take effect:

```
# stop-msg snmp
# start-msg snmp
```

Monitoring from an SNMP Client

The base OIDs for RFC 2788 and RFC 2789 are

mib-2.27 = 1.3.6.1.2.1.27

mib-2.28 = 1.3.6.1.2.1.28

Point your SNMP client at those two OIDs and access as the "public" SNMP community.

If you wish to load copies of the MIBs into your SNMP client, ASCII copies of the MIBs are located in the <*server-root*>`/plugins/snmp` directory under the file names `rfc2788.mib` and `rfc2789.mib`. For directions on loading those MIBs into your SNMP client software, consult the SNMP client software documentation. The `SnmpAdminString` data type used in those MIBs may not be recognized by some older SNMP clients. In that case, use the equivalent files `rfc2248.mib` and `rfc2249.mib` also found in the same directory.

Co-existence with Other iPlanet Products on Unix Platforms

Other Netscape or iPlanet products which provide SNMP support may do so by displacing the platform's native SNMP master agent. If you will be running such iPlanet products on the same host as Messaging Server and wish to monitor both via SNMP, then configure the iPlanet Proxy SNMP Agent as described in Chapter 7 of *Managing Servers with Netscape Console* (http://docs.iplanet.com/docs/manuals/console/42/html/7_snmp.htm#1024620). This allows the Messaging Server SNMP subagent—a native SNMP subagent—to co-exist with the non-native iPlanet SNMP subagents in the other iPlanet products.

SNMP Information from the Messaging Server

This section summarizes the Messaging Server information provided via SNMP. For detailed information refer to the individual MIB tables in RFC 2788 and RFC 2789. Note that the RFC/MIB terminology refers to the messaging services (MTA, HTTP, etc.) as *applications* (`appl`), Messaging Server network connections as *associations* (`assoc`), and MTA channels as *MTA groups* (`mtaGroups`).

Note that on platforms where more than one instance of Messaging Server may be concurrently monitored, there may then be multiple sets of MTAs and servers in the `applTable`, and multiple MTAs in the other tables.

NOTE The cumulative values reported in the MIBs (e.g., total messages delivered, total IMAP connections, etc.) are reset to zero after a reboot.

Each site will have different thresholds and significant monitoring values. A good SNMP client will allow you to do trend analysis and then send alerts when sudden deviations from historical trends occur.

applTable

The `applTable` provides server information. It is a one-dimensional table with one row for the MTA and an additional row for each of the following servers, if enabled: WebMail HTTP, IMAP, POP, SMTP, and SMTP Submit. This table provides version information, uptime, current operational status (up, down, congested), number of current connections, total accumulated connections, and other related data.

Below is an example of data from `applTable` (mib-2.27.1.1).

`applTable:`

```
    applName.1[1] = mailsrv-1[2] MTA on mailsrv-1.west.sesta.com
    applVersion.1 = 5.1
    applUptime.1 = 7322[3]
    applOperStatus.1 = up[4]
    applLastChange.1 = 7422[3]
    applInboundAssociations.1 = 5
    applOutboundAssociations.1 = 2
    applAccumulatedInboundAssociations.1 = 873
    applAccumulatedOutboundAssociations.1 = 234
    applLastInboundActivity.1 = 1054822[3]
    applLastOutboundActivity.1 = 1054222[3]
    applRejectedInboundAssociations.1 = 0[5]
    applFailedOutboundAssociations.1 = 17
    applDescription.1 = iPlanet Messaging Server 5.1
    applName.2[1] = mailsrv-1 HTTP WebMail server on mailsrv-1.west.sesta.com
    ...
    applName.3 = mailsrv-1 IMAP server on mailsrv-1.west.sesta.com
    ...
    applName.4 = mailsrv-1 POP server on mailsrv-1.west.sesta.com
    ...
    applName.5 = mailsrv-1 SMTP server on mailsrv-1.west.sesta.com
    ...
    applName.6 = mailsrv-1 SMTP Submit server on mailsrv-1.west.sesta.com
    ...
```

Notes:

1. The `.1`, `.2`, etc. suffixes here are the row numbers, `applIndex`. `applIndex` has the value 1 for the MTA, value 2 for the HTTP server, etc. Thus, in this example, the first row of the table provides data on the MTA, the second on the POP server, etc.
2. The name of the Messaging Server instance being monitored. In this example, the instance name is mailsrv-1.

3. These are SNMP TimeStamp values and are the value of `sysUpTime` at the time of the event. `sysUpTime`, in turn, is the count of hundredths of seconds since the SNMP master agent was started.

4. The operational status of the HTTP, IMAP, POP, SMTP, and SMTP Submit servers is determined by actually connecting to them via their configured TCP ports and performing a simple operation using the appropriate protocol (for example, a HEAD request and response for HTTP, a `HELO` command and response for SMTP, and so on). From this connection attempt, the status—`up` (1), `down` (2), or `congested` (4)—of each server is determined.

   Note that these probes appear as normal inbound connections to the servers and contribute to the value of the `applAccumulatedInboundAssociations` MIB variable for each server.

   For the MTA, the operational status is taken to be that of the Job Controller. If the MTA is shown to be up, then the Job Controller is up. If the MTA is shown to be down, then the Job Controller is down. This MTA operational status is independent of the status of the MTA's Service Dispatcher. The operational status for the MTA only takes on the value of up or down. Although the Job Controller does have a concept of "congested," it is not indicated in the MTA status.

5. For the HTTP, IMAP, and POP servers the `applRejectedInboundAssociations` MIB variable indicates the number of failed login attempts and not the number of rejected inbound connection attempts.

applTable Usage

Monitoring server status (`applOperStatus`) for each of the listed applications is key to monitoring each server.

If it's been a long time since the MTA last inbound activity as indicated by `applLastInboundActivity`, then something may be broken preventing connections. If `applOperStatus=2` (down), then the monitored service is down. If `applOperStatus=1` (up), then the problem may be elsewhere.

assocTable

This table provides network connection information to the MTA. It is a two-dimensional table providing information about each active network connection. Connection information is not provided for other servers.

Below is an example of data from applTable (mib-2.27.2.1).

```
assocTable:

    assocRemoteApplication.1.1[1] = 129.146.198.167[2]
    assocApplicationProtocol.1.1[1] = applTCPProtoID.25[3]
    assocApplicationType.1.1 = peerinitiator(3)[4]
    assocDuration.1.1 = 400[5]
...
```

Notes:

1. In the `.x.y` suffix, `x` is the application index, `applIndex`, and indicates which application in the `applTable` is being reported on. In this case, the MTA. The `y` serves to enumerate each of the connections for the application being reported on.
2. The source IP address of the remote SMTP client.
3. This is an OID indicating the protocol being used over the network connection. `aplTCPProtoID` indicates the TCP protocol. The `.n` suffix indicates the TCP port in use and `.25` indicates SMTP which is the protocol spoken over TCP port 25.
4. It is not possible to know if the remote SMTP client is a user agent (UA) or another MTA. As such, the subagent always reports `peer-initiator`; `ua-initiator` is never reported.
5. This is an SNMP `TimeInterval` and has units of hundredths of seconds. In this example, the connection has been open for 4 seconds.

assocTable Usage

This table is used to diagnose active problems. For example, if you suddenly have 200,000 inbound connections, this table can let you know where they are coming from.

mtaTable

This is a one-dimensional table with one row for each MTA in the `applTable`. Each row gives totals across all channels (referred to as groups) in that MTA for select variables from the `mtaGroupTable`.

Below is an example of data from `applTable` (mib-2.28.1.1).

```
mtaTable:

    mtaReceivedMessages.1[1] = 172778
    mtaStoredMessages.1 = 19
    mtaTransmittedMessages.1 = 172815
    mtaReceivedVolume.1 = 3817744
    mtaStoredVolume.1 = 34
    mtaTransmittedVolume.1 = 3791155
    mtaReceivedRecipients.1 = 190055
    mtaStoredRecipients.1 = 21
    mtaTransmittedRecipients.1 = 3791134
    mtaSuccessfulConvertedMessages.1 = 0[2]
    mtaFailedConvertedMessages.1 = 0
    mtaLoopsDetected.1 = 0[3]
```

Notes:

1. The `.x` suffix provides the row number for this application in the `applTable`. In this example, `.1` indicates this data is for the first application in the `applTable`. Thus, this is data on the MTA.
2. Only takes on non-zero values for the conversion channel.
3. Counts the number of `.HELD` message files currently stored in the MTA's message queues.

mtaTable Usage

If `mtaLoopsDetected` is not zero, then there is a looping mail problem. Locate and diagnose the `.HELD` files in the MTA queue to resolve the problem.

If the system does virus scanning with a conversion channel and rejects infected messages, then `mtaSuccessfulConvertedMessages` will give a count of infected messages in addition to other conversion failures.

mtaGroupTable

This two-dimensional table provides channel information for each MTA in the `applTable`. This information includes such data as counts of stored (that is, queued) and delivered mail messages. Monitoring the count of stored messages, `mtaGroupStoredMessages`, for each channel is critical: when the value becomes abnormally large, mail is backing up in your queues.

Below is an example of data from `mtaGroupTable` (mib-2.28.2.1).

```
mtaGroupTable:

mtaGroupName.1.1[1] = autoreply[2]
        ...
mtaGroupName.1.2[1] = ims-ms
        ...
mtaGroupName.1.3[1] = tcp_local
    mtaGroupDescription.1.3 = mailsrv-1 MTA tcp_local channel
    mtaGroupReceivedMessages.1.3 = 12154
    mtaGroupRejectedMessages.1.3 = 0
    mtaGroupStoredMessages.1.3 = 2
    mtaGroupTransmittedMessages.1.3 = 12148
    mtaGroupReceivedVolume.1.3 = 622135
    mtaGroupStoredVolume.1.3 = 7
    mtaGroupTransmittedVolume.1.3 = 619853
    mtaGroupReceivedRecipients.1.3 = 33087
    mtaGroupStoredRecipients.1.3 = 2
    mtaGroupTransmittedRecipients.1.3 = 32817
    mtaGroupOldestMessageStored.1.3 = 1103
    mtaGroupInboundAssociations.1.3 = 5
    mtaGroupOutboundAssociations.1.3 = 2
    mtaGroupAccumulatedInboundAssociations.1.3 = 150262
    mtaGroupAccumulatedOutboundAssociations.1.3 = 10970
    mtaGroupLastInboundActivity.1.3 = 1054822
    mtaGroupLastOutboundActivity.1.3 = 1054222
    mtaGroupRejectedInboundAssociations.1.3 = 0
    mtaGroupFailedOutboundAssociations.1.3 = 0
    mtaGroupInboundRejectionReason.1.3 =
    mtaGroupOutboundConnectFailureReason.1.3 =
    mtaGroupScheduledRetry.1.3 = 0
    mtaGroupMailProtocol.1.3 = applTCPProtoID.25
    mtaGroupSuccessfulConvertedMessages.1.3 = 0[3]
    mtaGroupFailedConvertedMessages.1.3 = 0
    mtaGroupCreationTime.1.3 = 0
    mtaGroupHierarchy.1.3 = 0
    mtaGroupOldestMessageId.1.3 = <01IFBV8AT8HYB4T6UA@red.iplanet.com>
    mtaGroupLoopsDetected.1.3 = 0[4]
    mtaGroupLastOutboundAssociationAttempt.1.3 = 1054222
```

Notes:

1. In the `.x.y` suffix, `x` is the application index, `applIndex`, and indicates which application in the `applTable` is being reported on. In this case, the MTA. The `y` serves to enumerate each of the channels in the MTA. This enumeration index, `mtaGroupIndex`, is also used in the `mtaGroupAssociationTable` and `mtaGroupErrorTable` tables.
2. The name of the channel being reported on. In this case, the autoreply channel.
3. Only takes on non-zero values for the conversion channel.

4. Counts the number of `.HELD` message files currently stored in this channel's message queue.

mtaGroupTable Usage

Trend analysis on `*Rejected*` and `*Failed*` might be useful in determining potential channel problems.

A sudden jump in the ratio of `mtaGroupStoredVolume` to `mtaGroupStoredMessages` could mean that a large junk mail is bouncing around the queues.

A large jump in `mtaGroupStoredMessages` could indicate unsolicited bulk email is being sent or that delivery is failing for some reason.

If the value of `mtaGroupOldestMessageStored` is greater than the value used for the undeliverable message notification times (`notices` channel keyword) this may indicate a message which cannot be processed even by bounce processing. Note that bounces are done nightly so you will want to use `mtaGroupOldestMessageStored` > (maximum age + 24 hours) as the test.

If `mtaGroupLoopsDetected` is greater than 0, a mail loop has been detected.

mtaGroupAssociationTable

This is a three-dimensional table whose entries are indices into the `assocTable`. For each MTA in the `applTable`, there is a two-dimensional sub-table. This two-dimensional sub-table has a row for each channel in the corresponding MTA. For each channel, there is an entry for each active network connection which that channel has currently underway. The value of the entry is the index into the `assocTable` (as indexed by the entry's value and the `applIndex` index of the MTA being looked at). This indicated entry in the `assocTable` is a network connection held by the channel.

In simple terms, the `mtaGroupAssociationTable` table correlates the network connections shown in the `assocTable` with the responsible channels in the `mtaGroupTable`.

Below is an example of data from `mtaGroupAssociationTable` (mib-2.28.3.1).

```
mtaGroupAssociationTable:

    mtaGroupAssociationIndex.1.3.1[1] = 1[2]
    mtaGroupAssociationIndex.1.3.2 = 2
    mtaGroupAssociationIndex.1.3.3 = 3
    mtaGroupAssociationIndex.1.3.4 = 4
```

```
    mtaGroupAssociationIndex.1.3.5 = 5
    mtaGroupAssociationIndex.1.3.6 = 6
    mtaGroupAssociationIndex.1.3.7 = 7
```

Notes:

1. In the `.x.y.z` suffix, `x` is the application index, `applIndex`, and indicates which application in the `applTable` is being reported on. In this case, the MTA. The `y` indicates which channel of the `mtaGroupTable` is being reported on. In this example, 3 indicates the tcp_local channel. The `z` serves to enumerate the associations open to or from the channel.
2. The value here is an index into the `assocTable`. Specifically, `x` and this value become, respectively, the values of the `applIndex` and `assocIndex` indices into the `assocTable`. Or, put differently, this is saying that (ignoring the `applIndex`) the first row of the `assocTable` describes a network connection controlled by the tcp_local channel.

mtaGroupErrorTable

This is another three-dimensional table which gives the counts of temporary and permanent errors encountered by each channel of each MTA while attempting delivery of messages. Entries with index values of 4000000 are temporary errors while those with indices of 5000000 are permanent errors. Temporary errors result in the message being re-queued for later delivery attempts; permanent errors result in either the message being rejected or otherwise returned as undeliverable.

Below is an example of data from `mtaGroupErrorTable` (mib-2.28.5.1).

```
mtaGroupErrorTable:

    mtaGroupInboundErrorCount.1.1.4000000[1] = 0
    mtaGroupInboundErrorCount.1.1.5000000 = 0
    mtaGroupInternalErrorCount.1.1.4000000 = 0
    mtaGroupInternalErrorCount.1.1.5000000 = 0
    mtaGroupOutboundErrorCount.1.1.4000000 = 0
    mtaGroupOutboundErrorCount.1.1.5000000 = 0

    mtaGroupInboundErrorCount.1.2.4000000[1] = 0
    ...

    mtaGroupInboundErrorCount.1.3.4000000[1] = 0
    ...
```

Notes:

1. In the `.x.y.z` suffix, `x` is the application index, `applIndex`, and indicates which application in the `applTable` is being reported on. In this case, the MTA. The `y` indicates which channel of the `mtaGroupTable` is being reported on. In this example, 1 specifies the autoreply channel, 2 the ims-ms channel, and 3 the tcp_local channel. Finally, the `z` is either 4000000 or 5000000 and indicates, respectively, counts of temporary and permanent errors encountered while attempting message deliveries for that channel.

mtaGroupErrorTable Usage

A large jump in error count may likely indicate an abnormal delivery problem. For instance, a large jump for a tcp_ channel may indicate a DNS or network problem. A large jump for the ims_ms channel may indicate a delivery problem to the message store (for example, a partition is full, `stored` problem, and so on).

Glossary

A record A type of DNS record containing a host name and its associated IP address. A records are used by messaging servers on the Internet to route email. *See also* **Domain Name System (DNS)** and **MX record**.

access control A method for controlling access to a server or to folders and files on a server.

access control information (ACI) A single item of information from an access control list.

access control list (ACL) A set of data associated with a directory that defines the permissions that users and / or groups have for accessing it.

access control rules Rules specifying user permissions for a given set of directory entries or attributes.

access domain Limits access to certain Messaging Server operations from within a specified domain. For example, an access domain can be used to limit where mail for an account can be collected.

account Information that defines a specific user or user group. This information includes the user or group name, valid email address or addresses, and how and where email is delivered.

address Information in an email message that determines where and how the message must be sent. Addresses are found both in message headers and in message envelopes. Envelope addresses determine how the message gets routed and delivered; header addresses are present merely for display purposes.

address handling The actions performed by the MTA to detect errors in addressing, to rewrite addresses if necessary, and to match addresses to recipients.

addressing protocol The addressing rules that make email possible. RFC 822 is the most widely used protocol on the Internet and the protocol supported by iPlanet Messaging Server. Other protocols include X.400 and UUCP (UNIX to UNIX Copy Protocol).

address token The address element of a rewrite rule pattern.

admin Administrator or administrative.

administration console See **Console**.

administration privileges The set of privileges that define a users administrative role.

administration server administrator User who has administrative privileges to start or stop a server even when there is no Directory Server connection. The administration server administrator has restricted server tasks (typically only Restart Server and Stop Server) for all servers in a local server group. When an administration server is installed, this administrator's entry is automatically created locally (this administrator is not a user in the user directory).

administrator A user with a defined set of administrative privileges. See also **configuration administrator**, **Directory Manager**, **administration server administrator**, **server administrator**, **message store administrator**, **top-level administrator**, **domain administrator**, **organization administrator**, **family group administrator**, **mailing list owner**.

alias An alternate name of an email address.

alias file A file used to set aliases not set in a directory, such as the postmaster alias.

Allow filter A Messaging Server access-control rule that identifies clients that are to be allowed access to one or more of the following services: POP, IMAP, or HTTP. Compare **Deny filter**.

alternate address A secondary address for an account, generally a variation on the primary address. In some cases it is convenient to have more than one address for a single account.

APOP Authenticated Post Office Protocol. Similar to the Post Office Protocol (POP), but instead of using a plaintext password for authentication, it uses an encoding of the password together with a challenge string.

AUTH An SMTP command enabling an SMTP client to specify an authentication method to the server, perform an authentication protocol exchange, and, if necessary, negotiate a security layer for subsequent protocol interactions.

authentication (1) The process of proving the identity of a client user to iPlanet Messaging Server. (2) The process of proving the identity of iPlanet Messaging Server to a client or another server.

authentication certificate A digital file sent from server to client or client to server to verify and authenticate the other party. The certificate ensures the authenticity of its holder (the client or server). Certificates are not transferable.

autoreply option file A file used for setting options for autoreply, such as vacation notices.

AutoReply utility A utility that automatically responds to messages sent to accounts with the AutoReply feature activated. Every account in iPlanet Messaging Server can be configured to automatically reply to incoming messages.

backbone The primary connectivity mechanism of a distributed system. All systems that have connectivity to an intermediate system on the backbone are connected to each other. This does not prevent you from setting up systems to bypass the backbone for reasons of cost, performance, or security.

backend server An email server whose only function is to store and retrieve email messages. Also called a message store server.

backup The process of backing up the contents of folders from the message store to a backup device. See also **restore**.

banner A text string displayed by a service such as IMAP when a client first connects to it.

base DN A distinguished name entry in the directory from which searches will occur. Also known as a search base. For example, ou=people, o=siroe.com.

Berkeley DB A transactional database store intended for high-concurrency read-write workloads, and for applications that require transactions and recoverability. iPlanet Messaging Server uses Berkeley databases for numerous purposes.

bind DN A distinguished name used to authenticate to the Directory Server when performing an operation.

body One part of an email message. Although headers and envelopes must follow a standard format, the body of the message has a content determined by the sender—the body can contain text, graphics, or even multimedia. Structured bodies follow the MIME standard.

capability A string, provided to clients, that defines the functionality available in a given IMAP service.

CA Certificate Authority. An organization that issues digital certificates (digital identification) and makes its public key widely available to its intended audience.

Certificate Authority See **CA**.

certificate-based authentication Identification of a user from a digital certificate submitted by the client. Compare **password authentication**.

certificate database A file that contains a server's digital certificate(s). Also called a certificate file.

certificate name The name that identifies a certificate and its owner.

channel The fundamental MTA component that processes a message. A channel represents a connection with another computer system or group of systems. Each channel consists of one or more channel programs and an outgoing message queue for storing messages that are destined to be sent to one or more of the systems associated with the channel. See also **channel block**, **channel host table, channel program**.

channel block A single channel definition. See also channel host table.

channel host table The collective set of channel definitions.

channel program Part of a channel that performs the following functions: (1) transmits messages to remote systems and deletes messages from the queue after they are sent and (2) accepts messages from remote systems placing them in the appropriate channel queues. See also **master channel program**, **slave channel program**.

cipher An algorithm used in encryption.

ciphertext Text that has been encrypted. Opposite of **cleartext**.

CLI Command Line Interface.

client A software entity that requests services or information from a server.

CNAME record A type of DNS record that maps a domain name alias to a domain name.

cleartext Unencrypted text.

client-server model A computing model in which networked computers provide specific services to other client computers. Examples include the name-server / name-resolver paradigm of the DNS and file-server / file-client relationships such as NFS and diskless hosts.

cn LDAP alias for common name.

comment character A character that, when placed at the beginning of a line, turns the line into a nonexecutable comment.

config Configuration.

configuration administrator Person who has administrative privileges to manage servers and configuration directory data in the entire iPlanet topology. The configuration administrator has unrestricted access to all resources in the iPlanet topology. This is the only administrator who can assign server access to other administrators. The configuration administrator initially manages administrative configuration until the administrators group and its members are in place.

Configuration Directory Server A Directory Server that maintains configuration information for a server or set of servers.

configuration file A file that contains the configuration parameters for a specific component of the iPlanet Messaging system.

configutil A command-line utility for making changes to various configuration parameters stored in the directory server or in the local configuration file, configdb.

congestion thresholds A disk space limit that can be set by the system administrator that prevents the database from becoming overloaded by restricting new operations when system resources are insufficient.

Console A GUI (graphical user interface) that enables you to configure, monitor, maintain, and troubleshoot many iPlanet components.

cookie Text-only strings entered into the browser's memory automatically when you visit specific web sites. Cookies are programmed by the web page author. Users can either accept or deny cookies. Accepting the cookies allows the web page to load more quickly and is not a threat to the security of your machine.

counterutil A command-line utility for displaying all counters in a counter object.

CRAM-MD5 A lightweight standards track authentication mechanism documented in RFC 2195. It provides a fast (albeit somewhat weaker) alternative to TLS (SSL) when only the user's login password needs to be protected from network eavesdroppers.

cronjob UNIX only. A task that is executed automatically by the cron daemon at a configured time. See **crontab file**.

crontab file UNIX only. A list of commands, one per line, that executes automatically at a given time.

daemon A UNIX program that runs in the background, independent of a terminal, and performs a function whenever necessary. Common examples of daemon programs are mail handlers, license servers, and print daemons. On Windows NT machines, this type of program is called a service. See also **service**.

data store A store that contains directory information, typically for an entire directory information tree.

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 would be cn=billbob,dc=bridge,dc=net,o=internet.

defragmentation The Multipurpose Internet Mail Extensions (MIME) feature that enables a large message that has been broken down into smaller messages or fragments to be reassembled. A Message Partial Content-Type header field that appears in each of the fragments contains information that helps reassemble the fragments into one message. See also **fragmentation**.

Delegated Administrator Console A web browser-based software console that allows domain administrators to add and modify users and groups to a hosted domain. Also allows end users to change their password, set message forwarding rules, set vacation rules, and list distribution list subscriptions.

Delegated Administrator for Messaging A set of interfaces (GUI and CLI) that allow domain administrators to add and modify users and groups to a hosted domain.

delegated administrator server A daemon program that handles access control to the directory by hosted domains.

delete message The act of marking a message for deletion. The deleted message is not removed from the message store until it is expunged or purged in a separate action by the user. See also **purge message**, **expunge message**.

deliver A command-line utility that delivers mail directly to the message store accessible by POP, IMAP, or HTTP mail clients.

delivery See **message delivery**.

delivery status notification A message giving status information about a message in route to a recipient. For example, a message indicating that delivery has been delayed because of network outages.

denial of service attack A situation where an individual intentionally or inadvertently overwhelms your mail server by flooding it with messages. Your server's throughput could be significantly impacted or the server itself could become overloaded and nonfunctional.

Deny filter A Messaging Server access-control rule that identifies clients that are to be denied access to one or more of the following services: POP, IMAP, or HTTP. Compare **Allow filter**.

dereferencing an alias Specifying, in a bind or search operation, that a directory service translate an alias distinguished name to the actual distinguished name of an entry.

DIGEST-MD5 A lightweight standards track authentication mechanism that is more secure then CRAM-MD5. Documented in RFC 2831 which also provides an option to protect the entire connection without the setup overhead of TLS (SSL).

directory context The point in the directory tree information at which a search begins for entries used to authenticate a user and password for message store access. See also **base DN**.

directory entry A set of directory attributes and their values identified by its distinguished name. Each entry contains an object class attribute that specifies the kind of object the entry describes and defines the set of attributes it contains.

directory information tree The tree-like hierarchical structure in which directory entries are organized. Also called a DIT. DITs can be organized along the DNS (DC Trees) or Open Systems Interconnect networks (OSI trees).

directory lookup The process of searching the directory for information on a given user or resource, based on that user or resource's name or other characteristic.

Directory Manager User who has administrative privileges to the directory server database. Access control does not apply this user (think of the directory manager as the directory's superuser).

directory schema The set of rules that defines the data that can be stored in the directory.

Directory Server The iPlanet directory service based on LDAP. See also **directory service**, **Lightweight Directory Access Protocol**, **Configuration Directory Server**, **User/Groups Directory Server**.

directory service A logically centralized repository of information about people and resources within an organization. See also **Lightweight Directory Access Protocol**.

directory synchronization The process of updating—that is, synchronizing—the MTA directory cache with the current directory information stored in the directory service. See also **MTA directory cache**.

disconnected state The mail client connects to the server, makes a cache copy of selected messages, then disconnects from the server.

Dispatcher The MTA component that handles connection requests for defined TCP ports. The Dispatcher is a multi-threaded connection dispatching agent that permits multiple multi-threaded servers to share responsibility for a given service. When using the Dispatcher, it is possible to have several multi-threaded SMTP server processes running concurrently.

distinguished name The comma-separated sequence of attributes and values that specify the unique location of an entry within the directory information tree. Often abbreviated as DN.

distribution list A list of email addresses (users) that can be sent a message by specifying one email address. Also called a mailing list or group. See also **expansion**, **member**, **moderator**, and **alias**.

distribution list owner An individual who is responsible for a distribution list. An owner can add or delete distribution list members. See also **distribution list**, **expansion**, **member**, and **moderator**.

DIT See **directory information tree**.

DN See distinguished name.

dn LDAP alias for distinguished name. See also **distinguished name**.

DNS See **Domain Name System**.

DNS alias A host name that the DNS server recognizes as pointing to a different host—specifically a DNS CNAME record. Machines always have one real name, but they can have one or more aliases. For example, www.siroe.domain might be an alias that points to a real machine called realthing.siroe.domain where the server currently exists.

DNS database A database of domain names (host names) and their corresponding IP addresses.

DNS spoofing A form of network attack in which a DNS server has been subverted to provide false information.

domain 1) A group of computers whose host names share a common suffix, the domain name. Syntactically, an Internet domain name consists of a sequence of names (labels) separated by periods (dots), for example, corp.mktng.siroe.com. 2) A region of administrative control.

domain administrator User who has administrative privileges to create, modify, and delete mail users, mailing lists, and family accounts in a hosted domain by using the Delegated Administrator for Messaging GUI or CLIs. By default, this user can act as a message store administrator for all messaging servers in the topology.

domain alias A domain entry that points to another domain. By using aliases, hosted domains can have several domain names.

domain hosting The ability to host one or more domains on a shared messaging server. For example, the domains siroe.com and sesta.org might both be hosted on the siroe.net mail server. Users send mail to and receive mail from the hosted domain—the name of the mail server does not appear in the email address.

domain name (1) A host name used in an email address. (2) A unique name that defines an administrative organization. Domains can contain other domains. Domain names are interpreted from right to left. For example, siroe.com is both the domain name of the Siroe Company and a subdomain of the top-level com domain. The siroe.com domain can be further divided into subdomains such as corp.siroe.com, and so on. See also **host name** and **fully-qualified domain name**.

Domain Name System (DNS) 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 into Internet addresses. See also **A record**, **MX record**, **CNAME record**.

domain organization A sub-domain below a hosted domain in the Organization Tree. Domain organizations are useful for companies that wish to organize their user and group entries along departmental lines.

domain part The part of an email address to the right of the @ sign. For example, siroe.com is the domain part of the email address dan@siroe.com.

domain quota The amount of space, configured by the system administrator, allocated to a domain for email messages.

domain rewrite rules See **rewrite rules**.

domain template The part of a rewrite rule that defines how the host / domain portion of an address is rewritten. It can include either a full static host / domain address or a single field substitution string, or both.

DSN. See **Delivery Status Notification**.

dsservd A daemon that accesses the database files that hold the directory information, and communicates with directory clients using the LDAP protocol.

dssetup A Directory Server preparation tool that makes an existing Directory Server ready for use by an iPlanet Messaging Server.

dynamic group A mail group defined by an LDAP search URL. Users usually join the group by setting an LDAP attribute in their directory entry.

EHLO command An SMTP command that queries a server to find out if the server supports extended SMTP commands. Defined in RFC 1869.

encryption The process of disguising information so that it cannot be deciphered (decrypted) by anyone but the intended recipient who has the code key.

enterprise network A network that consists of collections of networks connected to each other over a geographically dispersed area. The enterprise network serves the needs of a widely distributed company and is used by the company's mission-critical applications.

envelope A container for transport information about the sender and the recipient of an email message. This information is not part of the message header. Envelopes are used by various email programs as messages are moved from place to place. Users see only the header and body of a message.

envelope field A named item of information, such as RCPT TO, in a message envelope.

error handler A program that handles errors. In Messaging Server, issues error messages and processes error action forms after the postmaster fills them out.

Error-Handler Action form A form sent to the postmaster account that accompanies a received message that Messaging Server cannot handle. The postmaster fills out the form to instruct the server how to process the message.

error message A message reporting an error or other situation. iPlanet Messaging Server generates messages in a number of situations, notably when it gets an email message that it can't handle. Others messages, called notification errors, are for informational purposes only.

ESMTP See **Extended Simple Mail Transfer Protocol**.

ESP Enterprise Service Provider.

ETRN An SMTP command enabling a client to request that the server start the processing of its mail queues for messages that are waiting at the server for the client machine. Defined in RFC 1985.

expander Part of an electronic mail delivery system that allows a message to be delivered to a list of addressees. Mail expanders are used to implement mailing lists. Users send messages to a single address (e.g., hacks@somehost.edu) and the mail expander takes care of delivery to the mailboxes in the list. Also called mail exploders. See also **EXPN**.

expansion This term applies to the MTA processing of distribution lists. The act of converting a message addressed to a distribution list into enough copies for each distribution list member.

EXPN An SMTP command for expanding a mailing list. Defined in RFC 821.

expunge message The act of marking a message for deletion and then permanently removing it from the INBOX. See also **delete message**, **purge message**.

Extended Simple Mail Transfer Protocol (ESMTP) 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.

extranet The part of a company intranet that customers and suppliers can access. See also **intranet**.

facility In a Messaging Server log-file entry, a designation of the software subsystem (such as Network or Account) that generated the log entry.

failover The automatic transfer of a computer service from one system to another to provide redundant backup.

family group administrator User who has administrative privileges to add and remove family members in a family group. This user can grant family group administrative access to other members of group.

firewall A network configuration, usually both hardware and software, that forms a barrier between networked computers within an organization and those outside the organization. A firewall is commonly used to protect information such as a network's email, discussion groups, and data files within a physical building or organization site.

folder A named collection of messages. Folders can contain other folders. Also called a mailbox. See also **personal folder**, **shared folder, INBOX**.

forwarding See **message forwarding**.

FQDN See **fully-qualified domain name**.

fragmentation The Multipurpose Internet Mail Extensions (MIME) feature that allows the breaking up of a large message into smaller messages. See also **defragmentation**.

fully-qualified domain name (FQDN) The unique name that identifies a specific Internet host. See also **domain name**.

gateway The terms gateway and application gateway refer to systems that do translation from one native format to another. Examples include X.400 to/from RFC 822 electronic mail gateways. A machine that connects two or more electronic mail systems (especially dissimilar mail systems on two different networks) and transfers messages between them. Sometimes the mapping and translation can be complex, and it generally requires a store-and-forward scheme whereby the message is received from one system completely before it is transmitted to the next system after suitable translations.

greeting form A message usually sent to users when an account is created for them. This form acts as confirmation of the new account and verification of its contents.

group A group of LDAP mail entries that are organized under a distinguished name. Usually used as a distribution list, but may also be used to grant certain administrative privileges to members of the group. See also **dynamic group**, **static group**.

group folders These contain folders for shared and group folders. See **shared folder**.

GUI Graphical User Interface

HA See **High Availability**.

hashdir A command-line utility for determining which directory contains the message store for a particular user.

header The portion of an email message that precedes the body of the message. The header is composed of field names followed by a colon and then values. Headers contain information useful to email programs and to users trying to make sense of the message. For example, headers include delivery information, summaries of contents, tracing, and MIME information; they tell whom the message is for, who sent it, when it was sent, and what it is about. Headers must be written according to RFC 822 so that email programs can read them.

header field A named item of information, such as From: or To:, in a message header. Often referred to as a "header line".

High Availability Enables the detection of a service interruption and provides recovery mechanisms in the event of a system failure or process fault. In addition, it allows a backup system to take over the services in the event of a primary system failure.

hop A transmission between two computers.

host The machine on which one or more servers reside.

hosted domain An email domain that is outsourced by an ISP. That is, the ISP provides email domain hosting for an organization by operating and maintaining the email services for that organization. A hosted domain shares the same Messaging Server host with other hosted domains. In earlier LDAP-based email systems, a domain was supported by one or more email server hosts. With Messaging Server, many domains can be hosted on a single server. For each hosted domain, there is an LDAP entry that points to the user and group container for the domain. Hosted domains are also called virtual hosted domains or virtual domains.

host name The name of a particular machine within a domain. The host name is the IP host name, which might be either a "short-form" host name (for example, mail) or a fully qualified host name. The fully qualified host name consists of two parts: the host name and the domain name. For example, mail.siroe.com is the machine mail in the domain siroe.com. Host names must be unique within their domains. Your organization can have multiple machines named mail, as long as the machines reside in different subdomains; for example, mail.corp.siroe.com and mail.field.siroe.com. Host names always map to a specific IP address. See also **domain name**, **fully-qualified domain name**, and **IP address**.

host name hiding The practice of having domain-based email addresses that don't contain the name of a particular internal host.

HTTP See **HyperText Transfer Protocol**.

hub A host that acts as the single point of contact for the system. When two networks are separated by a firewall, for example, the firewall computer often acts as a mail hub.

HyperText Transfer Protocol A standard protocol that allows the transfer of hypertext documents over the Web. iPlanet Messaging Server provides an HTTP service to support web-based email. See **Messenger Express**.

iDA iPlanet Delegated Administrator for Messaging.

IDENT See **Identification Protocol**.

Identification Protocol A protocol that provides a means to determine the identity of a remote process responsible for the remote end of a particular TCP connection. Defined in RFC 1413.

IMAP4 See **Internet Message Access Protocol Version 4**.

imsadmin A set of command line utilities for managing domain administrators, users, and groups.

imsasm A utility that handles the saving and recovering of user mailboxes. The imsasm utility invokes the imsbackup and imsrestore utilities to create and interpret a data stream.

imsbackup A command-line utility for backing up the message store.

imscripter A command-line utility that talks to an IMAP server. You can use this utility to execute a command or batch of commands on IMAP folders.

imsimta commands A set of command line utilities for performing various maintenance, testing, and management tasks for the Message Transfer Agent (MTA).

imsrestore A command-line utility for restoring the message store.

INBOX The name reserved for a user's default mailbox for mail delivery. INBOX is the only folder name that is case-insensitive. For example: INBOX, Inbox, and inbox are all valid names for a users default mailbox.

installation directory The directory into which the binary (executable) files of a server are installed. For the Messaging Server, it is a subdirectory of the server root: *ServerRoot*/bin/msg/. Compare **instance directory**, **server root**.

instance A separately executable configuration of a server or other software entity on a given host. With a single installed set of binary files, it is possible to create multiple instances of iPlanet servers that can be run and accessed independently of each other.

instance directory The directory that contains the files that define a specific instance of a server. For the Messaging Server, it is a subdirectory of the server root: *ServerRoot*/msg-*InstanceName*/, where *InstanceName* is the name of the server as specified at installation. Compare **installation directory**, **server root**.

Internet The name given to the worldwide network of networks that uses TCP/IP protocols.

Internet Message Access Protocol Version 4 (IMAP4) 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 (IP) The basic network-layer protocol on which the Internet and intranets are based.

internet protocol address See **IP address**.

intranet A network of TCP/IP networks within a company or organization. Intranets enable companies to employ the same types of servers and client software used for the World Wide Web for internal applications distributed over the corporate LAN. Sensitive information on an intranet that communicates with the Internet is usually protected by a firewall. See also **firewall** and **extranet**.

invalid user An error condition that occurs during message handling. When this occurs, the message store sends a communication to the MTA, the message store deletes its copy of the message. The MTA bounces the message back to the sender and deletes its copy of the message.

IP See **Internet Protocol**.

IP address A set of numbers, separated by dots, such as 198.93.93.10, that specifies the actual location of a machine on an intranet or the Internet. A 32-bit address assigned to hosts using TCP/IP.

iPlanet Setup The installation program for all iPlanet servers and for iPlanet Console.

ISP 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.

Job Controller The MTA component responsible for scheduling and executing tasks upon request by various other MTA components.

key database A file that contains the key pair(s) for a server's certificate(s). Also called a key file.

knowledge information Part of the directory service infrastructure information. The directory server uses knowledge information to pass requests for information to other servers.

LDAP See **Lightweight Directory Access Protocol**.

LDAP Data Interchange Format (LDIF) The format used to represent Directory Server entries in text form.

LDAP filter A way of specifying a set of entries, based on the presence of a particular attribute or attribute value.

LDAP referrals An LDAP entry that consists of a symbolic link (referral) to another LDAP entry. An LDAP referral consists of an LDAP host and a distinguished name. LDAP referrals are often used to reference existing LDAP data so that this data does not have to be replicated. They are also used to maintain compatibility for programs that depend on a particular entry that may have been moved.

LDAP search string A string with replaceable parameters that defines the attributes used for directory searches. For example, an LDAP search string of "uid=%s" means that searches are based on the user ID attribute.

LDAP Server A software server that maintains an LDAP directory and services queries to the directory. The iPlanet Directory Services are implementations of an LDAP Server.

LDAP server failover A backup feature for LDAP servers. If one LDAP server fails, the system can switch over to another LDAP server.

LDBM LDAP Data Base Manager.

LDIF See **LDAP Data Interchange Format**.

Legato Networker A third-party backup utility distributed by Legato.

level A designation of logging verbosity, meaning the relative number of types of events that are recorded in log files. At a level of Emergency, for example, very few events are logged; at a level of Informational, on the other hand, very many events are logged.

Lightweight Directory Access Protocol (LDAP) 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, distribution lists, and configuration data across iPlanet servers. The iPlanet Directory Server uses the LDAP protocol.

listen port The port that a server uses to communicate with clients and other servers.

local part The part of an email address that identifies the recipient. See also **domain part**.

localizing, localization Refers to the process of translating

log directory The directory in which all of a service's log files are kept.

log expiration Deletion of a log file from the log directory after it has reached its maximum permitted age.

log rotation Creation of a new log file to be the current log file. All subsequent logged events are to be written to the new current file. The log file that was the previous current file is no longer written to, but remains in the log directory.

lookup Same as a search, using the specified parameters for sorting data.

mailbox A place where messages are stored and viewed. See **folder**.

mail client The programs that help users send and receive email. This is the part of the various networks and mail programs that users have the most contact with. Mail clients create and submit messages for delivery, check for new incoming mail, and accept and organize incoming mail.

mail exchange record See **MX record**.

mailing list A list of email addresses to which a message can be sent by way of a mailing list address. Sometimes called a group.

mailing list owner A user who has administrative privileges to add members to and delete members from the mailing list.

mail relay A mail server that accepts mail from a MUA or MTA and relays it to the mail recipient's message store or another router.

mail router See **mail relay**.

managed object A collection of configurable attributes, for example, a collection of attributes for the directory service.

master channel program A channel program that typically initiates a transfer to a remote system. See also **slave channel program**.

master directory server The directory server that contains the data that will be replicated.

mboxutil A command-line utility for managing mail folders. This utility lists, creates, deletes, renames, or moves mailboxes (folders). It can also be used to report quota information.

MD5 A message digest algorithm by RSA Data Security. MD5 can be used to produce a short digest of data that is unique with high probability. It is mathematically extremely hard to produce a piece of data that produces the same message digest email.

member A user or group who receives a copy of an email addressed to a distribution list. See also **distribution list**, **expansion**, **moderator**, **owner**.

message The fundamental unit of email, a message consists of a header and a body and is often contained in an envelope while it is in transit from the sender to the recipient.

message access services The protocol servers, software drivers, and libraries that support client access to the Messaging Server message store.

message delivery The act that occurs when an MTA delivers a message to a local recipient (a mail folder or a program).

message forwarding The act that occurs when an MTA sends a message delivered to a particular account to one or more new destinations as specified by the account's attributes. Forwarding may be configurable by the user. See also **message delivery**, **message routing**.

Message Handling System (MHS) A group of connected MTAs, their user agents, and message stores.

message routing The act of transferring a message from one MTA to another when the first MTA determines that the recipient is not a local account, but might exist elsewhere. Routing is normally configurable only by a network administrator. See also **message forwarding**.

message queue The directory where messages accepted from clients and other mail servers are queued for delivery (immediate or deferred).

message quota A limit defining how much disk space a particular folder can consume.

message store The database of all locally delivered messages for a Messaging server instance. Messages can be stored on a single physical disk or stored across multiple physical disks.

message store administrator User who has administrative privileges to manage the message store for a Messaging Server installation. This user can view and monitor mailboxes, and specify access control to the store. Using proxy authorization rights, this user can run certain utilities for managing the store.

message store partition A message store or subset of a message store residing on a single physical file system partition.

message submission The client User Agent (UA) transfers a message to the mail server and requests delivery.

Message Transfer Agent (MTA) 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 Multiplexor A specialized iPlanet Messaging Server that acts as a single point of connection to multiple mail servers, facilitating the distribution of a large user base across multiple mailbox hosts.

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 **webmail**.

MHS See **Message Handling System**.

MIME See **Multipurpose Internet Mail Extension**.

mkbackupdir A utility that creates and synchronizes the backup directory with the information in the message store. It is used in conjunction with Legato Networker.

MMP See Messaging Multiplexor.

moderator A person who first receives all email addressed to a distribution list before (A) forwarding the message to the distribution list, (B) editing the message and then forwarding it to the distribution list, or (C) not forwarding the message to the distribution list. See also **distribution list**, **expansion**, and **member**.

MoveUser A command-line utility for moving messages in a user's mail folder from one Messaging Server to another.

MTA See **Message Transfer Agent**.

MTA configuration file The file (imta.cnf) that contains all channel definitions for the Messaging Server as well as the rewrite rules that determine how addresses are rewritten for routing. See also **channel** and **rewrite rules**.

MTA directory cache a snapshot of the directory service information about users and groups required by the MTA to process messages. See also **directory synchronization**.

MTA hop The act of routing a message from one MTA to another.

MUA See **user agent**.

Multiplexor See **Messaging Multiplexor**.

Multipurpose Internet Mail Extension (MIME) A protocol you can use to include multimedia in email messages by appending the multimedia file in the message.

MX record Mail Exchange Record. A type of DNS record that maps one host name to another.

name resolution The process of mapping an IP address to the corresponding name. See also **DNS**.

namespace The tree structure of an LDAP directory. See **directory information tree**.

naming attribute The final attribute in a directory information tree distinguished name. See also **relative distinguished name**.

naming context A specific subtree of a directory information tree that is identified by its DN. In iPlanet Directory Server, specific types of directory information are stored in naming contexts. For example, a naming context which stores all entries for marketing employees in the Siroe Corporation at the Boston office might be called ou=mktg, ou=Boston, o=Siroe, c=US.

NDN See **nondelivery notification**.

network manager A program that reads, formats and displays SNMP data. Also called an SNMP client.

next-hop list A list of adjacent systems a mail route uses to determine where to transfer a message. The order of the systems in the next-hop list determines the order in which the mail route transfers messages to those systems.

NIS A distributed network information service containing key information about the systems and the users on the network. The NIS database is stored on the master server and all the replica or slave servers.

NIS+ A distributed network information service containing hierarchical information about the systems and the users on the network. The NIS+ database is stored on the master server and all the replica servers.

NMS Netscape Messaging Server.

node A domain entry in the DIT.

nondelivery notification During message transmission, if the MTA does not find a match between the address pattern and a rewrite rule, the MTA sends a nondelivery report back to the sender with the original message.

notary messages Nondelivery notifications (NDNs) and delivery status notifications (DSNs) that conform to the NOTARY specifications RFC 1892.

notification message A type of message, sent by the Messaging Server providing the status of message delivery processing, as well as the reasons for any delivery problems or outright failures. It is for informational purposes and requires no action from the postmaster. See **delivery status notications**.

object class A template specifying the kind of object the entry describes and the set of attributes it contains. For example, iPlanet Directory Server specifies an emailPerson object class which has attributes such as commonname, mail (email address), mailHost, and mailQuota.

off-line state A state in which the mail client downloads messages from a server system to a client system where they can be viewed and answered. The messages might or might not be deleted from the server.

online state A state in which messages remain on the server and are remotely responded to by the mail client.

organization administrator User who has administrative privileges to create, modify, and delete mail users and mailing lists in an organization or sub-organization by using the Delegated Administrator for Messaging GUI or CLIs.

OSI tree A directory information tree that mirrors the Open Systems Interconnect network syntax. An example of a distinguished name in an OSI tree would be cn=billt,o=bridge,c=us.

partition See **message store partition**.

password authentication Identification of a user through user name and password. Compare **certificate-based authentication**.

pattern A string expression used for matching purposes, such as in Allow and Deny filters.

permanent failure An error condition that occurs during message handling. When this occurs, the message store deletes its copy of an email message. The MTA bounces the message back to the sender and deletes its copy of the message.

personal folder A folder that can be read only by the owner. See also **shared folder**.

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 **SSL** and **SASL**.

plaintext authentication See **password authentication**.

POP3 See **Post Office Protocol Version 3**.

port number A number that specifies an individual TCP/IP application on a host machine, providing a destination for transmitted data.

postmaster account An alias for the email group and email addresses who receive system-generated messages from the Messaging Server. The postmaster account must point to a valid mailbox or mailboxes.

Post Office Protocol Version 3 (POP3) 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, where often the mail client and the message transfer agent are on different computers.

process A self-contained, fully functional execution environment set up by an operating system. Each instance of an application typically runs in a separate process. Compare **thread**.

protocol A formal description of messages to be exchanged and rules to be followed for two or more systems to exchange information.

provisioning The process of adding, modifying or deleting entries in the iPlanet Directory Server. These entries include users and groups and domain information.

proxy The mechanism whereby one system “fronts for” another system in responding to protocol requests. Proxy systems are used in network management to avoid having to implement full protocol stacks in simple devices, such as modems.

public key encryption A cryptographic method that uses a two-part key (code) that is made up of public and private components. To encrypt messages, the published public keys of the recipients are used. To decrypt the messages, the recipients use their unpublished private keys known only to them.

purge message The process of permanently removing messages that have been deleted and are no longer referenced in user and group folders and returning the space to the message store file system. See also **delete message**, **expunge message**.

queue See **message queue**.

RC2 A variable key-size block cipher by RSA Data Security.

RC4 A stream cipher by RSA Data Security. Faster than RC2.

RDN Relative distinguished name. The name of the actual entry itself, before the entry's ancestors have been appended to the string to form the full distinguished name.

readership A command-line utility for collecting readership information on shared mail folders.

reconstruct A command-line utility for reconstructing mail folders.

referral A process by which the directory server returns an information request to the client that submitted it, with information about the Directory Service Agent (DSA) that the client should contact with the request. See also **knowledge information**.

regular expression A text string that uses special characters to represent ranges or classes of characters for the purpose of pattern matching.

relative distinguished name The final attribute and its value in the attribute and value sequence of the distinguished name. See also **distinguished name**.

relaying The process of passing a message from one messaging server to another messaging server.

replica directory server The directory that will receive a copy of all or part of the data.

restore The process of restoring the contents of folders from a backup device to the message store. See also **backup**.

reverse DNS lookup The process of querying the DNS to resolve a numeric IP address into the equivalent fully qualified domain name.

rewrite rules Also known as domain rewrite rules. A tool that the MTA uses to route messages to the correct host for delivery. Rewrite rules perform the following functions: (1) extract the host / domain specification from an address of an incoming message, (2) match the host / domain specification with a rewrite rule pattern, (3) rewrite the host / domain specification based on the domain template, and (4) decide which channel queue the message should be placed in.

RFC Request For Comments. The document series, begun in 1969, describes the Internet suite of protocols and related experiments. Not all (in fact very few) RFCs describe Internet standards, but all Internet standards are published as RFCs. See http: / / www.imc.org / rfcs.html.

root entry The first entry of the directory information tree (DIT) hierarchy.

router A system responsible for determining which of several paths network traffic will follow. It uses a routing protocol to gain information about the network, and algorithms to choose the best route based on several criteria known as “routing matrix.” In OSI terminology, a router is a Network Layer intermediate system. See also **gateway**.

routing See **message routing**.

safe file system A file system performs logging such that if a system crashes it is possible to rollback the data to a pre-crash state and restore all data. An example of a safe file system is Veritas File System, VxFS.

SASL See **Simple Authentication and Security Layer**.

schema Definitions—including structure and syntax—of the types of information that can be stored as entries in iPlanet Directory Server. When information that does not match the schema is stored in the directory, clients attempting to access the directory might be unable to display the proper results.

SCM See **Service Control Manager**.

search base See **base DN**.

Secure Sockets Layer (SSL) A software library establishing a secure connection between two parties (client and server).

security-module database A file that contains information describing hardware accelerators for SSL ciphers. Also called secmod.

sendmail A common MTA used on UNIX machines. In most applications, iPlanet Messaging Server can be used as a drop-in replacement for sendmail.

server administrator Person who performs server management tasks. The server administrator provides restricted access to tasks for a particular server, depending upon task ACIs. The configuration administrator must assign user access to a server. Once a user has server access permissions, that user is a server administrator who can provide server access permissions to users.

server instance The directories, programs, and utilities representing a specific server installation.

server root The directory into which all iPlanet servers associated with a given Administration Server on a given host are installed. Typically designated *ServerRoot*. Compare **installation directory**, **instance directory**.

server side rules (SSR) A set of rules for enabling server-side filtering of mail. Based on the Sieve mail filtering language.

service (1) A function provided by a server. For example, iPlanet Messaging Server provides SMTP, POP, IMAP, and HTTP services. (2) A background process on Windows NT that does not have a user interface. iPlanet servers on Windows NT platforms run as services. Equivalent to **daemon**.

Service Control Manager Windows NT administrative program for managing services.

session An instance of a client-server connection.

shared folder A folder that can be read by more than one person. Shared folders have an owner who can specify read access to the folder and who can delete messages from the shared folder. The shared folder can also have a moderator who can edit, block, or forward incoming messages. Only IMAP folders can be shared. Compare **personal folder**.

Sieve A proposed language for filtering mail.

Simple Authentication and Security Layer (SASL) 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 (SMTP) 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.

SIMS Sun Internet Mail Server

single field substitution string In a rewrite rule, part of the domain template that dynamically rewrites the specified address token of the host / domain address. See also **domain template**.

single sign-on. The ability for a user to authenticate once and gain access to multiple services (mail, directory, file services, and so on).

SIZE An SMTP extension enabling a client to declare the size of a particular message to a server. The server may indicate to the client that it is or is not willing to accept the message based on the declared message size; the server can declare the maximum message size it is willing to accept to a client. Defined in RFC 1870.

slave channel program A channel program that accepts transfers initiated by a remote system. See also **master channel program**.

smart host The mail server in a domain to which other mail servers forward messages if they do not recognize the recipients.

SMTP See **Simple Mail Transfer Protocol**.

SMTP AUTH See **AUTH**.

sn Aliased directory attribute for surname.

spoofing A form of network attack in which a client attempting to access or send a message to a server misrepresents its host name.

SSL See **Secure Sockets Layer**.

SSR See **Server Side Rules**.

static group A mail group defined statically by enumerating each group member. See also **dynamic group**.

stored A command-line utility that performs daily maintenance tasks on the message store. This utility expunges and erases messages stored on disk.

subdomain A portion of a domain. For example, in the domain name corp.siroe.com, corp is a subdomain of the domain siroe.com. See also **host name** and **fully-qualified domain name**.

subnet The portion of an IP address that identifies a block of host IDs.

subordinate reference The naming context that is a child of the naming context held by your directory server. See also **knowledge information**.

synchronization (1) The update of data by a master directory server to a replica directory server. (2) The update of the MTA directory cache.

TCP See **Transmission Control Protocol**.

TCP/IP See **Transmission Control Protocol/Internet Protocol**.

thread A lightweight execution instance within a process.

TLS See **Transport Layer Security**.

top-level administrator User who has administrative privileges to create, modify, and delete mail users, mailing lists, family accounts, and domains in an entire Messaging Server namespace by using the Delegated Administrator for Messaging GUI or CLIs. By default, this user can act as a message store administrator for all messaging servers in the topology.

transient failure An error condition that occurs during message handling. The remote MTA is unable to handle the message when it's delivered, but may be able to later. The local MTA returns the message to the queue and schedules it for retransmission at a later time.

Transmission Control Protocol (TCP) The basic transport protocol in the Internet protocol suite that provides reliable, connection-oriented stream service between two hosts.

Transmission Control Protocol/Internet Protocol (TCP/IP) 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: TCP (Transmission Control Protocol), the transport layer protocol, and IP (Internet Protocol), the network layer protocol.

Transport Layer Security (TLS). The standardized form of SSL. See also **Secure Sockets Layer**.

transport protocols Provides the means to transfer messages between MTAs, for example SMTP and X.400.

UA See **user agent**.

UBE See **Unsolicited Bulk Email**.

UID (1) User identification. A unique string identifying a user to a system. Also referred to as a userID. (2) Aliased directory attribute for userID (login name).

unified messaging The concept of using a single message store for email, voicemail, fax, and other forms of communication. iPlanet Messaging Server provides the basis for a complete unified messaging solution.

Unsolicited Bulk Email (UBE) Unrequested and unwanted email, sent from bulk distributors, usually for commercial purposes.

upper reference Indicates the directory server that holds the naming context above your directory server's naming context in the directory information tree (DIT).

user account An account for accessing a server, maintained as an entry on a directory server.

user agent (UA) The client component, such as Netscape Communicator, that allows users to create, send, and receive mail messages.

User/Groups Directory Server A Directory Server that maintains information about users and groups in an organization.

user entry or user profile Fields that describe information about each user, required and optional, examples are: distinguished name, full name, title, telephone number, pager number, login name, password, home directory, and so on.

user folders A user's email mailboxes.

user quota The amount of space, configured by the system administrator, allocated to a user for email messages.

UUCP UNIX to UNIX Copy Program. A protocol used for communication between consenting UNIX systems.

vanity domain A domain name associated with an individual user—not with a specific server or hosted domain. A vanity domain is specified by using the MailAlternateAddress attribute. The vanity domain does not have an LDAP entry for the domain name. Vanity domains are useful for individuals or small organizations desiring a customized domain name, without the administration overhead of supporting their own hosted domain. Also called custom domain.

/var/mail A name often used to refer to Berkeley-style inboxes in which new mail messages are stored sequentially in a single, flat text file.

Veritas Cluster Server High availability clustering software from Veritas Software with which iPlanet Messaging Server can integrate.

virtual domain (1) An ISP hosted domain. See also **hosted domain**. (2) A domain name added by the Messaging Multiplexor to a client's user ID for LDAP searching and for logging into a mailbox server.

VRFY An SMTP command for verifying a user name. Defined in RFC 821.

webmail 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**.

wildcard A special character in a search string that can represent one or more other characters or ranges of characters.

workgroup Local workgroup environment, where the server performs its own routing and delivery within a local office or workgroup. Interdepartmental mail is routed to a backbone server. See also **backbone**.

X.400 A message handling system standard.

Index

SYMBOLS

! (exclamation point)
  as a comment indicator, 104
  in addresses, 132
$?, 148
$A, 146
$B, 146
$C, 145, 148
$E, 146
$F, 146
$M, 144, 148
$N, 144, 148
$P, 146
$Q, 145, 148
$R, 146
$S, 146
$T, 148
$U substitution sequence, 136
$X, 146
% (percent sign), 145
| vertical bar, 127
^ (at sign), 148

A

access control
  access to TCP services, overview, 301
  client access, 72
  creating access filters, 309
  filter syntax, 302
  HTTP service, 72, 301
  IMAP service, 72, 301
  mapping tables, 202
  message store, 242
  POP service, 72, 301
  SMTP service, 202
  testing mappings, 213
  when applied, 212
access control, *See Also* mapping tables
addresses
  envelope To:, 145
  invalid, 184
addressing information
  alternate addresses, 49, 57
  for mail users, 48
  for mailing lists, 57
  forwarding addresses, 52
  primary address, 49, 57
administrative topology, 40
administrator access control
  configuring, 298
  to message store, 242
  to server as a whole, 299
  to server tasks, 299
after channel keyword, 176
aging policies
  message store, 251
  number of days, 251
  number of messages, 251
  size of mailbox, 251
  specifying, 251

alarm attributes
disk space, 261
aliases
alias database, 114
alias file, 106, 114
including other files in aliases file, 115
allowetrn channel keyword, 162
allowswitchchannel channel keyword, 172
alternate channel for incoming mail, 172
alternate email addresses, 49, 57
at sign, 132, 145, 148
authentication
certificate-based, 284, 288
HTTP, 68
IMAP, 68
mechanisms, 284
Multiplexor, 85
password, 286
POP, 68
SASL, 284
SMTP, 287
authorized services, 54
automatic reply
configuring languages for, 32
settings, 53
autoreply option file, 106

B

backoff channel keyword, 176
backup groups, 273
backup procedure for message store
backup utilities, 274, 275
creating a policy, 272
creating backup groups, 273
description, 271
full backup, 273
incremental backup, 273
parallel backups, 273
peak business loads, 272
serial backups, 273
single copy procedure, 272
using Legato Networker, 277
bangoverpercent keyword, 132
bang-style (UUCP) addresses, 126
bang-style address conventions, 132
banners
IMAP, 68
POP, 68
blank lines
in a configuration file, 104
blocketrn channel keyword, 162

C

CA certificates
installing, 292
managing, 293
cacheeverything channel keyword, 169
cachefailures channel keyword, 169
cachesuccesses channel keyword, 169
certificate-based login, 69, 296
certificates
installing, server, 291
installing, trusted CA, 292
managing, 293
obtaining, 290
requesting, server, 291
channel block, 154
channel l, 104
channel processing
simultaneous requests, 111
channel protocol selection, 160
channel/host table, 154
channels
alternates, 172
channel-specific rule checks, 145
character set labeling, 163
comment lines in definitions, 154
configuring, 153
connection caching, 169
description, 96
DNS lookups, 169
eight-bit data, 164
IDENT lookups, 170
interpreting names of, 145

job processing pools, 179
keywords for, 158
master programs, 97
message queues, 99
nameserver lookups, 172
option files, 106
predefined, 156
protocol selection and line terminators, 160
protocol streaming, 164
SASL support, 174
slave programs, 97
SMTP authentication, 174
structure of, 154
submit only, 175
target host choice, 173
TCP/IP MX record support, 171
TCP/IP port selection, 168
TLS kewords, 174
character set labeling, 163
charset7 channel keyword, 163
charset8 channel keyword, 163
charsetesc channel keyword, 163
checkehlo channel keyword, 161
ciphers
about, 294
selecting, 295
command-line utilities
mboxutil, 257
MoveUser, 268
MTA, 115
reconstruct, 260
stored, 262
configuration directory, 40, 41
configuration files
aliases, 106
autoreply option, 106
blank lines in, 104
conversion, 107
dirsync option, 107
Dispatcher, 107
Job Controller, 110
local.conf, 24
mapping, 108
msg.conf, 24
MTA, 24, 103
nsswitch.conf, 172
options, 109
sslpassword.conf, 24, 293
tailor, 109
conn_throttle.so, 211
connection caching, 169
controlling error messages associated with rewriting, 148
conventions used in this document, 16
conversion channel, 189
configuration of, 190
conversion control, 107, 190
traffic for conversion processing, 190
conversion control, 107, 190
conversion file, 107
conversion processing, traffic for, 190
corresponding channel characteristics, 172

D

daemon channel keyword, 173
default error messages
rewrite and channel matching failures, 148
defaultmx channel keyword, 171
defaultnameservers channel keyword, 172
defaults channel
in a configuration file, 104, 155
delegated administration, 46, 298
Delegated Administrator for Messaging, 20, 46
delivery options
mail users, 50
POP/IMAP delivery, 51
program delivery, 51
UNIX delivery, 52
direction-specific rewrites, 146
directories
for log files, 318
message store, 239
Directory Server, 40
configuration directory, 40
configuration settings, 41
MTA cache, 116
requirement, 40

user directory, 40, 45
dirsync option file, 107
disk space
monitoring, 261
quotas for, 244
Dispatcher
configuration file, 107
controlling, 102
description, 101
MAX_CONNS option, 101
MIN_CONNS option, 101
MIN_PROCS option, 101
restarting, 102
starting, 102
stopping, 102
dispatcher configuration file, 107
DNS
domain verification, 163
IDENTprotocol, 170
lookups, 169
MX records, 171
reverse lookups, 170
DNS Lookups, 222
domain
database, 149
DNS verification, 163
literals, 135
specification in an address, 130
domainetrn channel keyword, 162

E

echo mode, 54
ehlo channel keyword, 161
EHLO command, 160
eightbit channel keyword, 164
eight-bit data, 164
eightnegotiate channel keyword, 164
eightstrict channel keyword, 164
email-only members (of a group), 56
encryption
accelerators for, 291
envelope To: address, 145
ETRN command, 161
exclamation point (!), 132
expandchannel channel keyword, 177
expandlimit channel keyword, 177
external modules (PKCS #11), 290

F

failed delivery attempts, 184
failed messages, 184
failure of rewrite rules, 135
filesperjob channel keyword, 177
filters
channel level, 229
description, 201
IP Address, 211
MTA-wide, 229
per-user, 229
forwardcheckdelete channel keyword, 169
forwardchecknone channel keyword, 169
forwardchecktag channel keyword, 169
forwarding addresses, 52
FROM_ACCESS mapping table, 203, 207
fully qualified domain name (FQDN), 131

G

glossary, 363
greeting message, 32
groups
See also mailing lists
email-only members of, 56
Members tab, 55

H

hold channel, 189
holdlimit channel keyword, 177

host location-specific rewrites, 146
host name
  extracting, 131
  hiding, 49, 58
host / domain specifications, 131
hosted domains
  description, 20
HTTP service
  access control filters, 309
  certificate-based login, 69
  client access control, 72
  configuring, 76
  connection settings, 78
  connections per process, 70
  disabling, 78
  dropping idle connections, 72
  enabling, 78
  logging out clients, 72
  login requirements, 68
  message settings, 78
  MTA settings, 78
  number of processes, 70
  password-based login, 68, 78
  performance parameters, 69
  port numbers, 66
  process settings, 78
  proxy authentication, 310
  security, 283
  session ID, 283
  specialized web server, 21, 76
  SSL port, 67
  starting and stopping, 28
  threads per process, 71

I

iddenttcpsymbolic channel keyword, 170
IDENT lookups, 170
identnone channel keyword, 171
identnonelimited channel keyword, 171
identnonenumeric channel keyword, 171
identnonesymbolic channel keyword, 171
identtcp channel keyword, 170
identtcplimited channel keyword, 170
identtcpnumeric channel keyword, 170
idle connections, dropping, 72
IMAP service
  access control filters, 309
  banner, 67, 75
  certificate-based login, 69, 296
  client access control, 72
  configuring, 74
  connection settings, 75
  connections per process, 70
  disabling, 75
  dropping idle connections, 72
  enabling, 75
  login requirements, 68
  number of processes, 70
  password-based login, 68, 287
  password-based long, 75
  performance parameters, 69
  port numbers, 66, 67
  process settings, 75
  readership utility, 260
  shared folders, 260
  SSL, 67, 288
  SSL port, 67
  starting and stopping, 28
  threads per process, 71
immnonurgent channel keyword, 176
imsbackup utility, 274, 275
imsrestore utility, 274, 275
INBOX, default mailbox, 258
incoming connection, 172
interfaceaddress channel keyword, 168
internal modules (PKCS #11), 290
invalid address, 184
IP Address filtering, 211

J

Job Controller
  commands, 111
  configuration file, 110
  creating processes, 110

description, 100
examples of use, 111
JOB_LIMIT option, 113
JOB_LIMIT pool option, 178
MAX_MESSAGES option, 178
maxjobs channel option, 178
restarting, 100
SLAVE_COMMAND option, 113
starting, 100
stopping, 100
job_controller.cnf file, 100
JOB_LIMIT Job Controller option, 113, 178

L

languages
for autoreply messages, 32
server site, 34
user-preferred, 33
last resort host, 172
lastresort channel keyword, 172
LDAP directory
configuration directory, 40
configuring lookups in user directory, 40
customizing lookups, 40
for user provisioning, 20
MTA cache, 116
requirements, 40
user directory, 40, 45
viewing settings in configuration directory, 41
local.conf file, 24
localvrfy channel keyword, 163
location-specific rewrites, 146
LOG_CONNECTION option, 328
LOG_FILENAME option, 328
LOG_MESSAGE_ID option, 328
LOG_MESSAGES_SYSLOG option, 328
LOG_PROCESS option, 328
LOG_USERNAME option, 328
logging
analyzing logs, 314
architecture of, 320
categories, 317
channels, 326
directories for log files, 318
levels of, 315
LOG_CONNECTION option, 328
LOG_FILENAME option, 328
LOG_MESSAGE_ID option, 328
LOG_MESSAGES_SYSLOG option, 328
LOG_PROCESS option, 328
LOG_USERNAME option, 328
message store and administration server, 315
MTA, 326
MTA entry codes, 329
options, 322
SEPARATE_CONNECTION_LOG option, 328
severity levels, 315
to syslog, 323, 328
viewing logs, 324
login
certificate-based, 69, 296
password-based, 68, 286
long-term service failures, 184

M

mail accounts. See mail users
mail filtering
channel-level filters, 229
description, 201
mapping tables, 202
MTA-wide filters, 229
per-user filters, 229
server-side rules, 228
mail forwarding, 171
Mail tab, 47, 48, 56
mail users
accessing an existing user, 48
address (primary), 49
addresses, specifying, 48
alternate addresses, 49
auto-reply settings, 53
creating a new user, 47
delivery-options configuration, 50
echo mode, 54
forwarding addresses for, 52

host name hiding, 49
Mail tab, 47, 48
Netscape Console access to, 47
POP/IMAP delivery option, 51
program delivery option, 51
UNIX delivery option, 52
vacation mode, 54
MAIL_ACCESS mapping table, 202, 205
mailboxes
aging policies for, 251
default mailbox for delivery, 258
INBOX, 258
managing, 257
mboxutil utility, 257
naming conventions for, 258
reconstruct utility, 263
reconstructing, 264
repairing, 263
mailfromdnsverify channel keyword, 163
mailing lists
accessing an existing group, 56
adding list (email-only) members, 61
address (primary), 57
creating a new group, 55
dynamic membership criteria, 60
email-only members, 56
host name hiding, 58
LDAP search URLs, 60
list members, 59
list owners, 58
Mail tab, 56
Members tab (of group), 55
message-rejection actions, 63
moderators for, 63
Netscape Console access to, 55
restrictions on message posting, 62
mapping file, 108
mapping tables
description, 202
FROM_ACCESS, 203
handling large numbers of entries, 224
MAIL_ACCESS, 202
ORIG_MAIL_ACCESS, 202
ORIG_SEND_ACCESS, 202
PORT_ACCESS, 203, 211
SEND_ACCESS, 202
mapping tables,*See Also* access control
master program, 111
master_command, 113
matching any address, 127
matching procedure, rewrite rules, 133
MAX_CONNS Dispatcher option, 101
MAX_MESSAGES Job Controller option, 178
MAX_PROCS Dispatcher option
Dispatcher
MAX_PROCS option, 101
maxjobs channel keyword, 177, 178
maysaslserver channel keyword, 174
maytls channel keword, 174
maytlsclient channel keyword, 174
maytlsserver channel keyword, 174
Members tab, 55
message store
access control, 242
administrator access, 242
aging policies, 251
backup groups, 273
backup policies, 272
cleaning up messages, 242
configuring disk quotas, 244
configuring partitions, 254
default partition, 255
deleting messages, 242
directory layout, 239
expunging messages, 242
imsbackup utility, 275
imsrestore utility, 275
logging, 315
maintenance and recovery procedures, 257
MoveUser utility, 268
overview, 237
partitions, 250
primary partition, 254
RAID technology, 254
reconstruct utility, 264
restoring data, 275
stored utility, 262
using Legato Networker for backup, 277
Message Transfer Agent. See also MTA
Messenger Express, 21, 65
migrating users, 189

MIN_CONNS Dispatcher option, 101
MIN_PROCS Dispatcher option, 101
moderators
  defining, 63
  for mailing lists, 63
MoveUser command-line utility, 268
moving mailboxes, 255
msg.conf file, 24
MTA
  adding relaying, 214
  channels, 96
  command-line utilities, 115
  configuration files, 103, 105
  description, 96
  directory cache, 116
  directory synchronization, 116
  Dispatcher, 101
  Job Controller, 100
  logging, 326
  relay blocking, 217
  rewrite rules, 99
MTA configuration file, 103
multiple $M clauses, 145
multiple outgoing channels, 172
Multiplexor
  certificate-based authentication, 86
  certmap plugins, 85
  configuration, 89
  description, 81
  DNComps, 85
  encryption, 85
  features, 82
  FilterComps, 85
  how it works, 84
  IMAP configuration file, 89
  instances (multiple), 87
  POP configuration file, 89
  pre-authentication, 86
  starting/stopping, 90
  store administrator, 85
  vdmap, 87
mustsaslserver channel keyword, 174
musttls channel keyword, 174
musttlsclient channel keyword, 174
musttlsserver channel keyword, 174
mx channel keyword, 171
MX record support, 171
myprocmail, with the Pipe channel, 188

N

nameserver lookups, 172
nameservers channel keyword, 172
network services, 111
nobangoverpercent keyword, 132
nocache channel keyword, 169
noehlo channel keyword, 161
nomailfromdnsverify channel keyword, 163
nomx channel keyword, 171
nonrandommx channel keyword, 171
nonurgentbackoff channel keyword, 176
nonurgentblocklimit channel keyword, 176
nonurgentnotices channel keyword, 177
normalbackoff channel keyword, 176
normalblocklimit channel keyword, 176
normalnotices channel keyword, 177
norules channel keyword, 145
nosasl channel keywrod, 174
nosaslserver channel keyword, 174
nosmtp channel keyword, 160
noswitchchannel keyword, 172
notices channel keyword, 177
notls channel keyword, 174
notlsclient channel keyword, 174
notlsserver channel keyword, 174
nsswitch.conf file, 172

O

options file, 109
ORIG_MAIL_ACCESS mapping table, 202, 205
ORIG_SEND_ACCESS mapping table, 202, 203

P

partitions
  adding, 255
  configuring for message store, 254
  default, 255
  full, 255
  message store, 250
  moving mailboxes between, 255
  nicknames, 255
  pathnames, 255
  primary, 254
  RAID technology, 254
password authentication
  See also login
  HTTP service, 68
  IMAP service, 68
  POP service, 68
  SMTP service, 287
  to LDAP user directory, 42
password file (for SSL), 293
password login, 68, 286
percent hack, 132
percent hack rules, 126
percent sign (%), 145, 148
performance parameters
  connections per process, 70
  number of processes, 70
  threads per process, 71
pipe channel, 187
PKCS #11
  internal and external modules, 290
pool channel keyword, 176
POP service
  access control filters, 309
  banner, 67
  certificate-based login, 296
  client access control, 72
  configuring, 73
  connections per process, 70
  dropping idle connections, 72
  login requirements, 68
  number of processes, 70
  password-based login, 68, 287
  performance parameters, 69
  port numbers, 66
  SSL, 288
  starting and stopping, 28
  threads per process, 71
port channel keyword, 168
PORT_ACCESS Mapping Table, 209
PORT_ACCESS mapping table, 203, 211
postheadbody channel keyword, 185
postheadonly channel keyword, 185
pre-authentication (Multiplexor), 86
primary email address, 49, 57
processes
  number of, 70
program delivery
  pipe channel, 187
  setting up, 187
  specifying, 51
programs
  master, 111
  slave, 111
provisioning users, 20

Q

quotas
  configuring, 244
  disk, 245
  disk space, 244
  domain, 246
  enforcement, 248
  family groups, 246
  grace period, 250
  message, 245
  notification, 248
  warning message, 249

R

RAID technology
  for message store, 254
randommx channel keyword, 171
RBL Checking, 222

reconstruct command-line utility, 260
recovery tasks
  mailboxes, 263
  reconstruct utility, 260
relay blocking, 217
relay blocking, removal of, 214
relaying
  adding, 214
remote system, 172
repeated percent signs, 132
restoring the message store, 271
rewrite process failure, 130
rewrite rules, 104
  bang-style, 126
  blank lines, 104, 154
  description, 99
  direction-specific, 146
  failure, 135
  Finishing the Rewriting Process, 134
  handling large numbers, 149
  host location-specific, 146
  location-specific, 146
  match any address, 127
  operation, 130
  pattern matching, 130
  percent hacks, 126
  scanning, 133
  structure, 122
  syntax checks after rewriting, 135
  tagged rule sets, 127
  templates, 134
  testing, 149
  UUCP addresses, 126
rewriting an address
  extracting the first host / domain specification, 131
rewriting error messages, 148
rules channel keyword, 145

S

SASL
  channel keywords, 174
  description, 284
saslswitchchannel channel keyword, 174
security
  about, 282
  authentication mechanisms, 284
  certificate-based login, 69, 296
  client access controls, 73
  client access to TCP services, 301
  HTTP service, 72, 283
  IMAP service, 72
  password-based login, 68
  POP service, 72
  SASL, 284
  SMTP service, 287
  SSL, 288
  TLS, 288
SEND_ACCESS mapping table, 202, 203
SEPARATE_CONNECTION_LOG option, 328
server certificates
  installing, 291
  managing, 293
  requesting, 291
server information, viewing, 28
server-side rules, 228
service banners, 67
services
  enabling and disabling, 66
  HTTP, 65
  IMAP, 65
  MTA, 95
  POP, 65
  SMTP, 95
  starting and stopping, 28
sevenbit channel keyword, 164
severity levels (of logging), 315
shared folders, IMAP, 260
silentetrn channel keyword, 162
single channel keyword, 173
single sign-on
  enabling, 35
  Messenger Express and Delegated Administrator, 37
  Messenger Express configuration parameters, 35
single_sys channel keyword, 173
slave program, 111

SLAVE_COMMAND Job Controller option, 113
SMTP AUTH, 214
smtp channel keyword, 160
SMTP MAIL TO command, 162
SMTP Relaying
  adding, 214
SMTP Relaying for External Sites,allowing in NMS, 216
SMTP service
  access control, 201
  adding relaying, 214
  authenticated SMTP, 287
  login requirements, 287
  password-based login, 287
  port number, 288
  relay blocking, 217
  starting and stopping, 28
smtp_cr channel keyword, 160
smtp_crlf channel keyword, 160
smtp_crorlf channel keyword, 160
smtp_lf channel keyword, 160
SNMP, 349
  applTable, 354
  applTable Usage, 355
  assocTable, 355
  assocTable Usage, 356
  channel errors, 360
  channel information, 357
  channel network connection, 359
  co-existence with other iPlanet products, 353
  configuring for Messaging Server, 351
  implementation, 349
  information provided, 353
  limitations, 350
  MIBs supported, 349
  MTA information, 356
  mtaGroupAssociationTable, 359
  mtaGroupErrorTable, 360
  mtaGroupErrorTable Usage, 361
  mtaGroupTable, 357
  mtaGroupTable Usage, 359
  mtaTable, 356
  mtaTable Usage, 357
  network connection information, 355
  operation, 350
  server information, 354
source channel-specific
  rewriting, 145
source-routed address, 131
SSL
  certificates, 290
  ciphers, 294
  enabling, 294
  hardware encryption accelerators, 291
  installing CA certificates, 292
  installing server certificates, 291
  internal and external modules, 290
  managing certificates, 293
  overview, 288
  password file for, 293
  requesting server certificates, 291
  sslpassword.conf file, 24
  turning on, 295
sslpassword.conf file, 24, 293
sticky error message, 148
streaming channel keyword, 164
submit channel keyword, 175
switchchannel channel keyword, 172
syntax checks after rewriting, 135
syslog
  message store logging, 323
  MTA logging, 328

T

tagged rewrite rule sets, 127
tailor file, 109
TCP client access control
  address-spoofing detection, 308
  examples, 306
  EXCEPT operator, 305
  filter syntax, 302
  host specification, 305
  how access filters work, 301
  identd service, 306, 307
  Netscape Console interface for, 309
  overview, 301
  username lookup, 306, 307
  virtual domains, 308

wildcard names, 304
wildcard patterns, 304
TCP/IP
channels, 106, 185
connections, 165
DNS lookups, 169
IDENT lookups, 170
interface address, 168
MX record support, 171
port number, 168
TCP/IP nameserver lookups, 172
threaddepth channel keyword, 177
threads per process, 71
throttle, 211
TLS
channel keywords, 174
description, 288
tlsswitchchannel keyword, 174
traffic for conversion processing, 190
Transport Layer Security (TLS), 288

U

Unauthorized Bulk Email, 222
unified messaging, 21
UNIX delivery, 52
unrecognized
domain specification, 148
host specification, 148
urgentbackoff channel keyword, 176
urgentblocklimit channel keyword, 176
urgentnotices channel keyword, 177
user directory, 40
user login. See login
UUCP address rewrite rules, 126

V

vacation mode, 54
vdmap (Multiplexor), 87
verbosity (of logging), 315
vertical bar (|), 127
virtual domains
controlling access to, 308
VRFY command, 162
vrfyallow channel keyword, 163
vrfydefault channel keyword, 163
vrfyhide channel keywrod, 163

W

webmail
HTTP service, 76
Messenger Express, 21, 65
support for, 21