Administrator's Guide

iPlanet™ Messaging Server

Release 5.2

February 2002

Contents

List of Tables

List of Figures

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 Guide is Organized
- Typographical Conventions
- Where to Find Related Information

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 guide assumes that you have a general understanding of the following:

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

How This Guide 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, “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 4, “Configuring and Administering Multiplexor Services“”

  This chapter provides concepts about iPlanet Messaging Multiplexor and iPlanet Messenger Express Multiplexor, specialized messaging servers that act as single points of connection to multiple messaging servers.

- Chapter 5, “MTA Concepts“”

  This chapter describes MTA concepts.

- Chapter 6, “About MTA Services and Configuration“”

  This chapter provides general information about configuring MTA services on 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, “Using Pre-defined Channels“”

  This chapter describes how to use pre-defined MTA channel definitions such as the hold channel and conversion channel.

- Chapter 10, “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 11, “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 12, “Configuring Security and Access Control“”

  Describes the security and access control features available with iPlanet Messaging Server.

- Chapter 13, “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.

- Chapter 14, “Troubleshooting the MTA“”

  This chapter describes common tools, methods, and procedures for troubleshooting the Message Transfer Agent (MTA).

- Chapter 15, “Monitoring the iPlanet Messaging Server“”

  This chapter describes iPlanet Message Server monitoring.

- Appendix A, “SNMP Support“”

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

- Appendix B, “MTA Direct LDAP Operation“”

  This appendix describes how direct LDAP operation for the MTA works.

- Appendix C, “Administering Event Notification Service in iPlanet Messaging Server“”

  This appendix describes what you need to do to enable and administer iPlanet Event Notification Service in iPlanet Messaging Server.

- Appendix D, “Managing Mail Users and Mailing Lists“”

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

- Glossary

  The glossary provides definitions for the terms and naming conventions.

Typographical Conventions

Table 1 Typographical Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, code, directories, hostnames, distinguished names, on-screen computer output.	Edit your `msg.conf` file. Use `ls -a` to list all files. `Error: illegal port #`
`AaBbCc123`	User entered text.	`% cd madonna`
the_variable	Command-line place holder or variable. Replace with a real name or value.	`#` *Instance_Root*`/start-msg`
AaBbCc123	Book titles, new words or terms, or words to be emphasized.	*iPlanet Messaging Server Provisioning Guide*

Command Line Prompts

Command line prompts (example: `%` for a C-Shell, or `$` for a Korn shell) are generally not displayed in examples. Different operating systems use different prompts. Enter the command as it appears in the document regardless of prompt.

Where to Find Related Information

iPlanet Messaging Server comes with the following supplementary information:

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

Listed below are the additional documents that are available:

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

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

iPlanet Messaging Server suuports most national, international, and industry standards related to electronic messaging. For a complete list, refer to Appendix A of the *iPlanet Messaging Server Reference Manual*.

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 remotely. A hosted domain can share 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 and provides various domain-specific default settings..

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:

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

- iPlanet Messaging Server Delegated Administrator Console online help.

NOTE You can also create users and groups with the Console interface, however, this is not recommended since doing so will prevent you from viewing and modifying these entries with the Delegated Administrator.

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 and certificate-based login to POP, IMAP, HTTP, or SMTP.
- 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, SMTP and HTTP.
- Filtering of unsolicited bulk email using system-wide and per-user 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 5, “MTA Concepts”
- Chapter 6, “About MTA Services and Configuration”
- Chapter 7, “Configuring Rewrite Rules”
- Chapter 8, “Configuring Channel Definitions”
- Chapter 9, “Using Pre-defined Channels”
- Chapter 10, “Mail Filtering and Access Control”
- Chapter 12, “Configuring Security and Access Control”
- Chapter 14, “Troubleshooting the MTA”
- Chapter 15, “Monitoring the iPlanet Messaging Server”

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 3, “Configuring POP, IMAP, and HTTP Services”
- Chapter 11, “Managing the Message Store”
- Chapter 12, “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 4, “Configuring and Administering Multiplexor Services,” 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 (*server_root*)	`/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`).
instance directory (*instance_root* or *instance_directory*)	*server_root* / msg-*instance_name* / (required location)
	instance_name 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 executables of the installed Messaging Server, such as `configutil`, start-`msg`, `stop-msg`, and so on.
installation directory (*installDirectory*)	*server_root*`/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
configuration directory `config`	*instance_root* / 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`	*instance_root* / imta / (required location)
	Contains several directories related to MTA configuration: `bin`, `config`, `db`, `dl`, `programs`, `queue`, `tmp`.
MTA configuration directory `config`	*instance_root* / imta / config / (required location)
	Contains the MTA configuration files, such as `imta.cnf`, `dispatcher.cnf`, `job_controller.cnf`, `aliases`, `imta_tailor`, and so on.
MTA queue directory `queue`	*instance_root* / imta / queue / (required location)
	Contains the message queue subdirectories. There is one subdirectory directly under this directory for each channel queue; for example: `ims-ms`, `tcp_intranet`, `tcp_local`, `autoreply`.
MTA program directory `programs`	*instance_root*`/imta/programs/` (required location)
	Contains the site-supplied executable programs—if any—for processing user mail.
MTA databases directory `db`	*instance_root*`/imta/db/` (required location)
	Contains the databases used by the MTA: `aliasesdb.db`, `domaindb.db`, `profiledb.db`, `reversedb.db`, `ssrdb.db`, and so on.
Message store directory `store`	*instance_root* / store (required location)
	Contains the directories related to message store processing: `mboxlist`, `partition`, `user`.
	For more information, see “Message Store Directory Layout,” on page 344.

Table 1-1 Post-Installation Directories and Files

Directory	Default Location and Description
Manuals directory `manual`	*server_root*`/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.

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:

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.

Managing Mail Users and Mailing Lists

All user and mailing list information is stored as entries in an LDAP directory. An LDAP 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.

Creating and managing mail users and mailing lists consists of creating and modifying user and mailing list entries in the directory. This is done using the iPlanet Delegated Administrator for Messaging, Delegated Administrator command line utilities , or by directly modifying the LDAP directory. Users and mailing list entries can also be created and using the Console, but this is NOT recommended. (See Appendix D.)

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 (see the *iPlanet Messaging Server Reference Manual*). 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*.

To View Basic Server Information

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

NOTE If you install iPlanet Directory Server 5.1, you must manage it through iPlanet Console 5.0 (installed with Directory Server 5.1). iPlanet Messaging Server 5.2 must be managed through Netscape Console 4.2 (installed with Messaging Server 5.2).

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 not.

To Start and Stop 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*
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 -offline` *resource* `-sys` *system*	`hagrp -offline` *group* `-sys` *system*

To Start and Stop 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 52.

Important: If a server process crashes, other processes may 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*.

To Configure a Greeting Message

The 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 “To Set a User-Preferred Language” on page 40) 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. If a domain preferred language is available, and if there’s a matching auto-reply message that is used.
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 or there is no domain preferred language specified, the version that matches the default server-site language (see “To Configure a Server Site Language” on page 41) 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 only one version of the message, regardless of language preference or site language, that is the version that is sent.
4. If there is no version of the message that matches either the user’s preferred language or the default site language or domain preferred language is available, and if there is more than one language version, the first message text found in the user’s LDAP entry is sent.

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

To Set a Domain Preferred Language

A domain preferred language is a default language specified for a particular domain. For example, you may wish to specify Spanish for a domain called `mexico.siroe.com`. Administrators can choose a domain preferred language for hosted domains by selecting the Preferred Language option when creating the domain in the Delegated Administrator for Messaging interface or by adding the LDAP attribute `preferredLanguage` to the domain's LDAP entry.

To Configure 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 “To Enable Single Sign-on Between Messenger Express and the Delegated Administrator for Messaging,” on page 44.

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.
`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 Messenger Express 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 Messenger Express 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.

Table 2-4 Messenger Express Single Sign-On Parameters

Parameter	Description
`local.sso.`*`appid`*`.verifyurl`	Sets the verify URL values for peer SSO applications. *appid* is the application ID of a peer SSO application 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 application. 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
```

To Enable Single Sign-on Between Messenger Express and the 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:

iDA_server_root`/nda/classes/netscape/nda/servlet/resource.properties`

Web_server_root`/https-`*instancename*`/config/servlets.properties`

Web_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. For example, the following is an example of a proxy user account entry:

```
dn: uid=proxy, ou=people, o=siroe.com, o=isp
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=isp";)
```

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 *iDA_server_root*/nda/classes/netscape/nda/servlet/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 *Web_Server_Root*/https-*instancename*/config/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 *iDA_server_root*/nda/classes/netscape/nda/servlet/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 *Web_Server_Root*/https-*instancename*/config/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 *iDA_server_root*/nda/classes/netscape/nda/servlet/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.

To Customize 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 , "."

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 specified 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 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 12, "Configuring Security and Access Control" which also contains background information on all security and access-control topics for Messaging Server.

Chapter 3

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.

NOTE If you install iPlanet Directory Server 5.1, you must manage it through the iPlanet Console 5.0 (installed with Directory Server 5.1). iPlanet Messaging Server 5.2 must be managed through Netscape Console 4.2 (installed with Messaging Server 5.2).

For information on configuring Simple Mail Transfer Protocol (SMTP) services, see Chapter 6, “About MTA Services and Configuration.”

This chapter contains the following sections:

• “To Configure HTTP Services,” on page 63

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 “To Configure POP Services” on page 59, “To Configure IMAP Services” on page 61, and “To Configure HTTP Services” on page 63.

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 35); 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.

You can enable or disable services at the server level. This process is described in this chapter. You can also enable or disable services at the user level by setting specified LDAP attributes. For further information see the *iPlanet Messaging Server Provisioning Guide*.

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 4, “Configuring and Administering Multiplexor Services.”)

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 400.

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 402), 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 (`service.imap.banner`, `service.pop.banner`) 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 "To Configure POP Services" on page 59, "To Configure IMAP Services" on page 61, or "To Configure HTTP Services" on page 63. In addition, you can specify the valid login separator for POP logins.

To Set the Login Separator for POP Clients

The messaging server will not accept `@` as the login separator for some POP mail clients (that is, the `@` in an address like `uid@domain`). Examples of these clients are Netscape Messenger 4.76, Netscape Messenger 6.0, and Microsoft Outlook Express on Windows 2000. The workaround is as follows:

1. Make `+` a valid separator with the following command:

   ```
   configutil -o service.loginseparator -v "@+"
   ```

2. Inform POP client users that they should login with `+` as the login separator, not `@`.

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 “To Enable SSL and Selecting Ciphers” on page 406.

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 “To Set Up Certificate-Based Login” on page 408.

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 "To Configure POP Services" on page 59, "To Configure IMAP Services" on page 61, or "To Configure HTTP Services" on page 63.

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 395.

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 395.

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 395.

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 413 and "Configuring Client Access to SMTP Services" on page 426.

To Configure POP Services

You can perform basic configuration of the Messaging Server POP service by using the `configutil` command or by using iPlanet Console. Some of the more common POP services options are given in this chapter. A complete listing can be found in the *iPlanet Messaging Server Reference Manual*.

For more information, see also:

- "Enabling and Disabling Services" on page 52
- "To Set the Login Separator for POP Clients" on page 54
- "Specifying Port Numbers" on page 52
- "Number of Connections per Process" on page 57
- "Dropping Idle Connections" on page 58
- "Number of Threads per Process" on page 58

- "Number of Processes" on page 56

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 57.
   - Set the maximum idle time for connections. For more information, see "Dropping Idle Connections" on page 58.
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 58.
   - Set the maximum number of processes. For more information, see "Number of Processes" on page 56.
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
```

To Configure IMAP Services

You can perform basic configuration of the Messaging Server IMAP service by using the `configutil` command or by using iPlanet Console. Some of the more common IMAP services options are given in this section. A complete listing can be found in the *iPlanet Messaging Server Reference Manual*.For more information, see also:

- "Enabling and Disabling Services" on page 52
- "Specifying Port Numbers" on page 52
- "Password-Based Login" on page 55
- "Number of Connections per Process" on page 57
- "Dropping Idle Connections" on page 58
- "Number of Threads per Process" on page 58
- "Number of Processes" on page 56

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 57.
   - Set the maximum idle time for connections. For more information, see “Dropping Idle Connections” on page 58.
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 58.
   - Set the maximum number of processes. For more information, see “Number of Processes” on page 56.
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
```

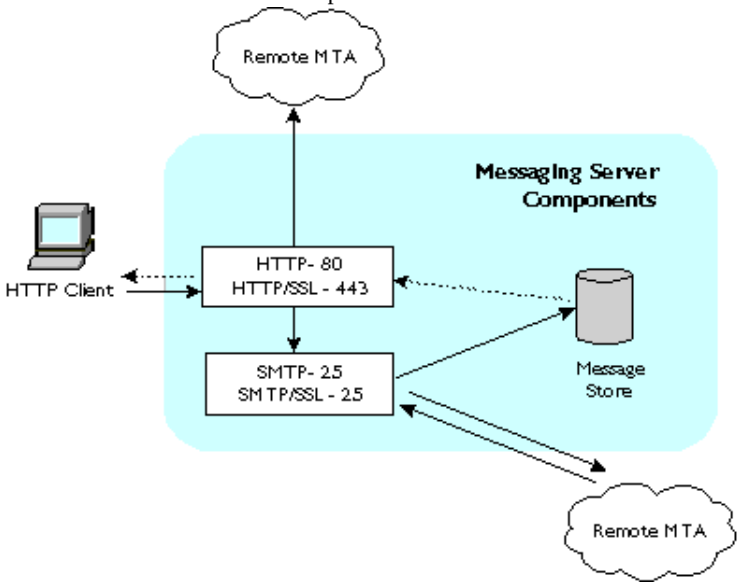
To specify a protocol welcome banner:

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

To Configure 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 3-1.

Figure 3-1 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. Some of the more common HTTP service options are given in this section. A complete listing can be found in the *iPlanet Messaging Server Reference Manual*. For more information, see also:

- “Enabling and Disabling Services” on page 52
- “Specifying Port Numbers” on page 52
- “Password-Based Login” on page 55
- “Number of Connections per Process” on page 57
- “Dropping Idle Connections” on page 58
- “Logging Out HTTP Clients” on page 59
- “Number of Threads per Process” on page 58
- “Number of Processes” on page 56

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 57.
   - Set the maximum idle time for connections. For more information, see “Dropping Idle Connections” on page 58.
   - Set the maximum idle time for client sessions. For more information, see “Logging Out HTTP Clients” on page 59.
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 58.
   - Set the maximum number of processes. For more information, see “Number of Processes” on page 56.
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 attachments being about 3.75M.

For more information, see "Message Settings" on page 65.

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 65.
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 attachments 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 4

Configuring and Administering Multiplexor Services

This chapter describes two Multiplexors that are included with iPlanet Messaging Server: the Messaging Multiplexor (MMP) for standard mail protocols (POP, IMAP and SMTP) and the Messenger Express Multiplexor used for the Messenger Express web interface.

The following topics are covered in this chapter:

- About Multiplexor Services
- About iPlanet Messaging Multiplexor (MMP)

NOTE For instructions on installing MMP, see the *iPlanet Messaging Server Installation Guide*. For details about MMP configuration parameters, see the *iPlanet MessagingServer Reference Manual*.

- About Messenger Express Multiplexor

About Multiplexor Services

A Multiplexor is necessary to achieve horizontal scalability (the ability to support more users by adding more machines), because it provides a single domain name that can be used to connect indirectly to multiple mail stores. A multiplexor can also provide security benefits.

While MMP is managed separately from iPlanet Messaging Server, Messenger Express Multiplexor is built-in to the HTTP service (`mshttpd`) that is included with the iPlanet Message Store and Message Access installation.

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 message stores, your organization can benefit in several ways from using the multiplexors. 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 more, if you have separate Multiplexor machines for POP, IMAP, SMTP or web access), 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 with identical configurations and manage connections to them by DNS round robin or by using a load-balancing system.

  Because the Multiplexors use 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 Multiplexors perform 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 a Multiplexor, you might be able to decrease overall costs by purchasing several small server machines that together cost less than one very large machine.

- **Better Scalability**

  With the Multiplexors, 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 Multiplexors 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 Multiplexor 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 Multiplexors support both unencrypted and encrypted communications with clients.

About iPlanet Messaging Multiplexor

The iPlanet Messaging Multiplexor (MMP) is a specialized messaging server that acts as a single point of connection to multiple back-end messaging servers. With Messaging Multiplexor, large-scale messaging-service providers can distribute POP and IMAP user mailboxes across many machines to increase message store 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 Messaging Multiplexor 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:

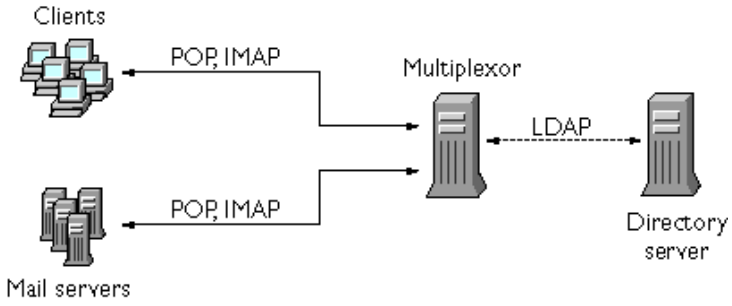
- Both unencrypted and encrypted (SSL) communications with mail clients.
- Client certificate-based authentication, described in “Certificate-Based Client Authentication” on page 74.
- User pre-authentication, described in “User Pre-Authentication” on page 75.
- Virtual domains that listen on different IP addresses and automatically append domain names to user IDs, described in “MMP Virtual Domains” on page 75.
- Multiple installations of the MMP on different machines (one installation per machine). See the *iPlanet Messaging Server Installation Guide*.
- Multiple instances of Multiplexor on a server machine, described in “Multiple Messaging Multiplexor Instances” on page 77. 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.
- POP before SMTP service for legacy POP clients. For more information, see “Enabling POP Before SMTP,” on page 423.

How the Messaging 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 determines the correct server for the users, connects to that server, and then passes data between the client and server. 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 4-1 shows how servers and clients relate to each other in an MMP installation.

Figure 4-1 Clients and Servers in an MMP Installation



All POP, IMAP, and SMTP 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.

When SSL is enabled, the MMP IMAP and SMTP services support STARTTLS and the MMP can also be configured to listen on additional ports for SSL IMAP, POP, and SMTP connections.

To enable SSL encryption for your IMAP, POP, and SMTP services, edit the `ImapProxyAService.cfg`, `PopProxyAService.cfg`, and `SmtpProxyAService.cfg` files, respectively. You must also edit the `default:ServiceList` option in the `AService.cfg` file to include the list of all IMAP, POP, and SMTP 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 install an SSL server certificate. Then, you should uncomment 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 a certificate mapping file (`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 73.

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 73.

3. Configure the MMP to use certificate-based client authentication by specifying the location of the `certmap.conf` file in your configuration files.
4. Install at least one trusted CA certificate, as described in "To Install Certificates of Trusted CAs," on page 404.

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`*hex*`) user` *name* `pre-authenticated - client IP` *address,* `server IP` *address*

Where *date* is in the format `yyyymmdd`, *time* is in UTC (Standard Coordinated Universal Time, also known as GMT (Greenwich Mean Time)) in the format `hhmmss`, *hex* is the session identifier (`sid`) represented as a hexidecimal number, the `user` *name* includes the virtual domain (if any), and the IP address is in dot-quad format.

MMP Virtual Domains

An MMP virtual domain is a set of configuration settings associated with a server IP address. The primary use of this feature is to provide different default domains for each server IP address.

A user can authenticate to MMP with either a short-form userID or a fully qualified userID in the form `user@domain`. When a short-form userID is supplied, the MMP will append the `DefaultDomain` setting, if specified. Consequently, a site which supports multiple hosted domains can permit the use of short-form userIDs simply by associating a server IP address and MMP virtual domain with each hosted domain.

The recommended method for locating the user subtree for a given hosted domain is via the i`netDomainBaseDN` attribute in the LDAP domain tree entry for that domain. The MMP's `LdapUrl` setting is not suitable for this purpose since the back-end mail store servers will also need to look up the user in LDAP and do not support virtual domains.

To enable virtual domains, edit the `ImapProxyAService.cfg`, `PopProxyAService.cfg`, or `SmtpProxyAService.cfg` file(s) in the instance directory such that the `VirtualDomainFile` setting specifies the full path 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 simply used to associate the IP address with the configuration parameters and can be any 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`
`CertMap`
`ClientLookup`
`CRAMs`
`DefaultDomain`
`DomainDelim`
`HostedDomains`
`LdapCacheSize` and `LdapCacheTTL`
`LdapURL`
`MailHostAttrs`
`PreAuth`
`ReplayFormat`
`StoreAdmin` and `StoreAdminPass`
`SearchFormat`
`TCPAccess`
`TCPAccessAttr`

NOTE Unless the `LdapURL` is correctly set, the `BindDN`, `BindPass`, `LdapCacheSize` and `LdapCacheTTL` settings will be ignored.

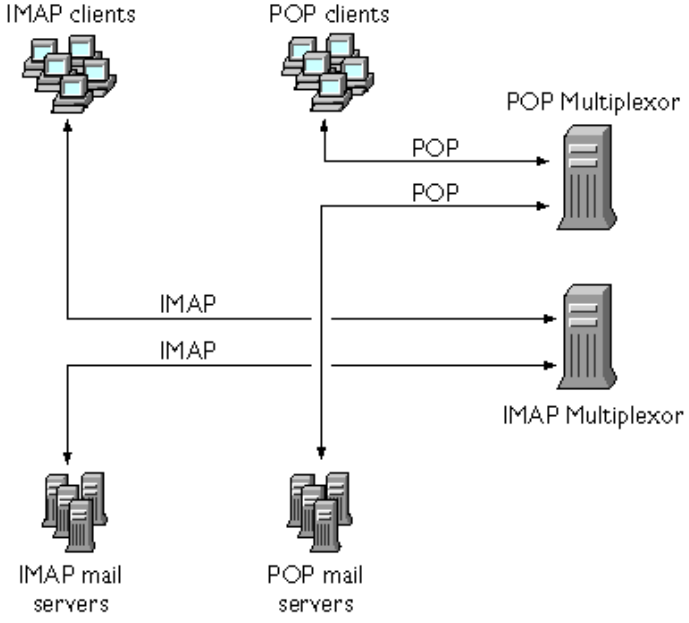
For detailed descriptions of these configuration parameters, see the *iPlanet Messaging Server Reference Manual*.

Multiple Messaging Multiplexor Instances

You can install multiple instances of the MMP on a single server. Each of these instances will run as a separate process and can have different configuration files. Multiple instances are necessary when you need different settings for different server IP addresses or ports, and those settings are ones that cannot be changed by a virtual domain. The SSL server certificate is an example of such a setting.

You can configure a single instance of the MMP to support both POP, IMAP, and SMTP protocols (as shown in Figure 4-1), or you can create separate MMP instances for each protocol, as shown in Figure 4-2. By splitting messaging services across different machines, you can tune the resources on each computer for maximum performance.

Figure 4-2 Separate MMP Instances for Each Protocol



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

About SMTP Proxy

The MMP includes an SMTP proxy which is disabled by default. Most sites do not need the SMTP proxy because Internet Mail standards already provide an adequate mechanism for horizontal scalability of SMTP (DNS MX records).

The SMTP proxy is useful for the security features it provides. First, the SMTP proxy is integrated with the POP proxy to implement the POP before SMTP authorization facility required by some legacy POP clients. For more information, see "Enabling POP Before SMTP," on page 423.

In addition, an investment in SSL acceleration hardware can be maximized by using the SMTP proxy. See "How to Optimize SSL Performance Using the SMTP Proxy," on page 410.

Configuring Messaging Multiplexor

To configure the Messaging Multiplexor, you must manually edit the configuration parameters in the Messaging Multiplexor configuration files listed in Table 4-1.

Table 4-1 Messaging Multiplexor Configuration Files

File	Description
`PopProxyAService.cfg`	Configuration file specifying configuration variables used for POP services.
`PopProxyAService-def.cfg`	POP services configuration template. If the `PopProxyAService.cfg` file does not exist, the `PopProxyAService-def.cfg` template is copied to create a new `PopProxyAService.cfg` file.
`ImapProxyAService.cfg`	Configuration file specifying configuration variables used for IMAP services.
`ImapProxyAService-def.cfg`	IMAP services configuration template. If the `ImapProxyAService.cfg` file does not exist, the `ImapProxyAService-def.cfg` template is copied to create a new `ImapProxyAService.cfg` file.
`AService.cfg`	Configuration file specifying which services to start and a few options shared by both POP and IMAP services.
`AService-def.cfg`	Configuration template specifying which services to start and a few options shared by both POP and IMAP services. If the `AService.cfg` file does not exist, the `AService-def.cfg` template is copied to create a new `AService.cfg` file.

Table 4-1 Messaging Multiplexor Configuration Files *(Continued)*

File	Description
`AService.rc`	Script used to start, stop, restart, and reload the MMP. To enable automatic startup of the MMP after reboot, the `AService.rc` script can be copied to `/etc/init.d` and symbolically linked to the appropriate `/etc/rc?.d` directories. For more information about initialization and termination scripts, refer to the man page on `init.d`.
`SmtpProxyAService.cfg`	Optional configuration file specifying configuration variables used for SMTP Proxy services. Required if you enable POP before SMTP; useful for maximizing support for SSL hardware even if POP before SMTP is not enabled. For more information on POP before SMTP, see “Enabling POP Before SMTP,” on page 423.
`SmtpProxyAService-def.cfg`	Configuration template specifying configuration variables used for SMTP Proxy services. If the `SmtpProxyAService.cfg` file does not exist, the `SmtpProxyAService-def.cfg` template is copied to create a new `SmtpProxyAService.cfg` file.

The Messaging 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 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 parameters for POP-related events. In `SmtpProxyAService.cfg`, they are used to specify logging for SMTP Proxy-related events.

In `AService.cfg`, however, `LogDir` and `LogLevel` are used for logging MMP-wide failures, such as the failure to start a POP, IMAP, or SMTP service.

NOTE When the MMP is installed or upgraded, the configuration template files will be overwritten.

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

To Start Messaging 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 4-2.

Table 4-2 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 4-3.

Table 4-3 Windows NT MMP Service Options

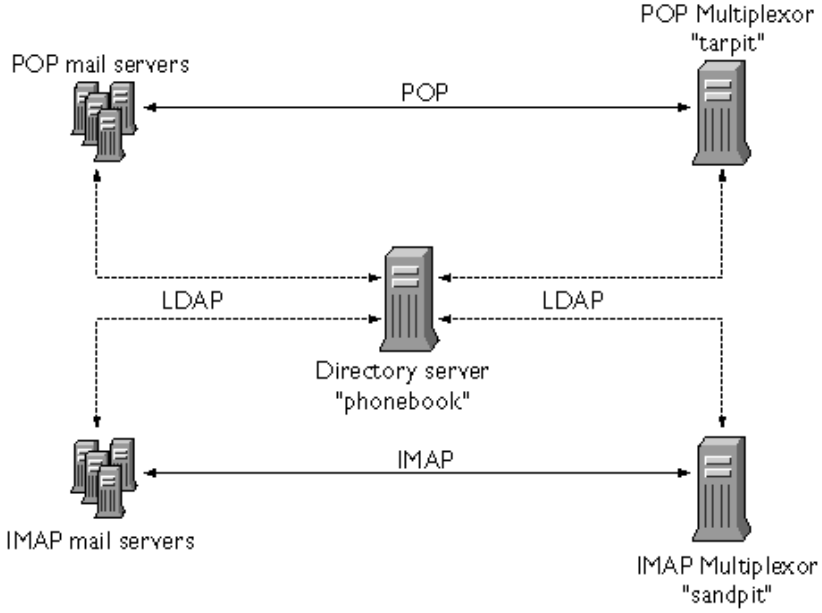
Option	Description
`start`	At the Control Panel, start the MMP (even if one is already running), or, at the command line run the command `AService.exe start`
`stop`	At the Control Panel, stop the most recently started MMP, or, at the command line run the command `AService.exe stop`
`restart`	To restart 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 Messaging 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 `ImapProxyAService` entry from the `ServiceList` setting; likewise, you can restrict client access to IMAP services alone by removing the `PopProxyAService` entry from the `ServiceList` setting.). Each Messaging 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 4-3.

Figure 4-3 Multiple MMPs Supporting Multiple Messaging Servers



IMAP Configuration Example

The IMAP Messaging Multiplexor in Figure 4-3 is installed on `sandpit`, a machine with two processors. This Messaging Multiplexor is listening to the standard port for IMAP connections (143). Messaging 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 Messaging Multiplexor example in Figure 4-3 is installed on `tarpit`, a machine with four processors. This Messaging Multiplexor is listening to the standard port for POP connections (110). Messaging 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
```

About Messenger Express Multiplexor

The iPlanet Messenger Express Multiplexor is a specialized server that acts as a single point of connection to the HTTP access service. Messenger Express is the client interface to the iPlanet Messaging Server HTTP service. All users connect to the single messaging proxy server, which directs them to their appropriate mailbox. As a result, an entire array of messaging servers will appear to your mail users to be a single host.

While the iPlanet Messaging Multiplexor (MMP) connects to POP and IMAP servers, the Messenger Express Multiplexor connects to an HTTP server. In other words, the Messenger Express Multiplexor is to Messenger Express as MMP is to POP and IMAP.

Like MMP, the Messenger Express Multiplexor supports:

- Both unencrypted and encrypted (SSL) communication with mail clients

  For more information on configuring SSL, refer to Security and Access Control in Chapter 12.

- Hosted Domains

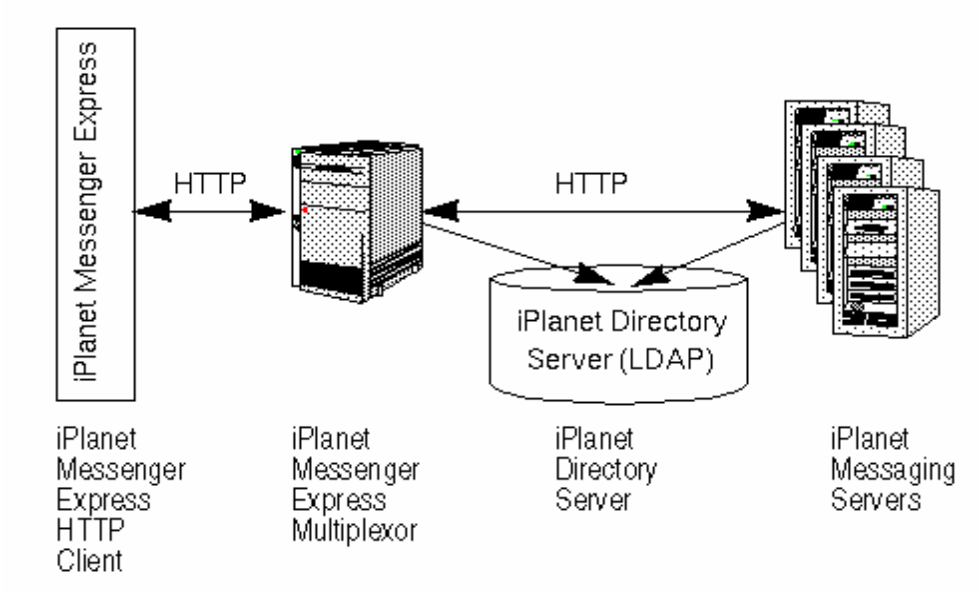
  For more information, refer to the *iPlanet Messaging Server Provisioning Guide*.

Unlike MMP, the Messenger Express Multiplexor is built into the `mshttpd` service, and consequently uses the same logging and configuration mechanisms.

How Messenger Express Multiplexor Works

The Messenger Express Multiplexor is made up of a proxy messaging server that acts as a Multiplexor; it allows you to connect to the HTTP service of iPlanet Messaging Server (Messenger Express). The Messenger Express Multiplexor facilitates distributing mailboxes across multiple server machines. Clients connect to the Multiplexor when logging onto iPlanet Messenger Express, which determines the correct server for the users, connects to that server, and then passes data between the client and server. This capability allows 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 4-4 on page 86 describes where the Messenger Express Multiplexor resides in an iPlanet Messaging Server installation.

Figure 4-4 Overview of iPlanet Messenger Express Multiplexor



The Messenger Express Multiplexor interfaces between iPlanet Messenger Express client and iPlanet Messaging Servers by accepting connections and routing them 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 HTTP connections are routed through the Messenger Express Multiplexor.

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

1. A user's client connects to the Messenger Express Multiplexor, which accepts preliminary authentication information.
2. The Messenger Express Multiplexor queries Directory Server to determine which messaging server contains the user's mailbox.
3. The Messenger Express Multiplexor connects to the associated Messaging Server, replays authentication, then acts as a pass-through pipe for the duration of the session.

To Set Up the Messenger Express Multiplexor

This section will describe the steps you should follow to set up and configure your Messenger Express Multiplexor. Topics that are covered include:

Install Messaging Server on Proxy Machine

The first step is to install Messaging Server on the proxy machine that will become the Messenger Express Multiplexor. For specific installation instructions, see the *iPlanet Messaging Server Installation Guide*.

Be sure to configure the Messaging Server to a users and groups directory server that points to the back-end messaging servers. This directory server will be used to authenticate users to Messenging Server through the Messenger Express Multiplexor.

To Configure Multiplexor Parameters

After you complete the Messaging Server installation on the proxy machine, configure the Messenger Express Multiplexor parameters:

1. Gather the needed back-end Messaging Server information.

   Run the `configutil` command in the directory of your back-end messaging servers to determine the values of the parameters that are later described in this section. The configuration of the proxy machine (where the Multiplexor will be enabled) must match the back-end messaging servers to ensure successful setup.

2. Set the configuration parameters for the Messenger Express Multiplexor.

   Run the `configutil` command in directory *server_root*`/bin/msg-`*instance*`/configutil` of your proxy machine messaging server to set the configuration values. Note that these values should match the values of the back-end messaging servers.

   For a detailed explanation on running the `configutil` command, see the *iPlanet Messaging Server Reference Manual*.

The following sections describe the `configutil` parameters needed to set up the Messenger Express Multiplexor:

- "LDAP Parameters," on page 88
- "dcroot," on page 88
- "Default Domain," on page 88
- "Login Separator," on page 89

LDAP Parameters

You will need to make sure that the Directory Server parameters are correctly specified prior to enabling the Messenger Express Multiplexor. To determine your LDAP parameters, run the following command in the appropriate back-end Messaging Server instance directory:

- `configutil -o local.ugldaphost`

  This parameter displays the users and groups LDAP Directory Server that the back-end messaging servers use. Make sure that `ldaphost` is set to the same value (or a replicated LDAP server containing the same data) that the back-end messaging servers use.

- `configutil -o local.ugldapbinddn`
  `configutil -o local.ugldapbindcred`

  These parameters display the DN and password of the users and groups Directory Server administrator. Both `ldapbinddn` and `ldapbindcred` must be the same as in your back-end messaging servers specifications.

dcroot

You will need to make sure that the *dcroot* is correctly specified. To determine your *dcroot*, run the following command in the appropriate messaging server instance directory:

```
configutil -o service.dcroot
```

Default Domain

You will need to make sure that the messaging server default domain (*defaultdomain*) is correctly indicated. To determine your messaging server default domain, run the following `configutil` command in the appropriate messaging server instance directory:

```
configutil -o service.defaultdomain
```

Login Separator

Make sure that the login separator (*loginseparator*) is consistent with the login separator used by the back-end messaging server. To determine your messaging server login separator, run the `configutil` command in the appropriate back-end messaging server instance directory:

```
configutil -o service.loginseparator
```

Enable Messenger Express Multiplexor

Once you set the configuration parameters, you can enable the Messenger Express Multiplexor on the proxy machine. To do so, run the following `configutil` command in the directory *server_root*`/bin/msg-`*instance*`/configutil` of the messaging server instance on the proxy machine:

```
configutil -o local.service.http.proxy -v 1
```

where `1` enables the Messenger Express Multiplexor (default `0`).

When a non-local user (users whose mail host is not on the server where they log in) logs in and the value of `local.service.http.proxy` is `0`, the user will be directed to his host, and the user will see the host name change; therefore, the Multiplexor is not enabled.

If the value of `local.service.http.proxy` is set to `1`, the Multiplexor is enabled, the host name does not change, and the entire array of messaging servers will appear to be a single host to your non-local mail users.

For local users (users whose mail host is the server where they log in), the server will use the local message store regardless of the `local.service.http.proxy` parameter value. It is possible to have both proxy and local users coexisting on the same messaging server.

For more information on the `configutil` command, see the *iPlanet Messaging Server Reference Manual*.

Testing Your Setup

In this section, you will learn how to test your Messenger Express Multiplexor setup and to look for messages in the log files. It is assumed that you have configured and enabled the Messenger Express Multiplexor.

Access Messenger Express Client

Prior to testing your installation, you should already be familiar with the Messenger Express product. In addition, you should already have a test account that you have previously created.

To test your Messenger Express Multiplexor proxy, follow these steps:

1. Through the Messenger Express Multiplexor, connect to Messenger Express by typing in the brower location:

   `http://`*msgserver_name* in the browser location.

   For example:

   `http://budgie.sesta.com`

2. Using a test account that you previously created, log in to Messenger Express.
3. You should be able to successfully log in and access messages from the back-end messaging servers.
4. If the messaging server name changes once you log in through Messenger Express, make sure `local.service.http.proxy` is set to `1` and that you have restarted the messaging proxy server. The Messenger Express Multiplexor should provide the appearance of a single mail host to your users.

Error Messages

If you receive an error message when you enter the user id, password, and click Connect, you should review the HTTP log file of the proxy machine. To view the error messages, go to the *server_root/*`msg-`*instance*`/log/http/` directory. In most cases, the error message will contain sufficient information to diagnose the problem. In those instances where there is not sufficient information to diagnose the problem, contact iPlanet Customer Support.

Administering Your Messenger Express Multiplexor

This section describes the basic administration capabilities of the Messenger Express Multiplexor.

Configure and Administer SSL

To configure and administer SSL (otherwise known as Secure Sockets Layer) for your Messenger Express Multiplexor, refer to “To Enable SSL and Selecting Ciphers,” on page 406.

Multiple Proxy Server Setup

To set up multiple Messenger Express Multiplexors that are addressed by a single name, you can use a session-aware load balancing device. With this device, all requests can be routed from any given client to a unique server.

Managing Different Versions of Messaging Server and Messenger Express Multiplexor

If you use different versions of iPlanet Messaging Server for the Messenger Express Multiplexor and the back-end mail hosts, you need to update the Messenger Express static files to ensure compatibility between the servers.

The static files which make up the Messenger Express interface are served directly from the Messenger Express Multiplexor, not user’s mail host. The Multiplexor finds these files in the *server_root*/msg-*instance*/html directory.

To update these files in order to ensure compatibility between servers, replace the entire contents (which consist of these static files that make up the Messenger Express interface) of the directory *server_root*/msg-*instance*/html in the newer version of Messaging Server with the entire contents of the same directory in the older version of Messaging Server.

For example, if the back-end messaging servers use iPlanet Messaging Server 5.1 and you have installed iPlanet Messaging Server 5.2 as the Messenger Express Multiplexor, you need to replace the entire contents of the directory *server_root*/msg-*instance*/html of the Messenger Express Multiplexor with the contents of the same directory from the iPlanet Messaging Server 5.1 back-end server. When you eventually upgrade iPlanet Messaging Server 5.1 to iPlanet Messaging Server 5.2, you can update these static files in directory *server_root*/msg-*instance*/html for the Messenger Express Multiplexor server as well.

Chapter 5

MTA Concepts

This chapter provides a conceptual description of the MTA. It consists of the following sections:

The MTA Functionality

The Message Transport Agent, or *MTA* (Figure 5-2 on page 95), is a component of the Messaging Server (Figure 5-1 on page 94)that performs the following functions:

- **Message Routing** - accepts messages and routes it to: A) another SMTP host, B) a local message store, or C) to the a program for processing (example: virus checking).
- **Message Blocking** - blocks or accepts messages based on specified source and/or destination IP address, mail address, port, channel, or by header strings.
- **Address Rewriting** - rewrite incoming `From:` or `To:` addresses to a more desired form.

- **Message Processing** - perform different types of message processing. For example:
  - Expand aliases
  - Control SMTP command and protocol support
  - Provide SASL support
  - Hold messages when the number of addresses exceeds a specified limit
  - Deliver messages to site-supplied program such as virus checkers and mail filing programs
  - Perform message-part-by-message-part conversion on messages
  - Customize of delivery status notification messages

Figure 5-1 iPlanet Messaging Server, Simplified Components View (Messenger Express not Shown)

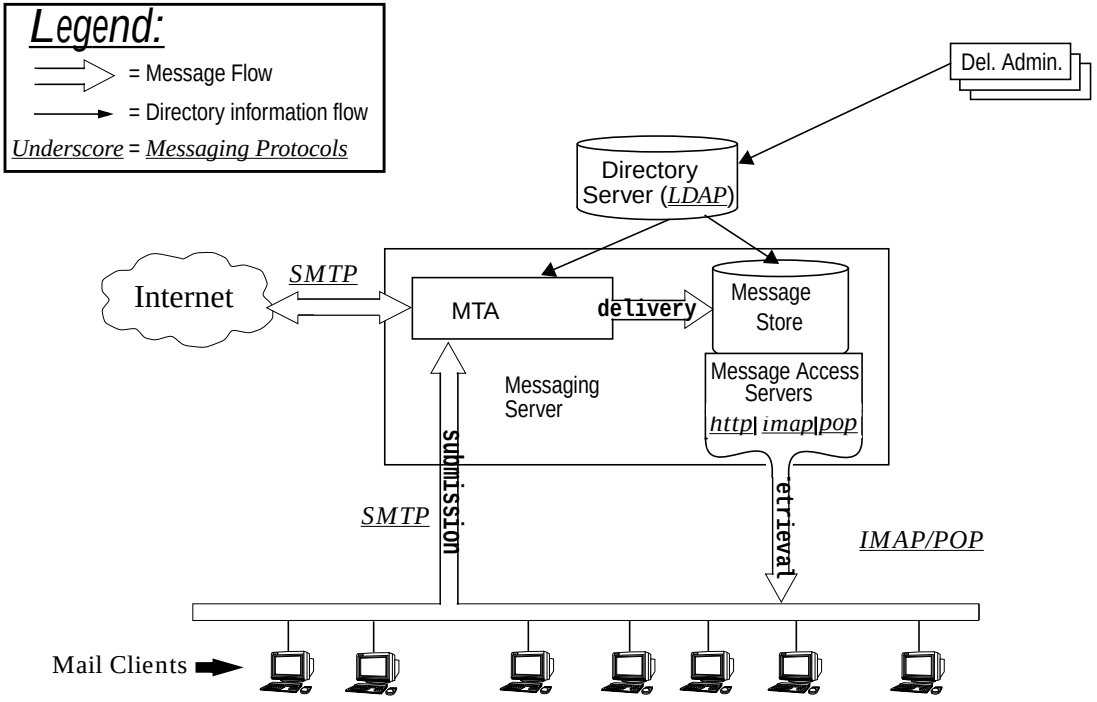
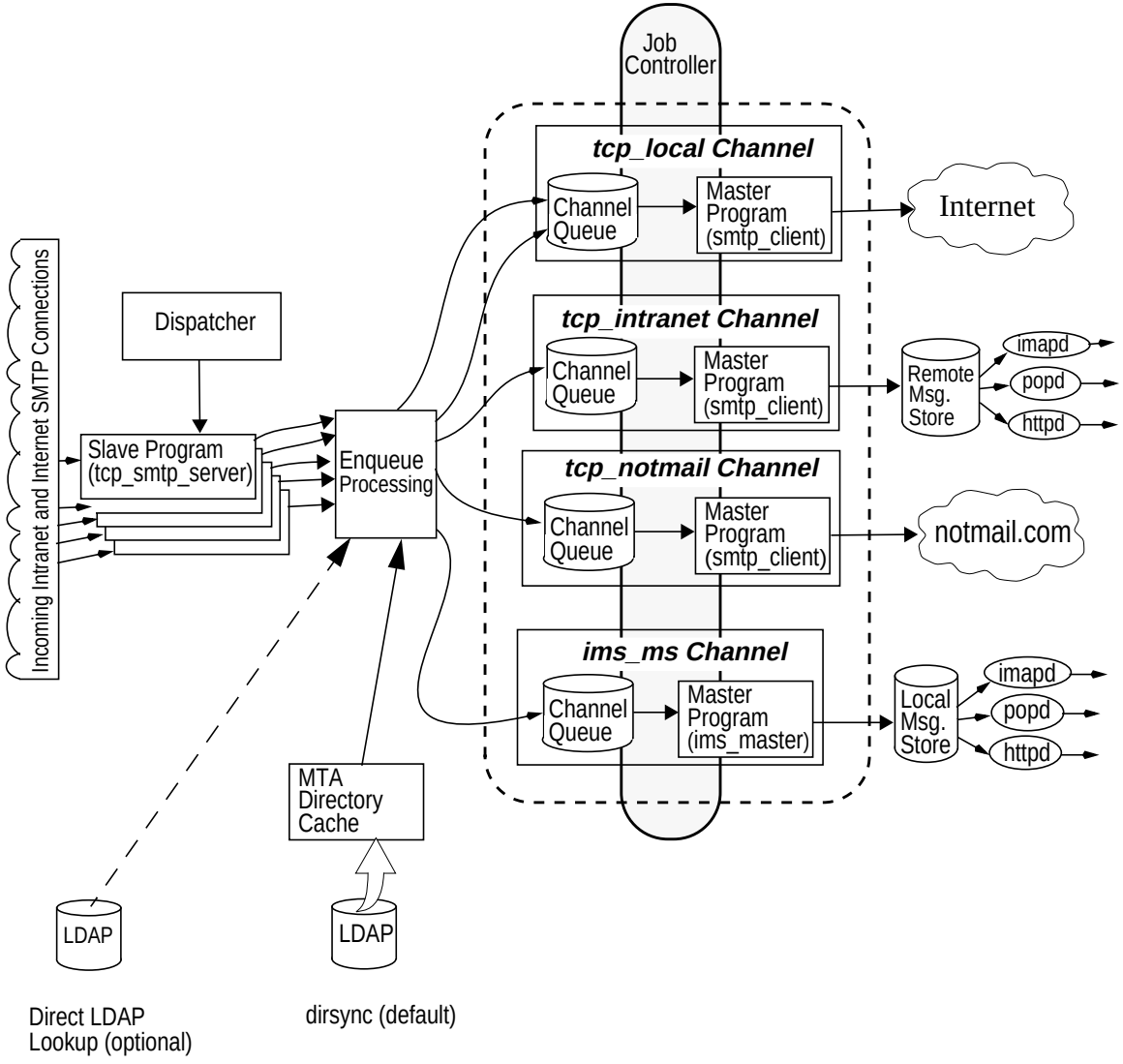


Figure 5-2 MTA Architecture



MTA Architecture and Message Flow Overview

This section provides a short overview of MTA architecture and message flow (Figure 5-2).

Dispatcher and SMTP Server (Slave Program)

Messages enter the MTA from the internet or intranet via SMTP sessions. When the MTA receives a request for an SMTP connection, the MTA *dispatcher* (a multithreaded connection dispatching agent), executes a *slave* program (`tcp_smtp_server`) to handle the SMTP session. As additional sessions are requested, the dispatcher activates an SMTP server program to handle each session. Together the dispatcher and slave program perform a number of different functions on each incoming message. Three primary functions are:

- Message blocking - messages from specified IP addresses, mail addresses, ports, channels, header strings and so on, may be blocked (Chapter 10, “Mail Filtering and Access Control.”)
- Address changing. Incoming `From:` or `To:` addresses may be rewritten to a different form.
- Channel enqueueing. Addresses are run through the rewrite rules to determine which channel the message should be sent.

For more information see “The Dispatcher” on page 98

Enqueue

A number of tasks are achieved during this phase of delivery, but the primary tasks are:

- Alias expansion.
- Running the addresses through the rewrite rules to determine to which channel the message should be enqueued and to rewriting the domain part of addresses into their proper or desired format.
- Channel keyword processing.
- Sending the message to the appropriate channel queue.

Channels

The channel is the fundamental MTA component used for message processing. A channel represents a message connection with another system (for example, another MTA, another channel, or the local message store). As mail comes in, different messages require different routing and processing depending on the message’s source and destination. For example, mail to be delivered to a local

message store will be processed differently from mail to be delivered to the internet which will be processed differently from mail to be sent to another MTA within the mail system. Channels provide the mechanism for customizing the processing and routing required for each connection. In a default installation, the majority of messages go to a channels handling internet, intranet, and local messages.

Specialized channels for specific situations can also be created. For example, suppose that a certain internet domain processes mail very slowly causing mail addressed to this domain to clog up the MTA. A special channel could be created to provide special handling for messages addressed to the slow domain, thus relieving the system of this domain bottleneck.

The domain part of the address determines to what channel the message is enqueued. The mechanism for reading the domain and determining the appropriate channel is called the rewrite rules (see “Rewrite Rules,” on page 100).

Channels typically consist of a channel queue and a channel processing program called a *master program*. After the slave program delivers the message to the appropriate channel queue, the master program performs the desired processing and routing. Channels, like rewrite rules, are specified and configured in the `imta.cnf` file. An example of a channel entry is shown below:

```
tcp_intranet smtp mx single_sys subdirs 20 noreverse maxjobs 7
SMTP_POOL  maytlsserver allowswitchchannel  saslswitchchannel
tcpauth
tcpintranet-daemon
```

The first word, in this case `tcp_intranet` is the channel name. The last word is called the channel tag. The words in between are called channel keywords and specify how messages are to be processed. Hundreds of different keywords allow messages to be processed in many ways. A complete description of channel keywords is provided in the *iPlanet Messaging Server Reference Manual* and Chapter 8, “Configuring Channel Definitions.”

Message Delivery

After the message is processed, the master program sends the message to the next stop along the message’s delivery path. This may be the intended recipient’s mailbox, another MTA, or even a different channel. Forwarding to another channel is not shown in the picture, but is common occurrence.

The Dispatcher

The Dispatcher is a multithreaded dispatching agent that permits multiple multithreaded server processes to share responsibility for SMTP connection services. When using the Dispatcher, it is possible to have several multithreaded SMTP server processes running concurrently, all handling connections to the same port. 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.

See “Dispatcher Configuration File,” on page 131 for more information.

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 131.

To Start and Stop the Dispatcher

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.

Rewrite Rules

Rewrite rules determine the following:

- How to rewrite the domain part of an address into its proper or desired format.
- To which channel the message should be enqueued after the address is rewritten.

Each rewrite rule consists of a *pattern* and a *template*. The pattern is a string to match against the domain part of an address. The template specifies the actions to take if the domain part matches the pattern. It consists of two things: 1) a set of instructions (that is, a string of control characters) specifying how the address should be rewritten and 2) the name of the channel to which the message shall be sent. After the address is rewritten, the message is enqueued to the destination channel for delivery to the intended recipient.

An example of a rewrite rule is shown below:

```
siroe.com               $U%$D@tcp_siroe-daemon
```

`siroe.com` is the domain pattern. Any message with the address containing `siroe.com` will be rewritten as per the template instructions (`$U%$D`). `$U` specifies that the rewritten address use the same user name. % specifies that the rewritten address use the same that the domain separator. `$D` specifies that the rewritten address use the same domain name that was matched in the pattern. `@tcp_siroe-daemon` specifies that the message with its rewritten address be sent to the the channel called `tcp_siroe-daemon`. See Chapter 7, “Configuring Rewrite Rules” for more details.

For more information about configuring rewrite rules, see “The MTA Configuration File,” on page 109 and Chapter 7, “Configuring Rewrite Rules.”

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.

Master and Slave Programs

Generally (but not always), a channel is associated with two programs: master and slave. The slave program accepts messages from another system and adds them to a channel's message queue. The master program transfers messages from the channel to another system.

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.

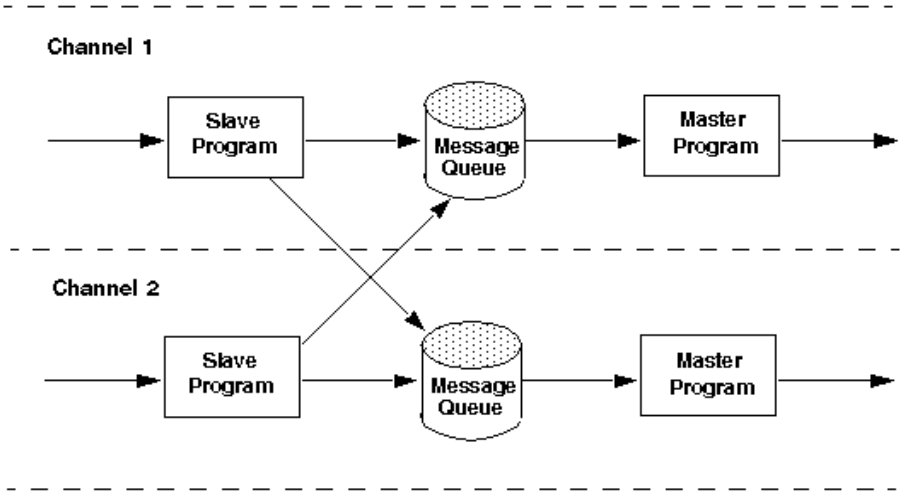
The slave channel program typically accepts incoming connections where the MTA is responding to an external request. The slave channel program:

- 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.

For example, Figure 5-3 shows two channel programs, Channel 1 and Channel 2. The slave program in Channel 1 receives a message from a remote system. It looks at the address, applies rewrite rules as necessary, then based on the rewritten address enqueues the message to the appropriate channel message queue.

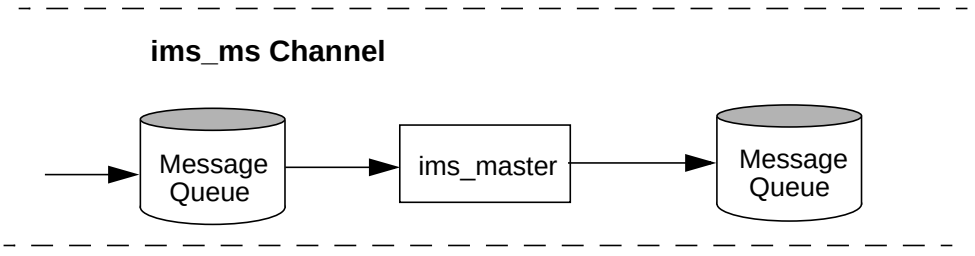
The master program dequeues the message from the queue and initiates network transport of the message. Note that the master program can only dequeue messages from its own channel queue.

Figure 5-3 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 dequeuing messages to the local message store, as shown in Figure 5-4.

Figure 5-4 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*/*.

Channel Definitions

Channel definitions appear in the lower half of the MTA configuration file, imta.cnf, following the rewrite rules (see “The MTA Configuration File” on page 109). 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
Channel_Name  keyword1  keyword2
Channel_Tag
[blank line]
```

Collectively, the channel definitions are referred to as the channel host table. An individual channel definition is called a channel block. For example, in Figure 5-5 the channel host table contains three channel definitions or blocks.

Figure 5-5 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
```

A typical channel entry will look something like this:

```
tcp_intranet smtp mx single_sys subdirs 20 noreverse maxjobs 7
SMTP_POOL  maytlsserver allowswitchchannel  saslswitchchannel
tcpauth
tcpintranet-daemon
```

The first word, in this case `tcp_intranet`, is the channel name. The last word, in this case `tcpintranet-daemon`, is called the *channel tag*. The channel tag is the name used by rewrite rules to direct messages. The words in between the channel name and channel tag are called channel *keywords* and specify how the message is to be processed. Hundreds of different keywords allow messages to processed in many ways. A complete listing of channel keywords is listed and described in the *iPlanet Messaging Server Reference Manual* and Chapter 8, “Configuring Channel Definitions.”

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.

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 on the option files, see “Option File,” on page 133, and “TCP/IP (SMTP) Channel Option Files,” on page 130. For details on configuring channels, see Chapter 8, “Configuring Channel Definitions.” For more information about creating MTA channels, see “The MTA Configuration File,” on page 109.

The MTA Directory Information

For each message that it processes, the MTA needs to access directory information about the users, groups, and domains that it supports. This information is stored in an LDAP directory service. The MTA has two methods of accessing this information. The first is by directly accessing the LDAP directory. This is called the *direct LDAP mode* and is fully described in Appendix B, “MTA Direct LDAP Operation.” The second and default method for accessing directory information is through the *directory cache*. This is called the `dirsync` mode.

In the `dirsync` mode, directory information about users and groups used by the MTA is accessed through a number of files and databases collectively called the directory cache. The data itself is stored in the LDAP directory, but actual information is accessed from the cache. Data in the cache is updated by the `dirsync` program which monitors changes to the LDAP directory and updates the files and databases accordingly.

Details on `dirsync` operation and configuration are described in “dirsync Configuration” on page 112

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. When the job limit is no longer exceeded, the Job Controller starts another job.

Channel jobs run inside processing pools within the Job Controller. A pool can be thought of a “place” where the channel jobs are run. The pool provides a computing area where a set of jobs can operate without vying for resources with jobs outside of the pool. For more information on pools see “Job Controller File,” on page 134 and “Processing Pools for Channel Execution Jobs” on page 240.)

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 normally attempts to deliver all messages immediately. 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’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 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 mean that starts or restarts of the Job Controller will incur more overhead than starts or restarts when no such backlog exists.

The Job Controller also runs a number of periodic jobs. These jobs are configured in the Job Controller configuration rather than using a more general facility such as cron so that the scheduling of these jobs is dependent on the Job Controller being up and running. This is an important point for high availability configurations where failover is a consideration.

For information about pools and configuring the Job Controller, see “Job Controller File,” on page 134 and “Configuring Message Processing and Delivery,” on page 235.

To Start and Stop the Job Controller

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.

Chapter 6

About MTA Services and Configuration

This chapter describes general MTA services and configuration. More specific and detailed explanations may be found in other chapters. It consists of the following sections:

The MTA Configuration File

The primary MTA configuration file is `imta.cnf`. By default, this file is found at *instance_root*`/imta/config/imta.cnf`. This file contains MTA channel definitions as well as the channel rewrite rules. 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 configuration file consists of two parts: domain rewriting rules and channel definitions. The domain rewriting rules appear first in the file and are separated from the channel definitions by a blank line. The channel definitions are collectively referred to as the channel table. An individual channel definition forms a channel block.

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 6-1 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.
!
! Part I: Rewrite rules
a     $U@a-daemon (2)
b     $U@b-daemon
c     $U%c@b-daemon
d     $U%d@a-daemon
              (3)
! Part II: Channel definitions
l             (4)
local-host

a_channel defragment charset7 usascii (5)
a-daemon

b_channel noreverse notices 1 2 3
b-daemon

</usr/iplanet/server5/msg-tango/table/internet.rules (6)
```

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. These definitions are collectively referred to as the *channel host table*, which defines the channels that the MTA can use and the names associated with each channel.
4. The first channel block to appear is usually the local or `l` channel. Blank lines then separate each channel block from one another. (An exception is the `defaults` channel, which can appear before the `l` channel).

5. A typical channel definition consists of a channel name (`a_channel`) some keywords which define the configuration of a channel (`defragment charset7 usascii`) and a routing system (`a-daemon`), which is also called a *channel tag*.

6. The contents of other files may be included in the configuration file. If a line is encountered with a less than sign (<) in column one, the rest of the line is treated as a file name; the file name should always be an absolute and full file path. The file is opened and its contents are spliced into the configuration file at that point. Include files may be nested up to three levels deep. Any files included in the configuration file must be world-readable just as the configuration file is world-readable.

Table 6-1 shows how some example addresses would be routed 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

Refer to “Rewrite Rules” on page 100, “Channel Definitions” on page 103, and Chapter 7, “Configuring Rewrite Rules” for more information on the MTA configuration file.

dirsync Configuration

The default Messaging Server installation uses the `dirsync` mode of operation. (The alternative is to use the direct LDAP mode which is described in Appendix B, “MTA Direct LDAP Operation.) In the dirsync mode, rather than querying the directory service each time it processes a message, the MTA caches the directory information and accesses the cache for the data required.

The directory information stored in the directory service is continuously updated by a program called `dirsync`. As a result, the directory cache must be updated periodically—that is, synchronized—with the current directory information in the directory service. There are 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 periodically 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 6-2 shows which updates occur on full and partial directory synchronizations.

Table 6-2 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

*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 removal of the directory entry 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*.

Directory Synchronization Configuration Parameters

Table 6-3 lists the directory synchronization configuration parameters.

Table 6-3 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 `mailAlternateAddress` 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,mailAlternateAddress, rfc822mailbox,rfc822mailalias`

Table 6-3 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”

Mapping File

Many components of the MTA employ table lookup-oriented information. This type 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 (left-hand) column provides possible input strings against which to match (pattern), and the second (right-hand) column gives the resulting output string for which the input string is mapped (template).

Most of the MTA databases—databases that contain different types of MTA data and which should not be confused with mapping tables—are instances of just this type of 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 MTA mapping file supports multiple mapping tables. Wildcard capabilities are provided, as well as multistep and iterative mapping methods. 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 serve to eliminate the need for most of the entries in an equivalent database, and this may result in lower overhead overall.

Table 6-4 lists the mapping tables described in this book.

Table 6-4 iPlanet Messaging Server Mapping Tables

Mapping Table	Page	Description
CHARSET-CONVERSION	297	Used to specify what sorts of channel-to-channel character set conversions and message reformatting should be done.
COMMENT_STRINGS	253	Used to modify address header comments (strings enclosed in parentheses).
CONVERSIONS	280	Used to select message traffic for the conversion channel.
“domain lookup”	546	Used for finding the base of the tree in which to search for aliases in direct LDAP mode.
FORWARD	148	Used to perform forwarding similar to that performed using the alias file or alias database.
FROM_ACCESS	306	Used to filter mail based on envelope From addresses. Use this table if the To address is irrelevant.
INTERNAL_IP	317	Used to recognize systems and subnets that are internal.

Table 6-4 iPlanet Messaging Server Mapping Tables

Mapping Table	Page	Description
MAIL_ACCESS	306	Used to block incoming connections based on combined information found in SEND_ACCESS and PORT_ACCESS table.
NOTIFICATION_LANGUAGE	149	Used to customize or localize notification messages.
ORIG_MAIL_ACCESS	306	Used to block incoming connections based on combined information found in ORIG_SEND_ACCESS and PORT_ACCESS tables
ORIG_SEND_ACCESS	306	Used to block incoming connections based on envelope From address, envelope To address, source and destination channels.
PERSONAL_NAMES	254	Used to modify personal names (strings preceding angle-bracket-delimited addresses).
PORT_ACCESS	306	Used to block incoming connections based on IP number.
REVERSE	144	Used to convert addresses from an internal form to a public, advertised form.
SEND_ACCESS	306	Used to block incoming connections based on envelope From address, envelope To address, source and destination channels.
X-ATT-NAMES	289	Used to retrieve a parameter value from a mapping table.

Locating and Loading the Mapping File

Mapping tables are kept in the MTA mapping file. This is the file specified with the IMTA_MAPPING_FILE option in the MTA tailor file; by default, this is *server_root*/msg-*instance*/imta/config/mappings. The contents of the mapping file is incorporated into the compiled configuration.

The mapping file should be world readable. Failure to allow world-read access leads to erratic behavior.

File Format in the Mapping File

The mapping file consists of a series of separate tables. Each table begins with its name. Names always have an alphabetic character in the first column. The table name is followed by a required blank line, and then by the entries in the table. Entries consist of zero or more indented lines. Each entry line consists of two columns separated by one or more spaces or tabs. Any spaces within an entry must be quoted using the $ character. A blank line must appear after each mapping table name and between each mapping table; no blank lines can appear between entries in a single table. Comments are introduced by an exclamation mark (!) in the first column.

The resulting format looks like:

```
TABLE1_NAME

   pattern1-1    template1-1
   pattern1-2    template1-2
   pattern1-3    template1-3
      .             .
      .             .
      .             .
   pattern1-n    template1-n

TABLE2_NAME

   pattern2-1    template2-1
   pattern2-2    template2-2
   pattern2-3    template2-3
       .            .
       .            .
       .            .
   pattern2-n    template2-n

          .
          .
          .

TABLE3_NAME

          .
          .
          .
```

An application using the mapping table `TABLE2_NAME` would map the string `pattern2-2` into whatever is specified by `template2-2`. Each pattern or template can contain up to 252 characters. There is no limit to the number of entries that can appear in a mapping (although excessive numbers of entries may consume huge amounts of CPU and can consume excessive amounts of memory). Long lines (over 252 characters) may be continued by ending them with a backslash (\). The white space between the two columns and before the first column may not be omitted.

Duplicate mapping table names are not allowed in the mapping file.

Including Other Files in the Mapping File

Other files may be included in the mapping file. This is done with a line of the form:

```
<file-spec
```

This effectively substitutes the contents of the file `file-spec` into the mapping file at the point where the include appears. The file specification should specify a full file path (directory, and so forth). All files included in this fashion must be world readable. Comments are also allowed in such included mapping files. Includes can be nested up to three levels deep. Include files are loaded at the same time the mapping file is loaded—they are not loaded on demand, so there is no performance or memory savings involved in using include files.

Mapping Operations

All mappings in the mapping file are applied in a consistent way. The only things that change from one mapping to the next is the source of input strings and what the output from the mapping is used for.

A mapping operation always starts off with an input string and a mapping table. The entries in the mapping table are scanned one at a time from top to bottom in the order in which they appear in the table. The left side of each entry is used as pattern, and the input string is compared in a case-blind fashion with that pattern.

Mapping Entry Patterns

Patterns can contain wildcard characters. In particular, the usual wildcard characters are allowed: an asterisk (*) matches zero or more characters, and each percent sign (%) matches a single character. Asterisks, percent signs, spaces, and tabs can be quoted by preceding them with a dollar sign ($). Quoting an asterisk or percent sign robs it of any special meaning. Spaces and tabs must be quoted to prevent them from ending prematurely a pattern or template. Literal dollar sign characters should be doubled ($$), the first dollar sign quoting the second one.

Table 6-5 Mapping Pattern Wildcards

Wildcard	**Description**
%	Match exactly one character.
*	Match zero or more characters, with maximal or “greedy” left-to-right matching
Back match	**Description**
$ n*	Match the nth wildcard or glob.
Modifiers	**Description**
$_	Use minimal or “lazy” left-to-right matching.
$@	Turn off “saving” of the succeeding wildcard or glob.
$^	Turn on “saving” of the succeeding wildcard or glob; this is the default.
Glob wildcard	**Description**
$A%	Match one alphabetic character, A-Z or a-z.
$A*	Match zero or more alphabetic characters, A-Z or a-z.
$B%	Match one binary digit (0 or 1).
$B*	Match zero or more binary digits (0 or 1).
$D%	Match one decimal digit 0-9.
$D*	Match zero or more decimal digits 0-9.
$H%	Match one hexadecimal digit 0-9 or A-F.
$H*	Match zero or more hexadecimal digits 0-9 or A-F.
$O%	Match one octal digit 0-7.
$O*	Match zero or more octal digits 0--7.
$S%	Match one symbol set character, for example, 0-9, A-Z, a-z, _, $.
$S*	Match zero or more symbol set characters, that is, 0-9, A-Z, a-z, _, $.

Table 6-5 Mapping Pattern Wildcards *(Continued)*

$T%	Match one tab or vertical tab or space character.
$T*	Match zero or more tab or vertical tab or space characters.
$X%	A synonym for $H%.
$X*	A synonym for $H*.
$[c]%	Match character c.
$[c]*	Match arbitrary occurrences of character c.
$[c_1 c_2 ... c_n]%	Match exactly one occurrence of character c_1, c_2, or c_n.
$[c_1 c_2 ... c_n]*	Match arbitrary occurrences of any characters c_1, c_2, or c_n.
$[c_1 -c_n]%	Match any one character in the range c_1 to c_n.
$[c_1 -c_n]*	Match arbitrary occurrences of characters in the range c_1 to c_n.
$< IPv4>	Match an IPv4 address, ignoring bits.
$(IPv4)	Match an IPv4 address, keeping prefix bits.
${IPv6}	Match an IPv6 address.

Within globs, that is, within a `$[...]` construct, the backslash character, `\`,is the quote character. To represent a literal hyphen, `-`, or right bracket, `]`, within a glob the hyphen or right bracket must be quoted with a backslash.

All other characters in a pattern just represent and match themselves. In particular, single and double quote characters as well as parentheses have no special meaning in either mapping patterns or templates; they are just ordinary characters. This makes it easy to write entries that correspond to illegal addresses or partial addresses.

To specify multiple modifiers, or to specify modifiers and a back match, the syntax uses just one dollar character. For instance, to back match the initial wild card, without saving the back match itself, one would use $@0, not $@$0.

Note that the `imsimta test -match` utility may be used to test mapping patterns and specifically to test wildcard behavior in patterns.

Asterisk wildcards maximize what they match by working from left to right across the pattern. For instance, when the string `a/b/c` is compared to the pattern `*/*`, the left asterisk matches `a/b` and the right asterisk matches the remainder, `c`.

The $_ modifier causes wildcard matching to be minimized, where the least possible match is considered the match, working from left to right across the pattern. For instance, when the string a/b/c is compared to the pattern $_*/$_*, the left $_* matches a and the right $_* matches b/c.

IP Matching

With IPv4 prefix matching, an IP address or subnet is specified, optionally followed by a slash and the number of bits from the prefix that are significant when comparing for a match. For example, the following matches anything in the 123.45.67.0 subnet:

```
$(123.45.67.0/24)
```

With IPv4 ignore bits matching, an IP address or subnet is specified, optionally followed by a slash and the number of bits to ignore when checking for a match. For example, the following matches anything in the 123.45.67.0 subnet:

```
$<123.45.67.0/8>
```

The following example matches anything in the range 123.45.67.4 through 123.45.67.7:

```
$<123.45.67.4/2>
```

IPv6 matching matches an IPv6 address or subnet.

Mapping Entry Templates

If the comparison of the pattern in a given entry fails, no action is taken; the scan proceeds to the next entry. If the comparison succeeds, the right side of the entry is used as a template to produce an output string. The template effectively causes the replacement of the input string with the output string that is constructed from the instructions given by the template.

Almost all characters in the template simply produce themselves in the output. The one exception is a dollar sign ($).

A dollar sign followed by a dollar sign, space, or tab produces a dollar sign, space, or tab in the output string. Note that all these characters must be quoted in order to be inserted into the output string.

A dollar sign followed by a digit *n* calls for a substitution; a dollar sign followed by an alphabetic character is referred to as a “metacharacter.” Metacharacters themselves do not appear in the output string produced by a template, but produce some special substitution or processing. See Table 6-6 for a list of the special substitution and standard processing metacharacters. Any other metacharacters are reserved for mapping-specific applications.

Note that any of the metacharacters $C, $E, $L, or $R, when present in the template of a matching pattern, influences the mapping process and control whether it terminates or continues. That is, it is possible to set up iterative mapping table entries, where the output of one entry becomes the input of another entry. If the template of a matching pattern does not contain any of the metacharacters $C, $E, $L, or $R, then $E (immediate termination of the mapping process) is assumed.

The number of iterative passes through a mapping table is limited to prevent infinite loops. A counter is incremented each time a pass is restarted with a pattern that is the same length or longer than the previous pass. If the string has a shorter length than previously, the counter is reset to zero. A request to reiterate a mapping is not honored after the counter has exceeded 10.

Table 6-6 Mapping Template Substitutions and Metacharacters

Substitution sequence	**Substitutes**
$n	The *n*th wildcarded field as counted from left to right starting from 0.
$#...#	Sequence number substitution.
$]...[	LDAP search URL lookup; substitute in result.
$|...|	Applies specified mapping table to supplied string.
${...}	General database substitution.
$[...]	Invokes site-supplied routine; substitute in result.
Metacharacter	**Description**
$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.
$E	Ends the mapping process now; uses the output string from this entry as the final result of the mapping process.
$L	Continues the mapping process starting with the next table entry; use the output string of this entry as the new input string; after all entries in the table are exhausted, makes one more pass, starting with the first table entry. A subsequent match may override this condition with a $C, $E, or $R metacharacter.
$R	Continues the mapping process starting with the first entry of the mapping table; uses the output string of this entry as the new input string for the mapping process.
$?x?	Mapping entry succeeds x percent of the time.
$\	Forces subsequent text to lowercase.

Table 6-6 Mapping Template Substitutions and Metacharacters *(Continued)*

Substitution sequence	Substitutes
$^	Forces subsequent text to uppercase.
$_	Leaves subsequent text in its original case.
$:*x*	Match only if the specified flag is set.
$;*x*	Match only if the specified flag is clear.

Wildcard Field Substitutions ($n)

A dollar sign followed by a digit n is replaced with the material that matched the *n*th wildcard in the pattern. The wildcards are numbered starting with 0. For example, the following entry would match the input string `PSI%A::B` and produce the resultant output string `b@a.psi.siroe.com`:

```
PSI$%*::*    $1@$0.psi.siroe.com
```

The input string `PSI%1234::USER` would also match producing `USER@1234.psi.siroe.com` as the output string. The input string `PSIABC::DEF` would not match the pattern in this entry and no action would be taken; that is, no output string would result from this entry.

Controlling Text Case ($\, $^, $_)

The metacharacter $\ forces subsequent text to lowercase, $^ forces subsequent text to uppercase, and $_ causes subsequent text to retain its original case. For instance, these metacharacters may be useful when using mappings to transform addresses for which case is significant.

Processing Control ($C, $L, $R, $E)

The $C, $L, $R, and $E metacharacters influence the mapping process, controlling whether and when the mapping process terminates. The metacharacter:

- $C causes the mapping process to continue with the next entry, using the output string of the current entry as the new input string for the mapping process.

- $L causes the mapping process to continue with the next entry, using the output string of the current entry as the new input string for the mapping process, and, if no matching entry is found, making one more pass through the table starting with the first table entry; a subsequent matching entry with a $C, $E, or $R metacharacter overrides this condition.
- $R causes the mapping process to continue from the first entry of the table, using the output string of the current entry as the new input string for the mapping process.
- $E causes the mapping process to terminate; the output string of this entry is the final output. $E is the default.

Mapping table templates are scanned left to right. To set a $C, $L, or $R flag for entries that may "succeed" or "fail" (for example, general database substitutions or random-value controlled entries), put the $C, $L, or $R metacharacter to the left of the part of the entry that may succeed or fail; otherwise, if the remainder of the entry fails, the flag is not seen.

Entry Randomly Succeeds or Fails ($?x?)

The metacharacters $?x? in a mapping table entry cause the entry to "succeed" *x* percent of the time; the rest of the time, the entry "fails" and the output of the mapping entry's input is taken unchanged as the output. (Note that, depending upon the mapping, the effect of the entry failing is not necessarily the same as the entry not matching in the first place.)The *x* should be a real number specifying the success percentage.

For instance, suppose that a system with IP address 123.45.6.78 is sending your site just a little too much SMTP email and you'd like to slow it down; you can use a PORT_ACCESS mapping table in the following way. Suppose you'd like to allow through only 25 percent of its connection attempts and reject the other 75 percent of its connection attempts. The following PORT_ACCESS mapping table uses $?25? to cause the entry with the $Y (accept the connection) to succeed only 25 percent of the time; the other 75 percent of the time, when this entry fails, the initial $C on that entry causes the MTA to continue the mapping from the next entry, which causes the connection attempt to be rejected with an SMTP error and the message: Try again later.

```
PORT_ACCESS

  TCP|*|25|123.45.6.78|*          $C$?25?$Y
  TCP|*|25|123.45.6.78|*          $N45s$ 4.40$ Try$ again$ later
```

Sequence Number Substitutions ($#...#)

A $#. . .# substitution increments the value stored in an MTA sequence file and substitutes that value into the template. This can be used to generate unique, increasing strings in cases where it is desirable to have a unique qualifier in the mapping table output; for instance, when using a mapping table to generate file names.

Permitted syntax is any one of the following:

$#*seq-file-spec* | *radix* | *width*#

$#*seq-file-spec* | *radix*#

$#*seq-file-spec*#

The required *seq-file-spec* argument is a full file specification for an already existing MTA sequence file, where the optional *radix* and *width* arguments specify the radix (base) in which to output the sequence value, and the number of digits to output, respectively. The default radix is 10. Radices in the range -36 to 36 are also allowed; for instance, base 36 gives values expressed with digits 0,...,9,A,...,Z. By default, the sequence value is printed in its natural width, but if the specified width calls for a greater number of digits, then the output is padded with 0's on the left to obtain the correct number of digits.

Note that if a width is explicitly specified, then the radix must be explicitly specified also.

As noted above, the MTA sequence file referred to in a mapping must already exist. To create an MTA sequence file, use the following UNXI command:

touch *seq-file-spec*

or

```
cat >seq-file-spec
```

A sequence number file accessed using a mapping table must be world readable in order to operate properly. You must also have an MTA user account (configured to be nobody in the `imta_tailor` file) in order to use such sequence number files.

LDAP query URL substitutions, $]...[

A substitution of the form `$]`*ldap-url*`[` is specially handled. *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 part are instead specified with the `LDAP_HOST` and `LDAP_PORT` options. That is, the LDAP URL should be specified as:

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

where the square bracket characters [and] shown above indicate optional portions of the URL. 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 the information to return. That is, *attributes* specifies the attribute or attributes to be returned from LDAP directory entries matching this LDAP query. The *scope* may be any of `base` (the default), `one`, or `sub`. *filter* describes the characteristics of matching entries.

Certain LDAP URL substitution sequences are available for use within the LDAP query URL.

Mapping Table Substitutions ($|...|)

A substitution of the form `$|`*mapping*`;`*argument*`|` is handled specially. The MTA looks for an auxiliary mapping table named *mapping* in the MTA mapping file, and uses *argument* as the input to that named auxiliary mapping table. The named auxiliary mapping table must exist and must set the `$Y` flag in its output if it is successful; if the named auxiliary mapping table does not exist or doesn't set the `$Y` flag, then that auxiliary mapping table substitution fails and the original mapping entry is considered to fail: the original input string is used as the output string.

Note that when you want to use processing control metacharacters such as `$C`, `$R`, or `$L` in a mapping table entry that does a mapping table substitution, the processing control metacharacter should be placed to the left of the mapping table substitution in the mapping table template; otherwise the “failure” of a mapping table substitution means that the processing control metacharacter is not seen.

General Database Substitutions (${...})

A substitution of the form ${*text*} is handled specially. The *text* part is used as a key to access the general database. This database is generated with the `imsimta crdb` utility. If *text* is found in the database, the corresponding template from the database is substituted. If *text* does not match an entry in the database, the input string is used unchanged as the output string.

If a general database exists, it should be world readable to insure that it operates properly.

When you want to use processing control metacharacters such as $C, $R, or $L in a mapping table entry that does a general database substitution, the processing control metacharacter should be placed to the left of the general database substitution in the mapping table template; otherwise the "failure" of a general database substitution means that the processing control metacharacter is not seen.

Site-Supplied Routine Substitutions ($[...])

A substitution of the form `$[`*image*`,`*routine*`,`*argument*`]` is handled specially. The `image`, `routine`, `argument` part is used to find and call a customer-supplied routine. At runtime on UNIX, the MTA uses `dlopen` and `dlsym` to dynamically load and call the routine *routine* from the shared library *image*. At runtime on Windows NT, the MTA calls the routine *routine* from the dynamic link library *image*. The routine *routine* is then called as a function with the following argument list:

```
status = routine (argument, arglength, result, reslength)
```

The `argument` and `result` are 252-byte long character string buffers. The `argument` and `result` are passed as a pointer to a character string (for example, in C, as `char*`). The `arglength` and `reslength` are signed, long integers passed by reference. On input, `argument` contains the *argument* string from the mapping table 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 then replaces the `$[image,routine,argument]` in the mapping table template. The *routine* routine should return 0 if the mapping table substitution should fail and -1 if the mapping table substitution should succeed. If the substitution fails, then normally the original input string is used unchanged as the output string.

If you want to use processing control metacharacters such as `$C`, `$R`, or `$L` in a mapping table entry that does a site-supplied routine substitution, you place the processing control metacharacter to the left of the site-supplied routine substitution in the mapping table template; otherwise, the “failure” of a mapping table substitution means that the processing control metacharacter is not seen.

The site-supplied routine callout mechanism allows the MTA's mapping process to be extended in all sorts of complex ways. For example, in a `PORT_ACCESS` or `ORIG_SEND_ACCESS` mapping table, a call to some type of load monitoring service could be performed and the result used to decide whether or not to accept a connection or message.

The site-supplied shared library image `image` should be world readable.

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 6-7.

Table 6-7 MTA Configuration Files

File	Description
Autoreply Option File	Options used by the `autoreply` program. *instance_root*`/imta/config/autoreply_option`
Alias File (mandatory)	Implements aliases not present in the directory. *instance_root*`/imta/config/aliases`
TCP/IP (SMTP) Channel Option Files (also called SMTP Option Files)	Sets channel-specific options. *instance_root*`/imta/config/`*channel*`_option`
Conversion File	Used by the conversion channel to control message body part conversions. *instance_root*`/imta/config/conversions`
Dirsync Option File (mandatory only if running in dirsync mode)	Options used by the `dirsync` program. *instance_root*`/imta/config/dirsync.opt`
Dispatcher Configuration File (mandatory)	Configuration file for the Dispatcher. *instance_root*`/imta/config/dispatcher.cnf`

Table 6-7 MTA Configuration Files

File	Description
Job Controller File (mandatory)	Configuration file used by the `Job Controller`. /*instance_root*/imta/config/job_controller.cnf
MTA Configuration File (mandatory)	Used for address rewriting and routing as well as channel definition. /*instance_root*/imta/config/imta.cnf
Mapping File (mandatory)	Repository of mapping tables. /*instance_root*/imta/config/mappings
Option File	File of global MTA options. /*instance_root*/imta/config/option.dat
Tailor File (mandatory)	File to specify locations and some tuning parameters. /*instance_root*/imta/config/imta_tailor

Autoreply Option File

The autoreply option file, `autoreply_option`, sets options for the autoreply or vacation program. For details see the *iPlanet 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 141.

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

TCP/IP (SMTP) 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, /*ServerInstance*/config/imta/tcp_local_option. For more information refer to "Configuring SMTP Channel Options," on page 215. For complete information on 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 Conversion Channel” on page 278.

Dirsync Option File

The dirsync option file, `dirsync.opt`, sets options for the `dirsync` program that cannot be set through the command line. See “dirsync Configuration” on page 112 and the *iPlanet Messaging Server Reference Manual* for details.

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. (For conceptual information, see “The Dispatcher” on page 98.)

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 “Mapping File” on page 116 and the *Messaging Server Reference Manual*.

Option File

The options file, `option.dat`, specifies global MTA options as opposed to channel-specific 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 *ServerInstance*`/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 complete information on this file, see the *Messaging Server Reference Manual*.

Job Controller File

The Job Controller creates and manages channel jobs for delivering messages. These channel jobs run inside processing pools within the Job Controller. A pool can be thought of a "place" where the channel jobs are run. The pool provides a computing area where a set of jobs can operate without vying for resources with jobs outside of the pool. (For information on Job Controller concepts and channel keyword configuration, refer to "The Job Controller" on page 105, "Processing Pools for Channel Execution Jobs" on page 240, and "Service Job Limits" on page 240.)

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

- 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.

Code Example 6-1 shows a sample Job Controller configuration file. Table 6-8 shows the available options.

Code Example 6-1 Sample Job Controller Configuration File in UNIX

```
!MTA Job Controller configuration file
!
!Global defaults
tcp_port=27442          (1)
secret=never mind
```

Code Example 6-1 Sample Job Controller Configuration File in UNIX

```
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_.

Example of Adding Additional Pools

The Job Controller creates and manages channel jobs for delivering messages. These channel jobs run inside processing pools within the Job Controller. A pool can be thought of a "place" where the channel jobs are run. The pool provides a computing area where a set of jobs can operate without vying for resources with jobs outside of the pool. Note that the job limit that is set in the job_controller is per pool. So, for example, if you have your SMTP_POOL defined with a job_limit of 10, then only 10 tcp_smtp client processes can run in that pool at any given time.

There are situations where one may want to create additional tcp_* channels (say, for example, a tcp channel for particularly slow mail sites). It is a good idea to have these channels run in different pools. The reason for this is if we created ten different tcp_* channels and they were all running in SMTP_POOL, it is possible to only have one tcp_smtp client running per tcp_* channel at any given time (depending on whether or not there is mail destined for all the tcp_* channels and given that SMTP_POOL defined with a job_limit of 10). Assuming there is heavy load on the system and that all queues have messages waiting to go out the various tcp_* channels, this would not be efficient. It is much more likely that one would want to define additional pools for the additional tcp_* channels so that there is no contention for slots.

For example, suppose we set up the following `tcp_*` channels:

```
tcp_yahoo smtp mx pool yahoo_pool keyword keyword keyword
tcp-yahoo-daemon

tcp_aol smtp mx keyword keyword keyword pool aol_pool
tcp-aol-daemon

tcp_hotmail smtp mx pool hotmail_pool keyword keyword keyword
tcp-hotmail-daemon

...

tcp_sun smtp mx pool sun_pool keyword keyword keyword
tcp-sun-daemon
```

In order to add have ten `tcp_smtp_client` processes for each of the new channels we would add the following in the `job_controller.cnf` file:

```
[POOL=yahoo_pool]
job_limit=10

[POOL=aol_pool]
job_limit=10

[POOL=hotmail_pool]
job_limit=10

...

[POOL=sun_pool]
job_limit=10
```

For more information about pools, see “Processing Pools for Channel Execution Jobs” on page 240. For more information about the syntax of the Job Controller file, see the *Messaging Server Reference Manual*.

Table 6-8 Job Controller Configuration File Options

Option	Description
General Options	**Description**
INTERFACE_ADDRESS=*adapter*	Specifies the IP address interface to which the Job Controller should bind. The value specified (adapter) can be one of ANY, ALL, LOCALHOST, or an IP address. By default the Job Controller binds to all addresses (equivalent to specifying ALL or ANY). Specifying INTERFACE_ADDRESS=LOCALHOST means that the Job Controller only accepts connections from within the local machine. This does not affect normal operation, since no inter-machine operation is supported by the Job Controller. However, this may be inappropriate in an HA environment where an HA agent may be checking if the Job Controller is responding. If the machine on which the Messaging Server is running is in an HA environment, has an "internal network" adapter and an "external network" adapter, and you are not confident of your firewall's ability to block connections to high port numbers, you should consider specifying the IP address of the "internal network" adapter.
MAX_MESSAGES=*integer*	The Job Controller keeps information about messages in an in-memory structure. In the event that a large backlog builds, it may need to limit the size of this structure. If the number of messages in the backlog exceeds the parameter specified here, information about subsequent messages is not kept in memory. Mail messages are not lost because they are always written to disk, but they are not considered for delivery until the number of messages known by the Job Controller drops to half this number. At this point, the Job Controller scans the queue directory mimicking an imsimta cache -sync command. The default is 100000.
SECRET=*file_spec*	Shared secret used to protect requests sent to the Job Controller.
SYNCH_TIME=*time_spec*	The Job Controller occasionally scans the queue files on disk to check for missing files. By default, this takes place every four hours, starting four hours after the Job Controller is started. The format of the *time_spec* is *HH*:*MM*/*hh*:*mm* or /*hh*:*mm*. The variable *hh*.*mm* is the interval between the events in hours (*h*) and minutes (*m*). The variable *HH*:*MM* is the first time in a day the even should take place. For example specifying, 15:45/7:15 starts the event at 15:45 and every seven hours and fifteen minutes from then.

Table 6-8 Job Controller Configuration File Options *(Continued)*

Option	Description
TCP_PORT=*integer*	Specifies the TCP port on which the Job Controller should listen for request packets. Do not change this unless the default conflicts with another TCP application on your system. If you do change this option, change the corresponding IMTA_JBC_SERVICE option in the MTA tailor file, *server_root*/msg-*instance*/imta/config/imta_tailor, so that it matches. The TCP_PORT option applies globally and is ignored if it appears in a [CHANNEL] or [POOL] section.
TIME=*time_spec*	Specifies the time and frequency that a periodic job is run in a PERIODIC_JOB section. By default, this is /4:00, which means every four hours. The format of *time_spec* is *HH*:*MM*/*hh*:*mm* or /*hh*:*mm*. *hh.mm* is the interval between the events in hours (*h*) and minutes (*m*). *HH*:*MM* is the first time in a day that a job should occur. For example, specifying 15:45/7:15 starts the event at 15:45 and every seven hours and fifteen minutes from then.
Pool Option	**Description**
JOB_LIMIT=*integer*	Specifies the maximum number of processes that the pool can use simultaneously (in parallel). The JOB_LIMIT applies to each pool individually; the maximum total number of jobs is the sum of the JOB_LIMIT parameters for all pools. If set outside of a section, it is used as the default by any [POOL] section that doesn't specify JOB_LIMIT. This option is ignored inside of a [CHANNEL] section.
Channel Option	**Description**
MASTER_COMMAND=*file_spec*	Specifies the full path to the command to be executed by the UNIX system process created by the Job Controller to run the channel and dequeue messages outbound on that channel. If set outside of a section, it is used as the default by any [CHANNEL] section that doesn't specify a MASTER_COMMAND. This option is ignored inside of a [POOL] section.
MAX_LIFE_AGE=*integer*	Specifies the maximum life time for a channel master job in seconds. If this parameter is not specified for a channel, then the global default value is used. If no default value is specified, 1800 (30 minutes) is used.
MAX_LIFE_CONNS=*integer*	In addition to the maximum life age parameter, the life expectancy of a channel master job is limited by the number of times it can ask the Job Controller if there are any messages. If this parameter is not specified for a channel, then the global default value is used. If no default value is specified, 300 is used.

Table 6-8 Job Controller Configuration File Options *(Continued)*

Option	Description
SLAVE_COMMAND=*file_spec*	Specifies the full path to the command to be executed by the UNIX system process created by the Job Controller in order to run the channel and poll for any messages inbound on the channel. Most MTA channels do not have a SLAVE_COMMAND. If that is the case, the reserved value NULL should be specified. If set outside of a section, it is used as the default by any [CHANNEL] section that doesn't specify a SLAVE_COMMAND. This option is ignored inside of a [POOL] section.

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.

NOTE This section primarily describes alias handling in the dirsync mode. Alias resolution in the direct LDAP mode is described "Resolving Addresses Using the Direct LDAP Rewrite Rule ($V)," on page 540.

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 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.

NOTE Messaging Server provides other facilities for address manipulation, such as the address reversal database and specialized mapping tables. For best performance, however, rewrite rules should be used whenever possible to perform address manipulations. See Chapter 7, “Configuring Rewrite Rules.”

A physical line in this file is limited to 1024 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. 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*.

SMTP Security and Access Control

For information about SMTP security and access control, see Chapter 10, “Mail Filtering and Access Control,” and Chapter 12, “Configuring Security and Access Control.”

Log Files

All MTA specific log files are kept in the MTA log directory, (*ServerInstance*/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 13, “Logging and Log Analysis.”

To Convert Addresses from an Internal Form to a Public Form

Addresses can be converted from an internal form to a public, advertised form using the Address-Reversal database and the REVERSE mapping table. For example, while uid@mailhost.siroe.com might be a valid address within the siroe.com domain, it might not be an appropriate address to expose to the outside world. You may prefer a public address like firstname.lastname@siroe.com.

NOTE Messaging Server provides other facilities for address manipulation, such as the aliases file and specialized mapping tables. For best performance, however, rewrite rules should be used whenever possible to perform address manipulations. See Chapter 7, “Configuring Rewrite Rules.”

In the reverse database, the public address for each user is specified by the mail attribute of the user entry in the directory. The private or internal address is specified by the mailAlternativeAddress attribute. The same is true for distribution lists.

The reverse database contains a mapping between any valid address and this public address. The reverse database is updated and created each time the `imsimta dirsync` command is run. If you have enabled direct MTA LDAP operation (see Appendix B, "MTA Direct LDAP Operation"), then the Address-Reversal Database is not used.

The reverse database is generally located in the MTA database directory. The database is the files whose names are specified with the `IMTA_REVERSE_DATABASE` option in the *server_root*`/msg-`*instance*`/imta/config/imta_tailor` file, which by default are the files *server_root*`/msg-`*instance*`/imta/db/reversedb.*`.

If an address is found in the database, the corresponding right side from the database is substituted for the address. If the address is not found, an attempt is made to locate a mapping table named `REVERSE` in the mapping file. No substitution is made, and rewriting terminates normally if the table does not exist or no entries from the table match.

If a `REVERSE` mapping table is found in the mapping file, and if the address matches a mapping entry, the resulting string replaces the address if the entry specifies a `$Y`. A `$N` discards the result of the mapping. If the mapping entry specifies `$D` in addition to `$Y`, the resulting string runs through the reversal database once more; and if a match occurs, the template from the database replaces the mapping result (and hence the address). The form of general `REVERSE` mapping table entries (that is, entries that apply to all channels) is shown below. Note that flags can be either in front of the new address or at the end.

```
REVERSE

  OldAddress        $Y[Flags]NewAddress
```

The form of *channel-specific* entries (that is, mapping that only occurs only on messages passing through a specific channel) is shown below. Note that you must set `use_reverse_database` to 13 in the `option.dat` for channel-specific entries to work.

```
REVERSE

  source-channel|destination-channel|OldAddress  $Y[Flags]NewAddresS
```

`REVERSE` mapping table flags as shown in Table 6-9.

Table 6-9 REVERSE mapping table flags

Flags	Description
$Y	Use output as new address.
$N	Address remains unchanged.
$D	Run output through the reversal database.
$A	Add pattern as reverse database entry.
$F	Add pattern as forward database entry.
Flag comparison	**Description**
$:B	Match only header (body) addresses.
$:E	Match only envelope addresses.
$:F	Match only forward pointing addresses.
$:R	Match only backwards pointing addresses.
$:I	Match only message-ids.

To Set Address Reversal Controls

The `reverse` and `noreverse` channel keywords, and the MTA options `USE_REVERSE_DATABASE` and `REVERSE_ENVELOPE` are used to control the specifics of when and how address reversal is applied. By default, the address reversal operation applies to all addresses, not just to backward pointing addresses.

Address reversal can be enabled or disabled by setting the value of the `REVERSE_ENVELOPE` system option (Default: 1-on, 0-off).

`noreverse` on the destination channel specifies that address reversal is not applied to addresses in messages. `reverse` specifies that address reversal is applied. See *iPlanet Messaging Server Reference Manual* for details.

`USE_REVERSE_DATABASE` controls whether the MTA uses the address reversal database and `REVERSE` mapping as a source of substitution addresses. A value of 0 means address reversal is not used with any channel. A value of 5 (default) specifies that address reversal is applied to all addresses—not just to backward pointing addresses— after they have been rewritten by the MTA address rewriting process. A value of 13 specifies that address reversal is applied to addresses with the `reverse` channel keyword—not just to backward pointing addresses— after

they have been rewritten by the MTA address rewriting process. Further granularity of address reversal operation can be specified by setting the bit values of the USE_REVERSE_DATABASE option. See *iPlanet Messaging Server Reference Manual* for details.

The REVERSE_ENVELOPE option controls whether or not address reversal is applied to envelope From addresses as well as message header addresses.

See the detailed descriptions of these options and keywords in the *iPlanet Messaging Server Reference Manual* for additional information on their effects.

General Reverse Mapping Example

An example of a general REVERSE Mapping is as follows: suppose that the internal addresses at siroe.com are of the form user@mailhost.siroe.com. However, the user name space is such that user@host1.siroe.com and user@host2.siroe.com specify the same person for all hosts at siroe.com. The following REVERSE mapping may be used in conjunction with the address-reversal database:

```
REVERSE

  *@*.siroe.com        $0@siroe.com$Y$D
```

In this example, addresses of the form *name@anyhost*.siroe.com would be changed to *name*@siroe.com. The $D metacharacter causes the address-reversal database to be consulted. The address-reversal database should contain entries of the form:

```
user@mailhost.siroe.com     first.last@siroe.com
```

Channel-Specific Reverse Mapping Example

By default, the address reversal database is used if the routability scope is set to the mail server domains. An example of a channel-specific REVERSE mapping table entry would be as follows:

```
REVERSE

  tcp_*|tcp_local|binky@macho.siroe.com    $D$YRebecca.Woods@siroe.com
```

This entry tells the MTA that for any mail with source channel of `tcp_*` going out the destination channel of `tcp_local` to change addresses of the form `binky@macho.siroe.com` to `Rebecca.Woods@siroe.com`

NOTE To enable channel-specific reverse mapping, you must set `USE_REVERSE_DATABASE` option in `option.dat` to 13. (Default=5.)

FORWARD Address Mapping

Address reversals are not applied to envelope `To` addresses. These addresses are continuously rewritten and modified as messages proceed through the mail system. The entire goal of routing is to convert envelope `To` addresses to increasingly system- and mailbox-specific formats. The canonization functions of address reversal are inappropriate for envelope `To` addresses.

The various substitution mechanisms for envelope `To` addresses provide functionality equivalent to the reversal database, but none of these things provides functionality equivalent to reverse mapping. Circumstances can arise where mapping functionality for envelope `To` addresses is useful and desirable.

The `FORWARD` mapping table provides this missing functionality. If a `FORWARD` mapping table exists in the mapping file, it is applied to each envelope `To` address. No changes are made if this mapping does not exist or no entries in the mapping match.

If the address matches a mapping entry, the result of the mapping is tested. The resulting string replaces the envelope To address if the entry specifies a `$Y`; a `$N` discards the result of the mapping.

Using the Forward Database to Forward Mail

The forward database can be used to perform forwarding similar to that performed using the alias file or alias database. But when the alias file or alias database can be used, their use is generally preferable to using the forward database as their use is more efficient.

The sort of case where use of the forward database for forwarding mail is appropriate is generally occurs when different sorts of forwarding need to be performed depending upon the source of the message being forwarded. Forward database forwarding can be made source specific, via the `USE_FORWARD_DATABASE` option. See the *iPlanet Messaging Server Reference Manual* for more information.

Controlling Delivery Status Notification Messages

Delivery status notifications or *notification messages* are email status messages sent by the MTA to the sender and, optionally, the postmaster. Messaging Server allows you to customize notification messages by content and language. You can also create different messages for each type of delivery status (for example, `FAILED`, `BOUNCED`, `TIMEDOUT`, etc.). In addition, you can create notification messages for messages originating from specific channels.

By default, notification messages are stored in the *server_root*`/msg-`*instance*`/imta/config/locale/C/LC_MESSAGES/` directory (specified by the `IMTA_LANG` setting in the */server_root*`/msg-`*instance*`/imta/config/imta_tailor` file). The filenames are as follows:

`return_bounced.txt`, `return_delivered.txt` `return_header.opt`, `return_timedout.txt`, `return_deferred.txt`, `return_failed.txt`, `return_prefix.txt`, `return_delayed.txt`, `return_forwarded.txt`, `return_suffix.txt`.

Note that you shouldn't change these files directly since they'll be overwritten when the iPlanet Messaging Server is upgraded. If you wish to modify these files and use them as your only set of notification message template files (`return_*.txt`), copy the files to a new directory and edit them there. Then, set the `IMTA_LANG` option in the `imta_tailor` file to point to the new directory containing those templates. If you wish to have multiple sets of notification files (for example, a set for each language) then you will need to set up a `NOTIFICATION_LANGUAGE` mapping table.

To Construct and Modify Notification Messages

A single notification message is constructed from a set of three files:
`return_prefix.txt + return_`*ActionStatus*`.txt + return_suffix.txt`

To customize or localize notifications, you would create a complete set of `return_*.txt` files for each locale and / or customization and store it in a separate directory. For example, you could have French notification files stored in one directory, Spanish for another, and notifications for a special unsolicited bulk email channel stored in a third.

NOTE Sample files for French, German, and Spanish are included in this release. These files can be modified to suit your specific needs.

The format and structure of notification messages sets is described below.

1. `return_prefix.txt` provides appropriate header text as well as introductory material for the body. The default for US-english locale is as follows:

   ```
   Content-type: text/plain; charset=us-asci
   Content-language: EN-US

   This report relates to a message you sent with the following
   header fields: %H
   ```

   Non-US-ASCII notification messages should change the `charset` parameter and `Content-Language` header value appropriately (for example, for French localized files the values would be `ISO-8859-1` and `fr`). `%H` is a header substitution sequence defined in Table 6-10.

2. `return_<`*ActionStatus*`>.txt` contains status-specific text. *ActionStatus* refers to a message's MTA status type. For example, the default text for `return_failed.txt` is as follows:

   ```
   Your message cannot be delivered to the following recipients:
   %R
   ```

   The default text for `return_bounced.txt` is:

   ```
   Your message is being returned.  It was forced to return by
   the postmaster.

   The recipient list for this message was:
   %R
   ```

3. `return_suffix.txt` contains concluding text. By default this file is blank.

Table 6-10 Notification Message Substitution Sequences

Substitutions	Definition
%H	Expands into the message's headers.
%C	Expands into the number of units[1] the message has been queued.
%L	Expands into the number of units[1] the message has left in the queue before it is returned.
%F	Expands into the number of units[1] a message is allowed to stay in the queue.
%S [%s]	Expands to the letter S or s if the previously expanded numeric value was not equal to one. Example: "%C day%s" can expand to "1 day" or "2 days" depending on how many days the message has been queued.

Table 6-10 Notification Message Substitution Sequences

Substitutions	Definition
%U [%u]	Expands into the time units[1] Hour [hour] or Day [day], in use. Example: "%C %U%s" can expand to "2 days" or "1 hour" depending on how many days or hours the message has been queued, and the value of the MTA option RETURN_UNITS. If you have set RETURN_UNITS=1 (hours) and your site is using localized notification messages, you will need to edit return_delayed.txt and return_timedout.txt and replace the word "days" with the word hours for all languages other than English. French, replace jour(s) with heure(s). German, replace Tag(e) with Stunde(n). Spanish, replace día(s) with hora(s)
%R	Expands into the list of the message's recipients.
%%	% (Note that the text is scanned byte by byte for substitutions sequences regardless of character set. Check for unintended % signs if you are using a double byte character set.)

1. Units is defined by the RETURN_UNITS option in the MTA Options file and can be hours or days (default).

To Customize and Localize Notification Messages

Notification messages can be localized such that messages will be returned to different users in different languages. For example, French notifications could be returned to users who have expressed a preference for French.

Localizing or customizing notification messages consists of two steps:

1. Create a set of localized / customized return_*.txt message files and store each set in a separate directory. This is described in "To Construct and Modify Notification Messages," on page 149.)
2. Set up a NOTIFICATION_LANGUAGE mapping table.

The NOTIFICATION_LANGUAGE mapping table (located in *server_root*/*msg*-instance/imta/config/mappings) specifies the set of localized or customized notification message files to use depending upon attributes (for example: language, country, domain, or address) of the *originating message* (the message causing the notification to be sent).

The original sender's message is parsed to determine status notification type, source channel, preferred language, return address and first recipient. Depending on how the table is constructed, a set of notification files is selected depending on one or more of these attributes.

The format of the NOTIFICATION_LANGUAGE mapping table is:

```
NOTIFICATION_LANGUAGE

 dsn-type-list|source-channel|preferred-language|return-address|first-recipient \
$Idirectory-spec
```

dsn-type-list is a comma-separated list of delivery status notification types. If multiple types are specified, they must be separated by commas and without spaces (a space will terminate the pattern of the mapping table entry). The types are as follows:

failed - Generic permanent failure messages (for example, no such user). Uses the return_failed.txt file.

bounced - Notification message used in conjunction with manual “bounces.” Done by the postmaster. Uses the return_bounced.txt file.

timedout - The MTA has been unable to deliver the message within the time allowed for delivery. The message is now being returned. Uses the return_timedout.txt file.

delayed - The MTA has been unable to deliver the message, but will continue to try to deliver it. Uses the return_delayed.txt file.

deferred - Non-delivery notification similar to “delayed” but without an indication of how much longer the MTA will continue to try to deliver. Uses the return_deferred.txt file.

forwarded - A delivery receipt was requested for this message, however, this message has now been forwarded to a system that does not support such receipts. Uses the return_forwarded.txt file.

source-channel is the channel generating the notification message, that is, the channel at which the message is currently queued. For example, ims-ms for the message store’s delivery queue, tcp_local for outbound SMTP queues, etc.

preferred-language is the language expressed in the message being processed (the message for which the notification is being generated). The sources for this information are first the accept_language field. If that doesn’t exist the Preferred-language: header field and X-Accept-Language: header field are used. For a list of standard language code values, refer to the file *server_root*/bin/msg/install/templates/msg-inst/msg/imta/config/languages.txt

This field, if not empty, will be whatever the originator of the message specified for the Preferred-language: or X-Accept-language: header line. As such, you may find nonsense characters in this field.

`return-address` is the envelope `From:` *address* of the originating message. This is the envelope address to which the notification message will be sent and hence a possible indicator of what language to use.

`first-recipient` is the envelope `To:` address (the first one, if the message is failing to more than one recipient) to which the original message was addressed. For example, in the notification "your message to dan@siroe.com could not be delivered"—in this case dan@siroe.com is the envelope `To:` address being reported on.

`directory-spec` is the directory containing the `return_*.txt` files to use if the mapping table probe matches. Note that a `$I` must precede the directory specification.

For instance, a site that stores French notification files (`return_*.txt`) in a directory `/lc_messages/table/notify_french/` and Spanish notification files in `return_*.txt` files in a directory `/lc_messages/table/notify_spanish/` might use a table as shown below. Note that each entry must start with one or more spaces, and that there can be no blank lines between entries.

```
NOTIFICATION_LANGUAGE

! Preferred-language: header value specified
!
   *|*|fr|*|*     $I/lc_messages/table/notify_french/
   *|*|es|*|*     $IIMTA_TABLE/notify_spanish/
   *|*|en|*|*     $I/imta/lang/
!
! If no Preferred-language value, then select notification based on the
! country code in the domain name. EX: PF=French Polynesia; BO=Bolivia
!
   *|*|*|*.fr|*   $I/imta/table/notify_french/
   *|*|*|*.fx|*   $I/imta/table/notify_french/
   *|*|*|*.pf|*   $I/imta/table/notify_french/
   *|*|*|*.tf|*   $I/imta/table/notify_french/
   *|*|*|*.ar|*   $I/imta/table/notify_spanish/
   *|*|*|*.bo|*   $I/imta/table/notify_spanish/
   *|*|*|*.cl|*   $I/imta/table/notify_spanish/
   *|*|*|*.co|*   $I/imta/table/notify_spanish/
   *|*|*|*.cr|*   $I/imta/table/notify_spanish/
   *|*|*|*.cu|*   $I/imta/table/notify_spanish/
   *|*|*|*.ec|*   $I/imta/table/notify_spanish/
   *|*|*|*.es|*   $I/imta/table/notify_spanish/
   *|*|*|*.gp|*   $I/imta/table/notify_spanish/
   *|*|*|*.gt|*   $I/imta/table/notify_spanish/
   *|*|*|*.gy|*   $I/imta/table/notify_spanish/
   *|*|*|*.mx|*   $I/imta/table/notify_spanish/
   *|*|*|*.ni|*   $I/imta/table/notify_spanish/
   *|*|*|*.pa|*   $I/imta/table/notify_spanish/
   *|*|*|*.ve|*   $I/imta/table/notify_spanish/
```

NOTE A default `mappings.locale` file is provided with the installation and will be included in the `mappings` file to enable notification language mapping. To disable notification language mapping, comment out the include line as follows:

```
! <IMTA_TABLE:mappings.locale
```

(Read the comments in the file and modify to suit your needs.)

Additional Notification Message Features

The essential procedures for setting up notification messages is describe in the previous sections. The following sections describe additional functionality.

To Block Content Return on Large Messages

Typically, when a message is bounced or blocked, the content of the message is returned to sender and to the local domain postmaster in the notification message. This can be a strain on resources if a number of very large messages are returned in their entirety. To block the return of content for messages over a certain size, set the `CONTENT_RETURN_BLOCK_LIMIT` option in the MTA options file.

To Remove non-US-ASCII Characters from Included Headers in the Notification Messages

The raw format for Internet message headers does not permit non-US-ASCII characters. If non-US-ASCII characters are used in a message header they are encoded using “MIME header encoding” described in RFC 2047. Thus, a Chinese “Subject” line in an email message will actually look like this:

```
Subject: =?big5?Q?=A4j=AB=AC=A8=B1=AD=B1=B0=D3=F5=A5X=AF=B2?=
```

and it is the responsibility of email clients to remove the encoding when displaying headers.

Because the `%H` template copies headers into the body of the notification message, the encoded header text will normally appear. However, Messaging Server will remove the encoding if the character set in the subject (in this case “`big5`”) matches the character set in the `Content-Type` header character set parameter in `return_prefix.txt`. If it doesn’t match, the Messaging Server will leave the encoding intact.

To Set Notification Message Delivery Intervals

Keywords: `notices`, `nonurgentnotices`, `normalnotices`, `urgentnotices`

Undeliverable messages are held in a given channel queue for specified amount of time before being returned to sender. In addition, a series of status / warning messages can be returned to the sender while Messaging Server attempts delivery. The amount of time and intervals between messages can be specified with the `notices`, `nonurgentnotices`, `normalnotices`, or `urgentnotices` keywords. Examples:

```
notices 1 2 3
```

Transient failure notification messages are sent after 1 and 2 days for all messages. If the message is still not delivered after 3 days, it is returned to its originator.

```
urgentnotices 2,4,6,8
```

Transient failure notifications are sent after 2, 4, and 6 days for messages of urgent priority. If the message is still not delivered after 8 days, it is returned to its originator.

Note that the `RETURN_UNITS` option in the MTA Options file allows you to specify the units in either hours (1) or days (0). The default is days (0).

If no `notices` keyword is specified, the default is to use the `notices` setting for the local, `l`, channel. If no setting has been made for the local channel, then the default is to use `notices 3, 6, 9, 12`.

To Include Altered Addresses in Notification Messages

Keywords: `includefinal`, `suppressfinal`, `useintermediate`

When the MTA generates a notification message (bounce message, delivery receipt message, and so on), there may be both an “original” form of a recipient address and an altered “final” form of that recipient address available to the MTA. The MTA always includes the original form (assuming it is present) in the notification message, because that is the form that the recipient of the notification message (the sender of the original message, which the notification message concerns) is most likely to recognize.

The `includefinal` and `suppressfinal` channel keywords control whether the MTA also includes the final form of the address. Suppressing the inclusion of the final form of the address may be of interest to sites that are “hiding” their internal mailbox names from external view. Such sites may prefer that only the original, “external” form of address be included in notification messages. `includefinal` is the default and includes the final form of the recipient address. `suppressfinal` causes the MTA to suppress the final address form, if an original address form is present, from notification messages.

The useintermediate keyword uses an intermediate form of the address produced after list expansion, but prior to user mailbox name generation. If this information isn't available, the final form is used.

To Send, Block and Specify Notification Messages to the Postmaster

By default, a copy of failure and warning notification messages are sent to the postmaster unless error returns and warnings are completely suppressed with a blank Errors-to: header line or a blank envelope From: address. Further granularity of notification message delivery to the postmaster can be controlled by a number of channel keywords described in the sections below and in Table 6-11.

Returned Failed Messages

Keywords: sendpost, nosendpost, copysendpost, errsendpost

A channel program may be unable to deliver a message because of long-term service failures or invalid addresses. When this occurs, the MTA channel program returns the message to the sender with an accompanying explanation of why the message was not delivered. Optionally, a copy of all failed messages is sent to the local postmaster. This is useful for monitoring message failures, but it can result in an excessive amount of traffic with which the postmaster must deal. (See Table 6-11.)

Warning Messages

Keywords:warnpost, nowarnpost, copywarnpost, errwarnpost

In addition to returning messages, the MTA can send detailed warnings for undelivered messages. This is generally due to time-outs 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 continue. In most cases they also contain the headers and the first few lines of the message in question.

Optionally, a copy of all warning messages can be sent to the local postmaster. This can be somewhat useful for monitoring the state of the various queues, although it does result in lots of traffic for the postmaster to deal with. The keywords warnpost, copywarnpost, errwarnpost, and nowarnpost are used to control the sending of warning messages to the postmaster. (See Table 6-11.)

Blank Envelope Return Addresses

Keywords: returnenvelope

The returnenvelope keyword takes a single integer value, which is interpreted as a set of bit flags. Bit 0 (value = 1) controls whether or not return notifications generated by the MTA are written with a blank envelope address or with the address of the local postmaster. Setting the bit forces the use of the local postmaster address; clearing the bit forces the use of a blank address.

NOTE The use of a blank address is mandated by RFC 1123. However, some systems do not properly handle blank envelope From: addresses and may require the use of this option.

Bit 1 (value = 2) controls whether or not the MTA replaces all blank envelope addresses with the address of the local postmaster. This is used to accommodate noncompliant systems that do not conform to RFC 821, RFC 822, or RFC 1123.

Bit 2 (value = 4) prohibits syntactically invalid return addresses.

Bit 3 (value = 8) same as mailfromdnsverify keyword.

Postmaster Returned Message Content

Keywords:postheadonly, postheadbody

When a channel program or the periodic message return job returns messages to both the postmaster and the original sender, the postmaster copy can either be the entire message or just the headers. 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. (See Table 6-11.)

Setting Per Channel Postmaster Addresses

Keywords: aliaspostmaster, returnaddress, noreturnaddress, returnpersonal, noreturnpersonal

By default, the Postmaster's return address that is used when the MTA constructs bounce or notification messages is postmaster@*local-host*, where *local-host* is the official local host name (the name on the local channel), and the Postmaster personal name is "MTA e-Mail Interconnect." Care should be taken in the selection of the Postmaster address—an illegal selection may cause rapid message looping and a great number of error messages.

The RETURN_ADDRESS and RETURN_PERSONAL options can be used to set an MTA system default for the Postmaster address and personal name. Or, if per channel controls are desired, the returnaddress and returnpersonal channel keywords may be used. returnaddress and returnpersonal each take a required argument

specifying the Postmaster address and Postmaster personal name, respectively. `noreturnaddress` and `noreturnpersonal` are the defaults and signify that the default values should be used. The defaults are established via the `RETURN_ADDRESS` and `RETURN_PERSONAL` options or the normal default values if such options are not set.

If the `aliaspostmaster` keyword is placed on a channel, then any messages addressed to the user name `postmaster` (lowercase, uppercase, or mixed case) at the official channel name is redirected to `postmaster`@local-host, where local-host is the official local host name (the name on the local channel). Note that Internet standards require that any domain in the DNS that accepts mail have a valid postmaster account that receives mail. So this keyword can be useful when it is desired to centralize postmaster responsibilities, rather than setting separate postmaster accounts for separate domains. That is, whereas `returnaddress` controls what return postmaster address is used when the MTA generates a notification message from the postmaster, `aliaspostmaster` affects what the MTA does with messages addressed to the postmaster.

Table 6-11 Notification Messages Sent to the Postmaster and Sender Keywords

Keyword	Description
Returned Message Content	**Specifies Addresses on Notifications**
`notices`	Specifies the time that may elapse before notices are sent and messages returned.
`nonurgentnotices`	Specifies the time that may elapse before notices are sent and messages returned for messages of non-urgent priority.
`normalnotices`	Specifies the time that may elapse before notices are sent and messages returned for messages of normal priority.
`urgentnotices`	Specify the time which may elapse before notices are sent and messages returned for messages of urgent priority.
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.

Table 6-11 Notification Messages Sent to the Postmaster and Sender Keywords

Keyword	Description
nosendpost	Disables sending a copy of all failed messages to the postmaster.
Warning Messages	**How to handle warning messages.**
warnpost	Enables sending a copy of warning messages to the postmaster. The default 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 does not guarantee message security as postmasters and system managers are able to read the contents of messages using `root` system privileges, if they choose.
postheadbody	Returns both the headers and the contents of the message.
Returned Message Content	**Specifies Addresses on Notifications**
includefinal	Include final form of address in delivery notifications (recipient address).
returnenvelope	Control use of blank envelope return addresses. The `returnenvelope` keyword takes a single integer value, which is interpreted as a set of bit flags. Bit 0 (value = 1) controls whether or not return notifications generated by the MTA are written with a blank envelope address or with the address of the local postmaster. Setting the bit forces the use of the local postmaster address; clearing the bit forces the use of a blank address. Bit 1 (value = 2) controls whether or not the MTA replaces all blank envelope addresses with the address of the local postmaster. This is used to accommodate noncompliant systems that do not conform to RFC 821, RFC 822, or RFC 1123. Bit 2 (value = 4) prohibits syntactically invalid return addresses. Bit 3 (value = 8) same as `mailfromdnsverify` keyword.
suppressfinal	Suppress the final address form from notification messages, if an original address form is present, from notification messages.
useintermediate	Uses an intermediate form of the address produced after list expansion, but prior to user mailbox name generation. If this information isn't available, the final form is used.

Table 6-11 Notification Messages Sent to the Postmaster and Sender Keywords

Keyword	Description
Returned Message Content	**Specifies Addresses on Notifications**
`aliaspostmaster`	Messages addressed to the user name postmaster at the official channel name is redirected to postmaster@local-host, where local-host is the local host name (the name on the local channel).
`returnaddress`	Specifies the return address for the local postmaster.
`noreturnaddress`	Use `RETURN_ADDRESS` option value as postmaster address name.
`returnpersonal`	Set the personal name for the local postmaster.
`noreturnpersonal`	Use `RETURN_PERSONAL` option value as postmaster personal name.

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:

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.com   $U@a-host
b.org   $U@b-host
c.edu   $U%c@b-daemon
d.com   $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 7-1 the patterns are `a.com`, `b.org`, `c.edu`, and `d.com`.

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 164.

template

Is one of the following:

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 7-1 , the $U substitution sequence is used. For more information, see “Template Substitutions and Rewrite Rule Control Sequences” on page 176.
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 176.

For more information about template syntax, see “Rewrite Rule Templates” on page 168.

Rewrite Rule Patterns and Tags

This section consists of the following subsections:

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
$*	Matches any address. This rule, if specified, is tried first regardless of its position in the file.
$%	Percent Hack Rule. Matches any host / domain specification of the form A%B.
$!	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 167.

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 168.) 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	Page	Usage
A%B	169	A becomes the new user / mailbox name, B becomes the new host / domain specification, rewrite again.
A@B	168	Treated as A%B@B.
A%B@C	168	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	169	Treated as A@B@C@C.
A@B@C@D	169	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."

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

Address	First Host Domain Specification	Comments
`@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.
`@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 `"."`.

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. Thie section consists of the following subsections:

- "Changing the Current Tag Value, $T" on page 188
- "Controlling Error Messages Associated with Rewriting ($?)" on page 189

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

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

Substitution Sequence	Substitutes
$$	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.
$(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**
$1M	Apply only if the channel is an internal reprocessing channel.
$1N	Apply only if the channel is not an internal reprocessing channel.
$1~	Perform any pending channel match checks. If the checks fail, successfully terminate processing of the current rewrite rule template.
$A	Apply if host is to the right of the at sign
$B	Apply only to header / body addresses

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

Substitution Sequence	Substitutes
$C *channel*	Fail if sending to *channel*
$E	Apply only to envelope addresses
$F	Apply only to forward-directed (e.g., To:) addresses
$M *channel*	Apply only if *channel* is rewriting the address
$N *channel*	Fail if *channel* is rewriting the address
$P	Apply if host is to the right of a percent sign
$Q *channel*	Apply if sending to *channel*
$R	Apply only to backwards-directed (e.g., From:) addresses
$S	Apply if host is from a source route
$T*newtag*	Set the rewrite rule tag to newtag
$V*host*	Fail if the host name is not defined in the LDAP directory (either in the DC tree or as a virtual domain). If the LDAP search times out, the remainder of the rewrite pattern from directly after the character following the host name is replaced with the MTA option string `DOMAIN_FAILURE`.
$X	Apply if host is to the left of an exclamation point
$Z*host*	Fail if the host name is defined in the LDAP directory (either in the DC tree or as a virtual domain). If the LDAP search times out, the remainder of the rewrite pattern from directly after the character following the host name is replaced with the MTA option string `DOMAIN_FAILURE`.
$?*errmsg*	If rewriting fails, return *errmsg* instead of the default error message. The error message must be in US ASCII.
$*number*?*errmsg*	If rewriting fails, return *errmsg* instead of the default error message, and set the SMTP extended error code to *a.b.c*: • *a* is *number* / 1000000 (the first digit) • *b* is (*number* / 1000) remainder 1000 (the value of the digits 2 through 4) • *c* is *number* remainder 1000 (the value of the last three digits. The following example sets the error code to 3.45.89: `$3045089?the snark is a boojum`

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 RFC2822 deprecates 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 perform a literal insertion of such a 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 187 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 169), 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.

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.
$!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.

Table 7-6 Single Field Substitutions

Control Sequence	Usage
$*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 use channel keyword definitions in the MTA configuration file `imta.cnf`. Please read Chapter 6, “About MTA Services and Configuration,” as well as “Channel Definitions” on page 103 and “The MTA Configuration File” on page 109 before reading this chapter. This chapter contains the following sections:

- Channel Keywords Listed Alphabetically
- Channel Keywords Categorized by Function
- Configuring Channel Defaults
- Configuring SMTP Channels
- Configuring Message Processing and Delivery
- Configuring Address Handling
- Configuring Header Handling
- Attachments and MIME Processing
- Size Limits on Messages, User Quotas and Privileges
- File Creation in the MTA Queue
- Specifying Mailbox Filter File Location
- Configuring Logging and Debugging
- Miscellaneous Keywords

NOTE If you make channel definition changes in `imta.cnf`, 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 Keywords Listed Alphabetically

The following table is an alphabetized list of keywords.

Table 8-1 Channel Keywords Alphabetized

Keyword	Page	Keyword	Page	Keyword	Page	Keyword	Page
733	245	822	245	addreturnpath	253	addrsperfile	267
aliaslocal	255	aliaspostmaster	157	allowetrn	219	allowswitchchannel	231
authrewrite	233	backoff	238	bangoverpercent	247	bangstyle	245
bidirectional	237	blocketrn	219	blocklimit	266	cacheeverything	227
cachefailures	227	cachesuccesses	227	channelfilter	271	charset7	222
charset8	222	charsetesc	222	checkehlo	218	commentinc	253
commentmap	253	commentomit	253	commentstrip	253	commenttotal	253
connectalias	249	connectcanonical	249	copysendpost	156	copywarnpost	156
daemon	232	datefour	260	datetwo	260	dayofweek	260
defaulthost	249	defaultmx	230	defaultnameservers	230	deferred	238
defragment	264	dequeue_removeroute	257	destinationfilter	271	disableetrn	219
domainetrn	219	domainvrfy	220	dropblank	251	ehlo	218
eightbit	222	eightnegotiate	222	eightstrict	222	errsendpost	156
errwarnpost	156	expandchannel	243	expandlimit	243	exproute	247
fileinto	271	filesperjob	240	filter	271	forwardcheckdelete	228
forwardchecknone	228	forwardchecktag	228	header_733	245	header_822	245

Table 8-1 Channel Keywords Alphabetized

Keyword	Page	Keyword	Page	Keyword	Page	Keyword	Page
header_uucp	245	headerlabelalign	262	headerlinelength	262	headerread	258
headertrim	258	holdexquota	267	holdlimit	243	identnone	228
identnonelimited	228	identnonenumeric	228	identnonesymbolic	228	identtcp	228
identtcplimited	228	identtcpsymbolic	228	ignoreencoding	263	immnonurgent	
improute	247	includefinal	155	indenttcpnumeric	228	inner	258
innertrim	258	interfaceaddress	227	interpretencoding	263	language	263
lastresort	231	linelength	265	linelimit	266	localvrfy	220
logging	269	loopcheck	270	mailfromdnsverify	221	master	237
master_debug	270	maxblocks	264	maxheaderaddrs	261	maxheaderchars	261
maxjobs	240	maxlines	264	maxprocchars	262	maysaslserver	232
maytls	234	maytlsclient	234	maytlsserver	234	missingrecipientpolicy	250
msexchange	233	multiple	267	mustsaslserver	232	musttls	234
musttlsclient	234	musttlsserver	234	mx	230	nameservers	230
noaddreturnpath	253	nobangoverpercent	247	noblocklimit	266	nocache	227
nochannelfilter	271	nodayofweek	260	nodefaulthost	249	nodeferred	238
nodefragment	264	nodestinationfilter	271	nodropblank	251	noehlo	218
noexproute	247	noexquota	267	nofileinto	271	nofilter	271
noheaderread	258	noheadertrim	258	noimproute	247	noinner	258
noinnertrim	258	nolinelimit	266	nologging	269	noloopcheck	270
nomailfromdnsverify	221	nomaster_debug	270	nomsexchange	233	nomx	230
nonrandomemx	230	nonurgentbackoff	238	nonurgentblocklimit	242	nonurgentnotices	154
noreceivedfor	253	noreceivedfrom	253	noremotehost	249	norestricted	252
noreturnaddress	157	noreturnpersonal	157	noreverse	252	normalbackoff	238
normalblocklimit	242	normalnotices	154	norules	257	nosasl	232
nosaslserver	232	nosaslswitchchannel	232	nosendetrn	219	nosendpost	156
noservice	244	noslave_debug	270	nosmtp	218	nosourcefilter	271
noswitchchannel	231	notices	154	notls	234	notlsclient	234
notlsserver	234	novrfy	220	nowarnpost	156	nox_env_to	260

Table 8-1 Channel Keywords Alphabetized

Keyword	Page	Keyword	Page	Keyword	Page	Keyword	Page
percentonly	247	percents	245	personalinc	254	personalmap	254
personalomit	254	personalstrip	254	pool	240	port	227
postheadbody	157	postheadonly	157	randommx	230	receivedfor	253
receivedfrom	253	remotehost	249	restricted	252	returnaddress	157
returnenvelope	156	returnpersonal	157	reverse	252	routelocal	249
rules	257	rules	257	saslswitchchannel	232	sendetrn	219
sendpost	156	sensitivitycompanyconfidential	262	sensitivitynormal	262	sensitivitypersonal	262
sensitivityprivate	262	service	244	sevenbit	222	silentetrn	219
single	267	single_sys	232	slave	237	slave_debug	270
smtp	218	smtp_cr	218	smtp_crlf	218	smtp_crorlf	218
smtp_lf	218	sourceblocklimit	266	sourcecommentinc	253	sourcecommentmap	253
sourcecommentomit	253	sourcecommentstrip	253	sourcecommenttotal	253	sourcefilter	271
sourcepersonalinc	254	sourcepersonalmap	254	sourcepersonalomit	254	sourcepersonalstrip	254
sourceroute	245	streaming	223	subaddressexact	256	subaddressrelaxed	256
subaddresswild	256	subdirs	269	submit	271	suppressfinal	155
switchchannel	231	threaddepth	243	tlsswitchchannel	234	unrestricted	252
urgentbackoff	238	urgentblocklimit	242	urgentnotices	154	useintermediate	155
user	271	uucp	245	viaaliasoptional	257	viaaliasrequired	257
vrfyallow	220	vrfydefault	220	vrfyhide	220	warnpost	156
x_env_to	260						

Channel Keywords Categorized by Function

The following table is a categorized list of keywords.

Table 8-2 Channel Keywords Categorized by Function (Default in **Bold**).

Keyword	Page	Definition
Address Handling		
733	245	Use % routing in the envelope; synonymous with percents.
822	245	Use source routes in the envelope; same as sourceroute.
addreturnpath	253	Add Return-path: header to messages enqueued to this channel.
aliaslocal	255	Look up rewritten addresses in the alias file and alias database.
authrewrite	233	Used on a source channel to have the MTA propagate authenticated originator information, if available, into the headers.
bangoverpercent	247	Group A!B%C as A!(B%C)
bangstyle	245	Use UUCP ! routing in the envelope; synonymous with uucp.
defaulthost	249	Specify a domain name to use to complete addresses
dequeue_removeroute	257	Removes source routes from envelope To: addresses.
exproute	247	Require explicit routing when addresses passed to remote systems.
holdlimit	243	Hold message when number of envelope recipient addresses exceeds this limit.
improute	247	Implicit routing for this channel's addresses
missingrecipientpolicy	250	Set policy for how to legalize (which header to add) messages that are lacking any recipient headers.
noaddreturnpath	253	Do not add Return-path: header when enqueuing message.
nobangoverpercent	247	Group A!B%C as (A!B)%C
nodefaulthost	249	Do not specify a domain name to use to complete addresses
noexproute	247	No explicit routing for this channel's addresses
noimproute	247	No implicit routing for this channel's addresses
noreceivedfrom	253	Construct Received: header lines without including the original envelope From: address.
noremotehost	249	Use local host's domain name as the default domain name to complete addresses
norestricted	252	Same as unsrestricted.
noreverse	252	Exempts addresses in messages from address reversal processing
norules	257	Do not enforce channel-specific rewrite rule checks for this channel.

Table 8-2 Channel Keywords Categorized by Function (Default in **Bold**).

Keyword	Page	Definition
percentonly	247	Ignores bang paths. Use % routing in the envelope.
percents	245	Use % routing in the envelope; synonymous with 733.
remotehost	249	Use remote host's name as the default domain name to complete addresses
restricted	252	The channel connects to mail systems that require encoding.
reverse	252	Checked addresses against address reversal database or REVERSE mapping
routelocal	249	Causes the MTA, when rewriting an address to the channel, to attempt to "short circuit" any explicit routing in the address.
rules	257	Enforce channel-specific rewrite rule checks for this channel.
sourceroute	245	Synonymous with 822.
subaddressexact	256	Perform no special subaddress handling during entry matching; the entire mailbox, including the subaddress, must match an entry in order for the alias to be considered to match.
subaddressrelaxed	256	After looking for an exact match and then a match of the form name+*, the MTA should make one additional check for a match on just the name portion.
subaddresswild	256	After looking for an exact match including the entire subaddress, the MTA should next look for an entry of the form name+*.
unrestricted	252	Tells the MTA not to perform RFC 1137 encoding and decoding.
uucp	245	Use UUCP! routing in the envelope; synonymous with bangstyle.
viaaliasoptional	257	Final recipient addresses that match the channel are not required to be produced by an alias.
viaaliasrequired	257	Final recipient address that matches the channel must be produced by an alias.
Attachments and MIME Processing		
defragment	264	Partial messages queued to the channel are placed in the defragmentation channel queue instead.
ignoreencoding	263	Ignore Encoding: header on incoming messages.
interpretencoding	263	Interpret Encoding: header on incoming messages, if the need arises.
nodefragment	264	Disables defragmention.
Character Sets and Eight Bit Data		

Table 8-2 Channel Keywords Categorized by Function (Default in **Bold**).

Keyword	Page	Definition
charset7	222	Default character set to associate with 7-bit text messages
charset8	222	Default character set to associate with 8-bit text messages
charsetesc	222	Default character set to associate with 7-bit text containing the escape character
eightbit	222	Channel supports eight-bit characters.
eightnegotiate	222	Channel should negotiate use of eight-bit transmission if possible.
eightstrict	222	Reject messages that contain unnegotiated eight-bit data.
sevenbit	222	Do not support 8-bit characters; 8-bit characters must be encoded.
File Creation in the MTA Queue Area		
addrsperfile	267	Limit on the maximum number of recipients that can be associated with a single message file in a channel queue
expandchannel	243	Specifies channel in which to perform deferred expansion due to application of expandlimit.
expandlimit	243	Processes an incoming message “off-line” when the number of addressees exceeds this limit.
multiple	267	No limit on the number of recipients in a message file, however the SMTP channel defaults to 99.
single	267	A separate copy of the message will be created for each destination address on the channel.
single_sys	267	Create a single message copy for each destination system used.
subdirs	269	Specifies the number of subdirectories across which to spread messages for the channel queues.
Headers		
authrewrite	233	Used on a source channel to have the MTA propagate authenticated originator information, if available, into the headers.
commentinc	253	Leave comments in message header lines intact.
commentmap	253	Runs comment strings in message header lines through the COMMENT_STRINGS mapping table.
commentomit	253	Remove comments from message header lines.
commentstrip	253	Remove problematic characters from comment fields in message header lines.

Table 8-2 Channel Keywords Categorized by Function (Default in **Bold**).

Keyword	Page	Definition
commenttotal	253	Strip comments (material in parentheses) from all header lines, except Received: header lines. Not recommended.
datefour	260	Expand all year fields to four digits.
datetwo	260	Remove the leading two digits from four-digit dates. Provides compatibility with mail systems that require two digit dates; it should never be used for any other purpose.
dayofweek	260	Retain day of the week information and adds this information to date and time headers if it is missing.
defaulthost	249	Specify a domain name to use to complete addresses
dropblank	251	Strip illegal blank headers from incoming messages.
header_733	245	Use % routing in the message header.
header_822	245	Use source routes in the message header.
headerlabelalign	262	Controls the alignment point for message headers enqueued on this channel; it takes an integer-valued argument.
headerlinelength	262	Controls the length of header lines enqueued on this channel.
headerread	258	Apply header trimming rules from an options file to the message headers upon message enqueue (use with caution) before the original message headers are processed.
headertrim	258	Applies header trimming rules from an options file to the message headers after the original message headers are processed.
header_uucp	245	Use ! routing in the header
inner	258	Parse messages and rewrite inner headers.
innertrim	258	Apply header trimming rules from an options file to inner message headers (use with caution).
language	263	Specifies the default language in headers.
maxheaderaddrs	261	Controls how many addresses can appear on a single line.
maxheaderchars	261	Controls how many characters can appear on a single line.
missingrecipientpolicy	250	Set policy for how to legalize (which header to add) messages that are lacking any recipient headers.
nodayofweek	260	Removes day of the week from date and time headers. Provides compatibility with mail systems that cannot process this information; it should never be used for any other purpose.
nodefaulthost	249	Do not specify a domain name to use to complete addresses

Table 8-2 Channel Keywords Categorized by Function (Default in **Bold**).

Keyword	Page	Definition
nodropblank	251	Do not strip illegal blank headers from incoming messages.
noheaderread	258	Do not apply header trimming rules from option file.
noheadertrim	258	Do not apply header trimming rules from options file.
noinner	258	Do not to rewrite inner message header lines.
noinnertrim	258	Do not apply header trimming to inner message headers.
noreceivedfor	253	Construct `Received:` header lines without including any envelope recipient information.
noreceivedfrom	253	Construct `Received:` header lines without including the original envelope From: address.
noremotehost	249	Use local host's domain name as the default domain name to complete addresses
noreverse	252	Exempts addresses in messages queued to the channel from address reversal processing
norules	257	Do not enforce channel-specific rewrite rule checks for this channel.
nox_env_to	260	Remove `X-Envelope-to` header lines.
personalinc	254	Leave personal name fields in message header lines intact.
personalmap	254	Run personal names through PERSONAL_NAMES mapping table.
personalomit	254	Remove personal name fields from message header lines.
personalstrip	254	Strip problem characters from personal name fields in header lines.
receivedfor	253	If a message is addressed to just one envelope recipient, to include that envelope `To:` address in the Received: header line it constructs.
receivedfrom	253	Include the original envelope `From:` address when constructing a `Received:` header line for an incoming message if the MTA has changed the envelope `From:` address.
remotehost	249	Use remote host's name as the default domain name to complete addresses
restricted	252	Channel connects to mail systems that require this encoding.
reverse	252	Check addresses against address reversal database or REVERSE mapping
rules	257	Enforce channel-specific rewrite rule checks for this channel.
sensitivitycompanyconfidential	262	`Companyconfidential` is the upper sensitivity limit of messages accepted.

Table 8-2 Channel Keywords Categorized by Function (Default in **Bold**).

Keyword	Page	Definition
sensitivitynormal	262	Normal is the upper sensitivity limit of messages accepted.
sensitivitypersonal	262	Personal is the upper sensitivity limit of messages accepted.
sensitivityprivate	262	Private is the upper sensitivity limit of messages accepted.
sourcecommentinc	253	Leave comments in incoming message header lines.
sourcecommentmap	253	Runs comment strings in header lines through source channels.
sourcecommentomit	253	Remove comments from incoming message header lines, for example, To:, From:, and Cc: headers.
sourcecommentstrip	253	Remove problematic characters from comment field in incoming header lines.
sourcecommenttotal	253	Strip comments (material in parentheses) in incoming messages.
sourcepersonalinc	254	Leave personal names in incoming message header lines intact.
sourcepersonalmap	254	Run personal names through source channels.
sourcepersonalomit	254	Remove personal name fields from incoming message header lines.
sourcepersonalstrip	254	Strip problematic characters from personal name fields in incoming message header lines.
unrestricted	252	Tells the MTA not to perform RFC 1137 encoding and decoding.
x_env_to	260	Enables generation of X-Envelope-to header lines.
Incoming channel matching and switching		
allowswitchchannel	231	Allows switching to this channel from a switchchannel channel
nosaslswitchchannel	232	No switching to this channel upon successful SASL authentication
noswitchchannel	231	No channel switching should be done to or from the channel.
switchchannel	231	Switches from the server channel to the channel associated with the originating host.
saslswitchchannel	232	Cause incoming connections to be switched to a specified channel upon a client's successful use of SASL.
tlsswitchchannel	234	Switches to another channel upon successful TLS negotiation.
Logging and debugging		
logging	269	Log message enqueues and dequeues into the log file and activates logging for a particular channel.
loopcheck	270	Places a string into the SMTP EHLO response banner in order for the MTA to check if it is communicating with itself.

Table 8-2 Channel Keywords Categorized by Function (Default in **Bold**).

Keyword	Page	Definition
master_debug	270	Create debugging output in the channel's master program output.
nologging	269	Do not log message enqueues and dequeues into the log file.
noloopcheck	270	No string into the SMTP EHLO response banner.
nomaster_debug	270	No debugging output in the channel's master program output.
noslave_debug	270	Do not generate slave debugging output.
slave_debug	270	Generate slave debug output.
Long Address Lists or Headers		
expandchannel	243	Specifies channel in which to perform deferred expansion due to application of expandlimit.
expandlimit	243	Processes an incoming message "off-line" when the number of addressees exceeds this limit.
holdlimit	243	Holds a message when the number of addresses exceeds this limit.
maxprocchars	262	Maximum length header that can be processed and rewritten.
Mailbox filters		
channelfilter	271	Location of channel filter file; same as destinationfilter.
destinationfilter	271	Location of channel filter file that applies to outgoing messages.
fileinto	271	Specify effect on address when a mailbox filter fileinto operation is applied.
filter	271	Specify the location of user filter files.
nochannelfilter	271	No channel filtering for outgoing messages. Also known as nodestinationfilter.
nodestinationfilter	271	Do not perform channel filtering for outgoing messages.
nofileinto	271	Mailbox filter fileinto operator has no effect.
nofilter	271	Do not perform user mailbox filtering.
nosourcefilter	271	Do not perform channel filtering for incoming messages.
sourcefilter	271	Specify the location of channel filter file for incoming messages.
Notification and Postmaster Messages (see page 149 for complete notification procedures)		
aliaspostmaster	157	Messages addressed to the user name postmaster at the official channel name are redirected to postmaster@local-host, where local-host is the local host name (the name on the local channel).

Table 8-2 Channel Keywords Categorized by Function (Default in **Bold**).

Keyword	Page	Definition
copysendpost	156	Sends a copy of the failure notice to the postmaster unless the originator address on the failing message is blank.
copywarnpost	156	Sends a copy of the warning message to the postmaster unless the originator address on the undelivered message is blank.
errsendpost	156	Sends a copy of the failure notice to the postmaster only when the notice cannot be returned to the originator.
errwarnpost	156	Sends a copy of the warning message to the postmaster when the notice cannot be returned to the originator.
includefinal	155	Include final form of recipient address in delivery notifications.
nonurgentnotices	154	Specifies the amount of time that may elapse before notices are sent and messages returned for messages of non-urgent priority.
noreturnaddress	157	Use RETURN_ADDRESS option value as postmaster address name.
noreturnpersonal	157	Use RETURN_PERSONAL option value as postmaster personal name.
normalnotices	154	Specifies the amount of time that may elapse before notices are sent and messages returned for messages of normal priority.
nosendpost	156	Disables sending a copy of all failed messages to the postmaster.
notices	154	Specifies the amount of time that may elapse before notices are sent and messages returned.
nowarnpost	156	Disables sending a copy of warning messages to the postmaster.
postheadbody	157	Returns both the headers and the contents of the message.
postheadonly	157	Returns only headers to the postmaster.
returnaddress	157	Specifies the return address for the local postmaster.
returnenvelope	156	Control use of blank envelope return addresses.
returnpersonal	157	Set the personal name for the local postmaster.
sendpost	156	Enables sending a copy of all failed messages to the postmaster.
suppressfinal	155	Suppress the final address form from notification messages, if an original address form is present, from notification messages.
urgentnotices	154	Specify the amount of time which may elapse before notices are sent and messages returned for messages of urgent priority.
useintermediate	155	Uses an intermediate form of the address produced after list expansion, but prior to user mailbox name generation.

Table 8-2 Channel Keywords Categorized by Function (Default in **Bold**).

Keyword	Page	Definition
warnpost	156	Enables sending a copy of warning messages to the postmaster.
Processing Control and Job Submission (See Table 8-7 on page 236 for greater functional granularity)		
backoff	238	Frequency of attempted redelivery of unsuccessfully delivered messages. Can be overridden by the keywords normalbackoff, nonurgentbackoff, urgentbackoff.
bidirectional	237	Channel served by a master and slave program.
deferred	238	Recognize and honor of the Deferred-delivery: header line.
expandchannel	243	Specifies channel in which to perform deferred expansion due to application of expandlimit.
expandlimit	243	Processes an incoming message “off-line” when the number of addressees exceeds this limit.
filesperjob	240	Number of queue entries to be processed by a single job.
immnonurgent		Starts delivery immediately after submission for urgent, normal, and non-urgent messages.
master	237	Channel served by a master program (master).
maxjobs	240	Maximum number of jobs that can be running concurrently for the channel.
nodeferred	238	Specifies that Deferred-delivery: header line not be honored.
nonurgentbackoff	238	The frequency for attempted redelivery of nonurgent messages.
nonurgentblocklimit	242	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.
normalbackoff	238	The frequency for attempted redelivery of normal messages.
normalblocklimit	242	Forces messages above this size to nonurgent priority.
noservice	244	Service conversions for messages coming into this channel must be enabled via CHARSET-CONVERSION.
pool	240	Specifies a pool for a channel. Must be followed by the pool name to which delivery jobs for the current channel should be pooled.
service	244	Unconditionally enables service conversions regardless of CHARSET-CONVERSION entry.
slave	237	Channel served by a master program (slave).
threaddepth	243	Number of messages triggering new thread with multithreaded SMTP client.

Table 8-2 Channel Keywords Categorized by Function (Default in **Bold**).

Keyword	Page	Definition
urgentbackoff	238	The frequency for attempted redelivery of urgent messages.
urgentblocklimit	242	Forces messages above this size to normal priority.
user	271	Used on pipe channels to indicate under what user name to run.
Sensitivity limits		
sensitivitycompanyconfidential	262	Upper sensitivity limit of messages accepted.
sensitivitynormal	262	Normal is the upper sensitivity limit of messages accepted.
sensitivitypersonal	262	Personal is the upper sensitivity limit of messages accepted.
sensitivityprivate	262	Private is the upper sensitivity limit of messages accepted.
Size Limits on Messages, and User Quotas and Privileges		
blocklimit	266	Maximum number of MTA blocks allowed per message.
holdexquota	267	Hold messages for users that are over quota.
holdlimit	243	Holds an incoming message when the number of addresses exceeds this limit.
linelength	265	Limits the maximum permissible message line length on a channel-by-channel basis.
linelimit	266	Maximum number of lines allowed per message.
maxblocks	264	Specifies the maximum number of blocks allowed in a message.
maxlines	264	Specifies the maximum number of lines allowed in a message.
noblocklimit	266	No limit for the number of MTA blocks allowed per message.
noexquota	267	Return to originator any messages to users who are over quota.
nolinelimit	266	No limit specified for the number of lines allowed per message.
nonurgentblocklimit	242	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	242	Forces messages above this size to nonurgent priority.
sourceblocklimit	266	Maximum number of MTA blocks allowed per incoming message.
urgentblocklimit	242	Forces messages above this size to normal priority.
SMTP Authentication, SASL and TLS (See 232 for greater functional granularity)		
authrewrite	233	Used on a source channel to have the MTA propagate authenticated originator information, if available, into the headers.

Table 8-2 Channel Keywords Categorized by Function (Default in **Bold**).

Keyword	Page	Definition
maysaslserver	232	Permit clients to attempt to use SASL authentication.
maytls	234	Causes the MTA to offer TLS to incoming connections and to attempt TLS upon outgoing connections.
maytlsclient	234	The MTA SMTP client will attempt TLS use when sending outgoing messages, if sending to an SMTP server that supports TLS.
maytlsserver	234	The MTA SMTP server will advertise support for the STARTTLS extension and will allow TLS use when receiving messages.
msexchange	233	Used on TCP/IP channels to tell the MTA that this is a channel that communicates with Microsoft Exchange gateways and clients.
mustsaslserver	232	SMTP server does not accept messages unless remote client successfully authenticates.
musttls	234	Insist upon TLS in both outgoing and incoming connections.
musttlsclient	234	The MTA SMTP client will insist on TLS use when sending outgoing messages (the MTA will issue the STARTTLS command and that command must succeed).
musttlsserver	234	The MTA SMTP server will advertise support for the STARTTLS extension and will insist upon TLS use when receiving incoming messages.
nomsexchange	233	Default.
nosasl	232	SASL authentication is not permitted or attempted.
nosaslserver	232	SASL authentication is not permitted.
notls	234	TLS will not be permitted or attempted.
notlsclient	234	TLS use will not be attempted by the MTA SMTP client on outgoing connections (the STARTTLS command will not be issued during outgoing connections).
notlsserver	234	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).
saslswitchchannel	232	Cause incoming connections to be switched to a specified channel upon a client's successful use of SASL.
tlsswitchchannel	234	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.
SMTP Commands and Protocol (See Table 8-4 on page 216 for greater functional granularity)		

Table 8-2 Channel Keywords Categorized by Function (Default in **Bold**).

Keyword	Page	Definition
allowetrn	219	Honors ETRN commands.
blocketrn	219	Blocks ETRN commands.
checkehlo	218	Checks the SMTP response banner to determine whether to use EHLO or HELO.
disableetrn	219	Disable support for the ETRN SMTP command.
domainetrn	219	Honors only those ETRN commands that specify a domain.
domainvrfy	220	Issues VRFY commands using a full address.
ehlo	218	Uses the SMTP EHLO command on initial connections.
eightbit	222	Channel supports eight-bit characters.
eightnegotiate	222	Channel should negotiate use of eight-bit transmission if possible.
eightstrict	222	Reject messages that contain unnegotiated eight-bit data.
localvrfy	220	Issues VRFY commands using a local address.
mailfromdnsverify	221	Verifies domain used on `MAIL FROM:` command exists in the DNS.
noehlo	218	Does not use the EHLO command.
nomailfromdnsverify	221	Does not verify that the domain used on the `MAIL FROM:` command exists in the DNS.
nosendetrn	219	Does not send ETRN commands.
nosmtp	218	Does not support the SMTP protocol. This is the default.
novrfy	220	Does not issue VRFY commands.
sendetrn	219	Sends ETRN commands.
sevenbit	222	Do not support 8-bit characters; 8-bit characters must be encoded.
silentetrn	219	Honors ETRN commands without echoing channel information.
smtp	218	Supports the SMTP protocol. The keyword `smtp` is mandatory for all SMTP channels. (This keyword is equivalent to `smtp_crorlf`.)
smtp_cr	218	Accepts lines terminated with a carriage return (CR) without a following line feed (LF).
smtp_crlf	218	Lines must be terminated with a carriage return (CR) line feed (LF) sequence.
smtp_crorlf	218	Lines may be terminated with any of a carriage return (CR), or a line feed (LF) sequence, or a full CRLF.
smtp_lf	218	Accepts lines terminated with linefeed (LF) without preceding CR.

Table 8-2 Channel Keywords Categorized by Function (Default in **Bold**).

Keyword	Page	Definition
streaming	223	Controls the degree of protocol streaming used in the protocol associated with a channel.
vrfyallow	220	Provides informative responses to VRFY commands.
vrfydefault	220	Provides default responses to VRFY command, according to channel's HIDE_VERIFY option setting.
vrfyhide	220	Provides obfuscatory responses to SMTP VRFY command.
TCP/IP Connection and DNS Lookup Support (See Table 8-5 on page 224 for greater functional granularity)		
cacheeverything	227	Caches all connection information.
cachefailures	227	Caches only connection failure information.
cachesuccesses	227	Caches only connection success information.
connectalias	249	Deliver to whatever host is listed in the recipient address.
connectcanonical	249	Connect to the host alias for the system to which the MTA would be connected.
daemon	232	Connects to a specific host system regardless of the envelope address.
defaultmx	230	Channel determines whether to do MX lookups from network.
defaultnameservers	230	Consults TCP/IP stack's choice of nameservers.
forwardcheckdelete	228	If reverse DNS lookup 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	228	Does not perform a forward lookup after a DNS reverse lookup.
forwardchecktag	228	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 *.
identnone	228	No perform IDENT lookups; performs IP to hostname translation; includes both hostname and IP address in `Received:` header.
identnonelimited	228	No 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	228	Does not perform IDENT lookups or IP to hostname translation.
identnonesymbolic	228	Does not perform IDENT lookups; does perform IP to hostname translation; includes only the hostname in `Received:` header.

Table 8-2 Channel Keywords Categorized by Function (Default in **Bold**).

Keyword	Page	Definition
identtcp	228	Performs IDENT lookups on incoming SMTP connections and IP to hostname translation; include both hostname and IP address in Received: header
identtcplimited	228	Performs IDENT lookups on incoming SMTP connections and IP to hostname translation, but do not use the hostname during channel switching. Include hostname and IP address in Received: header.
indenttcpnumeric	228	Performs IDENT lookups on incoming SMTP connections, but does not perform IP to hostname translation.
identtcpsymbolic	228	Performs IDENT lookups on incoming SMTP connections and IP to hostname translation; includes only hostname in Received: header.
interfaceaddress	227	Binds to the specified TCP/IP interface address.
lastresort	231	Specifies a last resort host.
mailfromdnsverify	221	Verifies that the domain used on the MAIL FROM: command exists in the DNS.
mx	230	TCP/IP network and software supports MX records lookup.
nameservers	230	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.
nocache	227	Does not cache any connection information.
nomailfromdnsverify	221	Does not verify that the domain used on the MAIL FROM: command exists in the DNS.
nomx	230	TCP/IP network does not support MX lookups.
nonrandomemx	230	Does MX lookups; does not randomize returned entries with equal precedence.
port	227	Specifies the default port number for SMTP connections. The standard port is 25.
randommx	230	Does MX lookups; randomizes returned entries with equal precedence.
single	232	Specifies that a separate copy of the message should be created for each destination address on the channel.
single_sys	232	Creates single copy of message for each destination system used.
threaddepth	243	Number of messages triggering new thread with multithreaded SMTP client.

Table 8-2 Channel Keywords Categorized by Function (Default in **Bold**).

Keyword	Page	Definition
Miscellaneous		
submit	271	Used to mark a channel as a submit-only channel.
user	271	Used on pipe channels to indicate under what user name to run.

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 SMTP Channels

Depending on the type of installation, Messaging Server provides several SMTP channels at installation time (see table below). These channels implement 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.

Table 8-3 SMTP Channels

Channel	Definition
`tcp_local`	Receives inbound messages from remote SMTP hosts. Depending on whether you use a smarthost/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`	Used as a switch channel for tcp_local; authenticated users switch to the tcp_auth channel to avoid relay-blocking restrictions.
`tcp_submit`	Accepts message submissions—usually from user agents—on the reserved submission port 587 (see RFC 2476).
`tcp_tas`	IA special channel used by sites doing Unified Messaging.

You can modify the definitions of these channels or create new channels by adding or removing channel keywords as described in this section. In addition, an option file may be used to control various characteristics of TCP/IP channels. Such an option file must be stored in the MTA configuration directory (*ServerRoot*`/msg-`*instance*`/imta/config`) and named *x_option*, where *x* is the name of the channel. Refer to the *iPlanet Messaging Server Reference Manual* for details.

This section is divided into the following subsections:

- Configuring SMTP Channel Options
- SMTP Command and Protocol Support
- TCP/IP Connection and DNS Lookup Support
- SMTP Authentication, SASL, and TLS

- Using Authenticated Addresses from SMTP AUTH in Header
- Using Authenticated Addresses from SMTP AUTH in Header
- Specifying Microsoft Exchange Gateway Channels
- Transport Layer Security

Configuring SMTP 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, /*ServerInstance*/imta/config/tcp_local_option.

The option file consists of one or more keywords and associated values. 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*.

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 8-4 summarizes the keywords described in this section.

Table 8-4 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.
`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.

Table 8-4 SMTP Command and Protocol Keywords

Channel Keyword(s)	Description
`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.
`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	**Specify degree of protocol streaming for channel to use**

Table 8-4 SMTP Command and Protocol Keywords

Channel Keyword(s)	Description
`streaming`	Controls the degree of protocol streaming used in the protocol associated with a channel.

Channel Protocol Selection and Line Terminators

Keywords: `smtp`, `nosmtp`, `smtp_crlf`, `smtp_cr`, `smtp_crorlf`, `smtp_lf`

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

Keywords: `ehlo`, `noehlo`, `checkehlo`

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

Keywords: allowetrn, blocketrn, disableetrn, domainetrn, silentetrn, sendetrn, nosendetrn, novrfy

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.

disableetrn disables support for the ETRN command entirely; ETRN is not advertised by the SMTP server as a supported command.

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

Keywords: domainvrfy, localvrfy, vrfyallow, vrfydefault, vrfyhide

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

Keywords: mailfromdnsverify, nomailfromdnsverify

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

Keywords: `charset7`, `charset8`, `charsetesc`, `sevenbit`, `eightbit`, `eightnegotiate`, `eightstrict`

Character Set Labeling

The MIME specification provides a mechanism to label the character set used in a plain text message. Specifically, a `charset=` parameter can be specified as part of the `Content-type:` header line. Various character set names are defined in MIME, including US-ASCII (the default), ISO-8859-1, ISO-8859-2, and many more that have been subsequently defined.

Some existing systems and user agents do not provide a mechanism for generating these character set labels; as a result, some plain text messages may not be properly labeled. 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.

Note that the `charset8` keyword also controls the MIME encoding of 8-bit characters in message headers (where 8-bit data is unconditionally illegal). The MTA normally MIME-encodes any (illegal) 8-bit data encountered in message headers, labeling it as the UNKNOWN charset if no `charset8` value has been specified.

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. It is usually appropriate to label MTA local channels as follows:

```
l ... charset7 US-ASCII charset8 ISO-8859-1 ...
hostname
```

If there is no `Content-type` header in the message, it is added. This keyword also adds the `MIME-version:` header line if it is missing.

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

Keywords: 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.

Under normal circumstances, the extent of streaming support available is negotiated using the SMTP pipelining extension. As such this keyword should never be used under normal circumstances.

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.

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
- Reverse DNS Lookups
- IDENT Lookups
- TCP / IP MX Record Support
- Nameserver Lookups
- Last Resort Host
- Alternate Channels for Incoming Mail (Switch Channels)
- Target Host Choice

Table 8-5 lists the TCP / IP connection and DNS lookup keywords described in this section.

Table 8-5 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.

Table 8-5 TCP/IP Connection and DNS Lookup Keywords

Channel Keyword(s)	Description
cachesuccesses	Caches only connection success information.
nocache	Does not cache any connection information.
Reverse DNS Lookups	**Specifies how to handle Reverse 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.
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.

Table 8-5 TCP / IP Connection and DNS Lookup Keywords

Channel Keyword(s)	Description
`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
`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.**

Table 8-5 TCP/IP Connection and DNS Lookup Keywords

Channel Keyword(s)	Description
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

Keywords: port, interfaceaddress

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

Keywords: cacheeverything, nocache, cachefailures, cachesuccesses

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.

Reverse DNS Lookups

Keywords: `forwardchecknone`, `forwardchecktag`, `forwardcheckdelete`

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

Keywords: `identnone`, `identnonelimited`, `identtnonnumeric`, `identnonesymbolic`, `identtcp`, `identtcpnumeric`, `identtcpsymbolic`, `identtcplimited`

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

Keywords: mx, nomx, defaultmx, randommx, nonrandommx

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

Keywords: nameservers, defaultnameservers

When name server lookups are being performed, the nameservers channel keyword may be used to specify a list of name servers to consult rather than consulting the TCP/IP stack's own choice of name servers. The nameservers keyword requires a space separated list of IP addresses for the name servers, 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 name servers.

To prevent name server lookups on UNIX, you can modify the nsswitch.conf file. On NT, modify the TCP/IP configuration.

Last Resort Host

Keywords: lastresort

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.

The keyword requires a single parameter specifying the name of the “system of last resort.” For example:

```
tcp_local single_sys smtp mx lastresort mailhub.siroe.com
TCP-DAEMON
```

Alternate Channels for Incoming Mail (Switch Channels)

Keywords: switchchannel, allowswitchchannel, noswitchchannel. See also saslswitchchannel on 232, and tlsswitchchannel on 234

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

Keywords: daemon, single, single_sys

The interpretation and usage of the daemon keyword depends upon the type of channel to which it is applied.

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, SASL, and TLS

Keywords: maysaslserver, mustsaslserver, nosasl, nosaslserver, saslswitchchannel, nosaslswitchchannel)

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 and or more information about SASL, SMTP authentication, and security is in Chapter 12, “Configuring Security and Access Control.”

The maysaslserver, mustsaslserver, nosasl, nosaslserver, itchchannel, and saslswitchchannel channel keywords are used to configure SASL (SMTP AUTH) use during the SMTP protocol by SMTP channels such as TCP/IP channels.

nosasl is the default and means that SASL authentication is not permitted or attempted. It subsumes nosaslserver, which means that SASL authentication is not permitted. Specifying maysaslserver causes the SMTP server to permit clients to attempt to use SASL authentication. Specifying mustsaslserver causes the SMTP server to insist that clients use SASL authentication; the SMTP server does not accept messages unless the remote client successfully authenticates.

Use saslswitchchannel to cause 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.

Using Authenticated Addresses from SMTP AUTH in Header

Keywords: authrewrite

The authrewrite channel keyword may be used on a source channel to have the MTA propagate authenticated originator information, if available, into the headers. Normally the SMTP AUTH information is used, though this may be overridden via the FROM_ACCESS mapping. The authrewrite keyword takes a required integer value, according to Table 8-6.

Table 8-6 authrewrite Integer Values

Value	Usage
1	Add a Sender: header, or a Resent-sender: header if a Resent-from: or Resent-sender: was already present containing the AUTH originator.
2	Add a Sender: header containing the AUTH originator.

Specifying Microsoft Exchange Gateway Channels

Keywords: msexchange, nomsexchange

The msexchange channel keyword may be used on TCP/IP channels to tell the MTA that this is a channel that communicates with Microsoft Exchange gateways and clients. When placed on an incoming TCP/IP channel which has SASL enabled (via a maysaslserver or mustsaslserver keyword), it causes the MTA's SMTP server to advertise AUTH using an "incorrect" format (based upon the original ESMTP AUTH specification, which was actually incompatible with correct ESMTP usage, rather than the newer, corrected AUTH specification). Some Microsoft Exchange clients, for instance, does not recognize the correct AUTH format and only recognizes the incorrect AUTH format.

The msexchange channel keyword also causes advertisement (and recognition) of broken TLS commands.

nomsexchange is the default.

Transport Layer Security

Keywords: maytls, maytlsclient, maytlsserver, musttls, musttlsclient, musttlsserver, notls, notlsclient, notlsserver, tlsswitchchannel

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.

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:

- “Setting Channel Directionality” on page 237
- “Implementing Deferred Delivery Dates” on page 238
- “Specifying the Retry Frequency for Messages that Failed Delivery” on page 238
- “Processing Pools for Channel Execution Jobs” on page 240
- “Service Job Limits” on page 240
- “Message Priority Based on Size” on page 242
- “SMTP Channel Threads” on page 243
- “Expansion of Multiple Addresses” on page 243
- “Enable Service Conversions” on page 244

For conceptual information on message processing and delivery, refer to “The Job Controller” on page 105 and “Job Controller File” on page 134.

Table 8-7 summarizes the keywords described in this section.

Table 8-7 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 redelivery of deferred messages. Can be overridden by `normalbackoff`, `nonurgentbackoff`, `urgentbackoff`.
`deferred`	Implements recognition and honoring of the `Deferred-delivery:` header line.
`nodeferred`	Default. Specifies that `Deferred-delivery:` header line not be honored.
`nonurgentbackoff`	The frequency for attempted redelivery of nonurgent messages.
`normalbackoff`	The frequency for attempted redelivery of normal messages.
`urgentbackoff`	The frequency for attempted redelivery 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.
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.

Table 8-7 Message Processing and Delivery Keywords

Keyword	Definition
SMTP Channel Threads	
`threaddepth`	Number of messages triggering new thread with multithreaded SMTP client.
Multiple Address Expansion	**Defines processing for messages with many recipients**
`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.

Setting Channel Directionality

Keywords: `master`, `slave`, `bidirectional`

Three keywords are used to specify whether a channel is served by a master program (`master`), a slave program (`slave`), or both (`bidirectional`). The default, if none of these keywords are specified, is `bidirectional`. These keywords determine whether the MTA initiates delivery activity when a message is queued to the channel.

The use of these keywords reflects certain fundamental characteristics of the corresponding channel program or programs. The descriptions of the various channels the MTA supports indicate when and where these keywords should be used.

Implementing Deferred Delivery Dates

Keywords: `deferred`, `nodeferred`

The `deferred` channel keyword implements recognition and honoring of the `Deferred-delivery:` header line. Messages with a `deferred` delivery date in the future are held in the channel queue until they either expire and are returned or the deferred delivery date is reached. See RFC 1327 for details on the format and operation of the `Deferred-delivery:` header line.

The keyword `nodeferred` is the default. It is important to realize that while support for deferred message processing is mandated by RFC 1327, actual implementation of it effectively lets people use the mail system as an extension of their disk quota.

Specifying the Retry Frequency for Messages that Failed Delivery

Keywords: `backoff`, `nonurgentbackoff`, `normalbackoff`, `urgentbackoff`, `notices`

By default, the frequency of delivery retries for messages that have had delivery failures depends on the message's priority. The default intervals between delivery attempts (in minutes) is shown below. The first number after the priority indicates the number of minutes after the initial delivery failure that the first delivery retry is attempted:

urgent: 30, 60, 60, 120, 120, 120, 240
normal: 60, 120, 120, 240, 240, 240, 480
nonurgent: 120, 240, 240, 480, 480, 480, 960

For urgent messages, a retry is attempted 30 minutes after the initial delivery failure, 60 minutes after the first delivery retry, 60 minutes after the second retry, 120 minutes after the third and so on. Retries after the last specified attempt repeat at the same interval. Thus, for urgent messages, retries occur every 240 minutes.

Delivery attempts continue for a period of time specified by the `notices`, `nonurgentnotices`, `normalnotices`, or `urgentnotices` keywords. If a successful delivery cannot be made, a *delivery failure notification* is generated and the message is returned to sender. (For details on the `notices` keyword, see "To Set Notification Message Delivery Intervals" on page 154.)

The backoff keywords allow you to specify customized sets of delivery retry intervals for messages of varying priorities. `nonurgentbackoff` specifies the intervals for nonurgent messages. `normalbackoff` specifies the intervals for normal messages. `urgentbackoff` specifies the intervals for urgent messages. If none of these keywords is specified, `backoff` specifies the intervals for all messages regardless of priority.

An example, is shown below:

```
urgentbackoff "pt30m" "pt1h" "pt2h" "pt3h" "pt4h" "pt5h" "pt8h"
"pt16h"
```

Here, delivery retries of urgent messages is attempted 30 minutes after the initial delivery failure, one hour after the first delivery attempt (1 hour 30 minutes after initial failure), two hours after the second delivery attempt, three hours after the third, four hours after the fourth, five hours after the fifth, eight hours after the sixth, 16 hours after the seventh delivery attempt. Subsequent attempts are made every 16 hours until the period of time specified by the `notices` keyword. If a successful delivery cannot be made, a delivery failure notification is generated and the message is returned to sender. Note that the interval syntax is in ISO 8601P and is described in the *iPlanet Messaging Server Reference Manual*.

In this next example,

```
normalbackoff "pt30m" "pt1h" "pt8h" "p1d" "p2d″ "p1w"
```

a delivery retry of normal messages is attempted 30 minutes after the initial delivery failure, one hour after the first delivery attempt, eight hours after the second attempt, one day after the third, two days after the fourth, one week after the fifth and repeating each week until the period of time specified by the `notices` keyword. If a successful delivery cannot be made, a delivery failure notification is generated and the message is returned to sender.

In this final example,

```
backoff "pt30m" "pt120m" "pt16h" "pt36h" "p3d"
```

all failed message deliveries, regardless of message priority—unless overridden by `nonurgentbackoff`, `normalbackoff`, or `urgentbackoff`—will be retried 30 minutes after the initial delivery failure, two hours after the first retry attempt, 16 hours after the second attempt, 36 hours after the third, three days after the fourth and repeating every three days until the period of time specified by the `notices` keyword. If a successful delivery cannot be made, a delivery failure notification is generated and the message is returned to sender.

Processing Pools for Channel Execution Jobs

Keywords: `pool`

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 242.)

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.

For further information on Job Controller concepts and configuration, refer to “Job Controller File,” on page 134, “The Job Controller” on page 105 and “Service Job Limits” on page 240.)

Service Job Limits

Keywords: `maxjobs`, `filesperjob`

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 further information on Job Controller concepts and configuration, refer to “Job Controller File,” on page 134, “Processing Pools for Channel Execution Jobs” on page 240 and “The Job Controller” on page 105.)

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 243.

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

Keywords: urgentblocklimit, normalblocklimit, nonurgentblocklimit

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

Keywords: `threaddepth`,

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

Keywords: `expandlimit`, `expandchannel`, `holdlimit`

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 time-outs 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.

Enable Service Conversions

Keywords: `service`, `noservice`

The `service` keyword unconditionally enables service conversions regardless of `CHARSET-CONVERSION` entry. If the `noservice` keyword is set, service conversions for messages coming into this channel must be enabled via `CHARSET-CONVERSION`.

Configuring Address Handling

This section describes keywords that deal with address handling. It consists of the following sections:

Address Types and Conventions

Keywords: 822, 733, uucp, header_822, header_733, header_uucp

This group of keywords control what types of addresses the channel supports. A distinction is made between the addresses used in the transport layer (the message envelope) and those used in message headers.

822 (sourceroute)

Source route envelope addresses. This channel supports full RFC 822 format envelope addressing conventions including source routes. The keyword `sourceroute` is also available as a synonym for 822. This is the default if no other envelope address type keyword is specified.

733 (percents)

Percent sign envelope addresses. This channel supports full RFC 822 format envelope addressing with the exception of source routes; source routes should be rewritten using percent sign conventions instead. The keyword `percents` is also available as a synonym for 733.

NOTE Use of 733 address conventions on an SMTP channel results in these conventions being carried over to the transport layer addresses in the SMTP envelope. This may violate RFC 821. Only use 733 address conventions when you are sure they are necessary.

uucp (bangstyle)

Bang-style envelope addresses. This channel uses addresses that conform to RFC 976 bang-style address conventions in the envelope (for example, this is a UUCP channel). The keyword `bangstyle` is also available as a synonym for uucp.

header_822

Source route header addresses. This channel supports full RFC 822 format header addressing conventions including source routes. This is the default if no other header address type keyword is specified.

header_733

Percent sign header addresses. This channel supports RFC 822 format header addressing with the exception of source routes; source routes should be rewritten using percent sign conventions instead.

NOTE Use of 733 address conventions in message headers may violate RFC 822 and RFC 976. Only use this keyword if you are sure that the channel connects to a system that cannot deal with source route addresses.

header_uucp

UUCP or bang-style header addresses. The use of this keyword is not recommended. Such usage violates RFC 976.

Interpreting Addresses that Use ! and %

Keywords: `bangoverpercent`, `nobangoverpercent`, `percentonly`

Addresses are always interpreted in accordance with RFC 822 and RFC 976. However, there are ambiguities in the treatment of certain composite addresses that are not addressed by these standards. In particular, an address of the form `A!B%C` can be interpreted as either:

- `A` as the routing host and `C` as the final destination host

or

- `C` as the routing host and `A` as the final destination host

While RFC 976 implies that mailers can interpret addresses using the latter set of conventions, it does not say that such an interpretation is required. Some situations may be better served by the former interpretation.

The `bangoverpercent` keyword forces the former `A!(B%C)` interpretation. The `nobangoverpercent` keyword forces the latter `(A!B)%C` interpretation. `nobangoverpercent` is the default.

NOTE This keyword does not affect the treatment of addresses of the form `A!B@C`. These addresses are always treated as `(A!B)@C`. Such treatment is mandated by both RFC 822 and RFC 976.

The `percentonly` keyword ignores bang paths. When this keyword is set, percents are interpreted for routing.

Adding Routing Information in Addresses

Keywords: `exproute`, `noexproute`, `improute`, `noimproute`

The addressing model that the MTA deals with assumes that all systems are aware of the addresses of all other systems and how to get to them. Unfortunately, this ideal is not possible in all cases, such as when a channel connects to one or more systems that are not known to the rest of the world (for example, internal machines

on a private TCP/IP network). Addresses for systems on this channel may not be legal on remote systems outside of the site. If you want to be able to reply to such addresses, they must contain a source route that tells remote systems to route messages through the local machine. The local machine can then (automatically) route the messages to these machines.

The `exproute` keyword (short for "explicit routing") tells the MTA that the associated channel requires explicit routing when its addresses are passed on to remote systems. If this keyword is specified on a channel, the MTA adds routing information containing the name of the local system (or the current alias for the local system) to all header addresses and all envelope `From:` addresses that match the channel. `noexproute`, the default, specifies that no routing information should be added.

The `EXPROUTE_FORWARD` option can be used to restrict the action of `exproute` to backward-pointing addresses. Another scenario occurs when the MTA connects to a system through a channel that cannot perform proper routing for itself. In this case, all addresses associated with other channels need to have routing indicated when they are used in mail sent to the channel that connects to the incapable system.

Implicit routing and the `improute` keyword is used to handle this situation. The MTA knows that all addresses matching other channels need routing when they are used in mail sent to a channel marked `improute`. The default, `noimproute`, specifies that no routing information should be added to addresses in messages going out on the specified channel. The `IMPROUTE_FORWARD` option can be used to restrict the action of `improute` to backward-pointing addresses.

The `exproute` and `improute` keywords should be used sparingly. They make addresses longer, more complex, and may defeat intelligent routing schemes used by other systems. Explicit and implicit routing should not be confused with specified routes. Specified routes are used to insert routing information from rewrite rules into addresses. This is activated by the special `A@B@C` rewrite rule template.

Specified routes, when activated, apply to all addresses, both in the header and the envelope. Specified routes are activated by particular rewrite rules and as such are usually independent of the channel currently in use. Explicit and implicit routing, on the other hand, are controlled on a per-channel basis and the route address inserted is always the local system.

Disabling Rewriting of Explicit Routing Addresses

Keywords: `routelocal`

The `routelocal` channel keyword causes the MTA, when rewriting an address to the channel, to attempt to “short circuit” any explicit routing in the address. Explicitly routed addresses (using !, %, or @ characters) are simplified.

Use of this keyword on “internal” channels, such as internal TCP / IP channels, can allow simpler configuration of SMTP relay blocking.

Note that this keyword should not be used on channels that may require explicit % or other routing.

Address Rewriting Upon Message Dequeue

Keywords: `connectalias`, `connectcanonical`

The MTA normally rewrites addresses as it enqueues messages to its channel queues. No additional rewriting is performed during message dequeue. This presents a potential problem when host names change while there are messages in the channel queues still addressed to the old name.

The `connectalias` keyword tells the MTA to deliver to whatever host is listed in the recipient address. This is the default. The keyword `connectcanonical` tells the MTA to connect to the host alias for the system that to which the MTA would be connected.

Specifying a Host Name to Use When Correcting Incomplete Addresses

Keywords: `remotehost`, `noremotehost`, `defaulthost`, `nodefaulthost`

The MTA often receives addresses that do not contain domain names from misconfigured or incompliant mailers and SMTP clients. The MTA attempts to make such addresses legal before allowing them to pass further. The MTA does this by appending a domain name to the address (for example, appends `@siroe.com` to `mrochek`).

For envelope `To:` addresses missing a domain name, the MTA always assumes that the local host name should be appended. However for other addresses, such as From: addresses, in the case of the MTA SMTP server there are at least two reasonable choices for the domain name: the local MTA host name and the remote host name reported by the client SMTP. Or in some cases, there may be yet a third reasonable choice—a particular domain name to add to messages coming in that channel. Now, either of these two first choices are likely to be correct as both may occur operationally with some frequency. The use of the remote host's domain name is appropriate when dealing with improperly configured SMTP clients. The use of the local host's domain name may be appropriate when dealing with a lightweight remote mail client such as a POP or IMAP client that uses SMTP to post messages. Or if lightweight remote mail clients such as POP or IMAP, clients should have their own specific domain name which is not that of the local host. Then add that specific other domain name may be appropriate. The best that the MTA can do is to allow the choice to be made on a channel by channel basis.

The `noremotehost` channel keyword specifies that the local host's name should be used. The keyword `noremotehost` is the default.

The `defaulthost` channel keyword is used to specify a particular host name to append to incoming bare user id's. It must be followed by the domain name to use in completing addresses (in envelope `From:` and in headers) that come into that channel. (In the case of submit channels, the `defaulthost` keyword's first argument also affects bare envelope `To:` addresses.) An optional second domain name (that has at least one period in it) may be specified to use in completing envelope `To:` addresses. `nodefaulthost` is the default.

The `switchchannel` keyword as described, in the preceding section, "Alternate Channels for Incoming Mail (Switch Channels)" can be used to associate incoming SMTP connections with a particular channel. This facility can be used to group remote mail clients on a channel where they can receive proper treatment. Alternatively, it is simpler to deploy standards-compliant remote mail clients (even if a multitude of noncompliant clients are in use) rather than attempting to fix the network-wide problem on your MTA hosts.

Legalizing Messages Without Recipient Header Lines

Keywords: `missingrecipientpolicy`

RFC 822 (Internet) messages are required to contain recipient header lines: `To:`, `Cc:`, or `Bcc:` header lines. A message without such header lines is illegal. Nevertheless, some broken user agents and mailers (for example, many older versions of sendmail) emit illegal messages.

The `missingrecipientpolicy` keyword takes an integer value specifying the approach to use for such messages; the default value, if the keyword is not explicitly present, is 0, meaning that envelope To: addresses are placed in a To: header.

Table 8-8 `missingrecipientpolicy` Values

Value	Action
0	Place envelope To: recipients in a `To:` header line.
1	Pass the illegal message through unchanged.
2	Place envelope To: recipients in a `To:` header line.
3	Place all envelope To: recipients in a single `Bcc:` header line.
4	Generate a group construct (for example, ;) `To:` header line, To: Recipients not specified.
5	Generate a blank `Bcc:` header line.
6	Reject the message.

Note that the MISSING_RECIPIENT_POLICY option can be used to set an MTA system default for this behavior. The initial Messaging Server configuration sets MISSING_RECIPIENT_POLICY to 1.

Stripping Illegal Blank Recipient Headers

Keywords: `dropblank`, `nodropblank`

In RFC 822 (Internet) messages, any `To:`, `Resent-To:`, `Cc:`, or `Resent-Cc:` header is required to contain at least one address—such a header may not have a blank value. Nevertheless, some mailers may emit such illegal headers. The `dropblank` channel keyword, if specified on a source channel, causes the MTA to strip any such illegal blank headers from incoming messages.

Enabling Channel-Specific Use of the Reverse Database

Keywords: `reverse`, `noreverse`

The `reverse` keyword tells the MTA that addresses in messages queued to the channel should be checked against, and possibly modified, by the address reversal database or `REVERSE` mapping, if either exists. `noreverse` exempts addresses in messages queued to the channel from address reversal processing. The `reverse` keyword is the default. Refer to “To Convert Addresses from an Internal Form to a Public Form” on page 144 for more information.

Enabling Restricted Mailbox Encoding

Keywords: `restricted`, `unrestricted`

Some mail systems have difficulty dealing with the full spectrum of addresses allowed by RFC 822. A particularly common example of this is sendmail-based mailers with incorrect configuration files. Quoted local-parts (or mailbox specifications) are a frequent source of trouble:

```
"smith, ned"@siroe.com
```

This is such a major source of difficulty that a methodology was laid out in RFC 1137 to work around the problem. The basic approach is to remove quoting from the address, then apply a translation that maps the characters requiring quoting into characters allowed in an atom (see RFC 822 for a definition of an atom as it is used here). For example, the preceding address would become:

```
smith#m#_ned@siroe.com
```

The `restricted` channel keyword tells the MTA that the channel connects to mail systems that require this encoding. The MTA then encodes quoted local-parts in both header and envelope addresses as messages are written to the channel. Incoming addresses on the channel are decoded automatically. The `unrestricted` keyword tells the MTA not to perform RFC 1137 encoding and decoding. The keyword `unrestricted` is the default.

NOTE The `restricted` keyword should be applied to the channel that connects to systems unable to accept quoted local-parts. It should not be applied to the channels that actually generate the quoted local-parts. (It is assumed that a channel capable of generating such an address is also capable of handling such an address.)

Generating of Return-path: Header Lines

Keywords: addreturnpath, noaddreturnpath

Normally, adding the Return-path: header line is the responsibility of a channel performing a final delivery. But for some channels, like the ims-ms channel, it is more efficient for the MTA to add the Return-path: header rather than allowing the channel to perform add it. The addreturnpath keyword causes the MTA to add a Return-path: header when enqueuing to this channel.

Constructing Received: Header Lines from Envelope To: and From: Addresses

Keywords: receivedfor, noreceivedfor, receivedfrom, noreceivedfrom

The receivedfor keyword instructs the MTA that if a message is addressed to just one envelope recipient, to include that envelope To: address in the Received: header line it constructs. The keyword receivedfor is the default. The noreceivedfor keyword instructs the MTA to construct Received: header lines without including any envelope addressee information.

The receivedfrom keyword instructs the MTA to include the original envelope From: address when constructing a Received: header line for an incoming message if the MTA has changed the envelope From: address due to, for example, certain sorts of mailing list expansions. receivedfrom is the default. The noreceivedfrom keyword instructs the MTA to construct Received: header lines without including the original envelope From: address.

Handling Comments in Address Header Lines

Keywords: commentinc, commentmap commentomit, commentstrip, commenttotal, sourcecommentinc, sourcecommentmap, sourcecommentomit, sourcecommentstrip, sourcecommenttotal

The MTA interprets the contents of header lines only when necessary. However, all registered header lines containing addresses must be parsed to rewrite and eliminate short form addresses and otherwise convert them to legal addresses. During this process, comments (strings enclosed in parentheses) are extracted and may be modified or excluded when the header line is rebuilt.

This behavior is controlled by the use of the commentinc, commentmap, commentomit, commentstrip, and commenttotalkeywords. The commentinc keyword tells the MTA to retain comments in header lines. It is the default. The keyword commentomit tells the MTA to remove any comments from addressing headers, for example, To:, From:, or Cc: header lines.

The keyword commenttotal tells the MTA to remove any comments from all header lines, except Received: header lines; this keyword is not normally useful or recommended. commentstrip tells the MTA to strip any nonatomic characters from all comment fields. The commentmap keyword runs comment strings through the COMMENT_STRINGS mapping table.

On source channels, this behavior is controlled by the use of the sourcecommentinc, sourcecommentmap, sourcecommentomit, sourcecommentstrip, and sourcecommenttotal keywords. The sourcecommentinc keyword indicates to the MTA to retain comments in header lines. It is the default. The sourcecommentomit keyword indicates to the MTA to remove any comments from addressing headers, for example, To:, From:, and Cc: headers. The sourcecommenttotal keyword indicates to the MTA to remove any comments from all headers, except Received: headers; as such, this keyword is not normally useful or recommended. And finally, the sourcecommentstrip keyword indicates to the MTA to strip any nonatomic characters from all comment fields. The sourcecommentmap keyword runs comment strings through source channels.

These keywords can be applied to any channel.

The syntax for the COMMENT_STRINGS mapping table is as follows:

(comment_text) | *address*

If the entry template sets the $Y flag, the original comment is replaced with the specified text (which should include the enclosing parentheses).

Handling Personal Names in Address Header Lines

Keywords: personalinc, personalmap, personalomit, personalstrip, sourcepersonalinc, sourcepersonalmap, sourcepersonalomit, sourcepersonalstrip

During the rewriting process, all header lines containing addresses must be parsed in order to rewrite and eliminate short form addresses and otherwise convert them to legal addresses. During this process personal names (strings preceding angle-bracket-delimited addresses) are extracted and can be optionally modified or excluded when the header line is rebuilt.

This behavior is controlled by the use of the `personalinc`, `personalmap`, `personalomit`, and `personalstrip` keywords. The keyword `personalinc` tells the MTA to retain personal names in the headers. It is the default. The keyword `personalomit` tells the MTA to remove all personal names. The keyword `personalstrip` tells the MTA to strip any nonatomic characters from all personal name fields. The `personalmap` keyword indicates to the MTA to run the personal names through the PERSONAL_NAMES mapping table.

On source channels, this behavior is controlled by the use of a `sourcepersonalinc`, `sourcepersonalmap`, `sourcepersonalomit`, or `sourcepersonalstrip` keyword. The `sourcepersonalinc` keyword indicates to the MTA to retain personal names in the headers. It is the default. The `sourcepersonalomit` keyword indicates to the MTA to remove all personal names. And finally, the `sourcepersonalstrip` indicates to the MTA to strip any nonatomic characters from all personal name fields. The `sourcepersonalmap` keyword indicates to the MTA to run the personal names through source channels.

These keywords can be applied to any channel.

The syntax of the PERSONAL_NAMES mapping table probes is:

personal_name | *address*

If the template sets the $Y flag, the original personal name is replaced with the specified text.

Specifying Alias File and Alias Database Probes

Keywords: `aliaslocal`

Normally only addresses rewritten to the local channel (that is, the l channel on UNIX) are looked up in the alias file and alias database. The `aliaslocal` keyword may be placed on a channel to cause addresses rewritten to that channel to be looked up in the alias file and alias database also. The exact form of the lookup probes that are made is then controlled by the `ALIAS_DOMAINS` option.

Subaddress Handling

Keywords: `subaddressexact`, `subaddressrelaxed`, `subaddresswild`

As background regarding the concept of subaddresses, the native and ims-ms channels interpret a + character in the local portion of an address (the mailbox portion) specially: in an address of the form *name+subaddress@domain* the MTA considers the portion of the mailbox after the plus character a subaddress. The native channel treats a subaddress as additional cosmetic information and actually deliver to the account name, without regard to the subaddress; the ims-ms channel interprets the subaddress as the folder name to which to deliver.

Subaddresses also affect the lookup of aliases by the local channel (that is, the L channel on UNIX) and the lookup of aliases by any channel marked with the `aliaslocal` keyword, and the lookup of mailboxes by the directory channel. The exact handling of subaddresses for such matching is configurable: when comparing an address against an entry, the MTA always first checks the entire mailbox including the subaddress for an exact match; whether or not the MTA performs additional checks after that is configurable.

The `subaddressexact` keyword instructs the MTA to perform no special subaddress handling during entry matching; the entire mailbox, including the subaddress, must match an entry in order for the alias to be considered to match. No additional comparisons (in particular, no wildcard comparisons or comparisons with the subaddress removed) are performed. The `subaddresswild` keyword instructs the MTA that after looking for an exact match including the entire subaddress, the MTA should next look for an entry of the form name+*. The `subaddressrelaxed` keyword instructs the MTA that after looking for an exact match and then a match of the form name+*, that the MTA should make one additional check for a match on just the name portion. With `subaddressrelaxed`, an alias entry of the following form matches either name or name+subaddress, transforming a plain name to newname, and transforming name+subaddress to newname+subaddress. The `subaddressrelaxed` keyword is the default.

```
name:   newname+*
```

Thus the `subaddresswild` keyword or the `subaddressrelaxed` keyword may be useful when aliases or a directory channel are in use yet users wish to receive mail addressed using arbitrary subaddresses. These keywords obviate the need for a separate entry for every single subaddress variant on an address.

Note that these keywords only make sense for the local channel (that is, the L channel on UNIX) and the directory channel, or any channel marked with the `aliaslocal` keyword.

Standard Messaging Server configurations relay upon the L channel indeed having `subaddressrelaxed` behavior (the default, when other keywords have not been explicitly used).

Enabling Channel-specific Rewrite Rules Checks

Keywords: `rules`, `norules`

The `rules` keyword tells the MTA to enforce channel-specific rewrite rule checks for this channel. This is the default. The `norules` keyword tells the MTA not to check for this channel. These two keywords are usually used for debugging and are rarely used in actual applications.

Removing Source Routes

Keywords: `dequeue_removeroute`

The `dequeue_removeroute` keyword removes source routes from envelope `To:` addresses as messages are dequeued. This keyword is currently only implemented on `tcp-*` channels. It is useful for transferring messages to systems that do not handle source routes correctly.

Specifying Address Must be from an Alias

Keywords: `viaaliasoptional`, `viaaliasrequired`

`viaaliasrequired` specifies that any final recipient address that matches the channel must be produced by an alias. A final recipient address refers to the match after alias expansion (if relevant) has been performed. The address cannot be handed directly to the MTA as a recipient address; that is, it is not sufficent for an address to merely rewrite to the channel. After rewriting to the channel, an address must also expand through an alias to be considered to have truly matched the channel.

The `viaaliasrequired` keyword may be used, for example, on the local channel to prevent delivery to arbitrary accounts (such as arbitrary native Berkeley mailboxes on a UNIX system).

The default is `viaaliasoptional`, which means that the final recipient addresses that match the channel are not required to be produced by an alias.

Configuring Header Handling

This section describes keywords that deal with header and envelope information. It consists of the following sections:

Rewriting Embedded Headers

Keywords: `noinner`, `inner`

The contents of header lines are interpreted only when necessary. However, MIME messages can contain multiple sets of message headers as a result of the ability to imbed messages within messages (message/RFC822). The MTA normally only interprets and rewrites the outermost set of message headers. The MTA can optionally be told to apply header rewriting to inner headers within the message as well.

This behavior is controlled by the use of the `noinner` and `inner` keywords. The keyword `noinner` tells the MTA not to rewrite inner message header lines. It is the default. The keyword `inner` tells the MTA to parse messages and rewrite inner headers. These keywords can be applied to any channel.

Removing Selected Message Header Lines

Keywords: `headertrim`, `noheadertrim`, `headerread`, `noheaderread`, `innertrim` `noinnertrim`

The MTA provides per-channel facilities for trimming or removing selected message header lines from messages. This is done through a combination of a channel keyword and an associated header option file or two. The `headertrim` keyword instructs the MTA to consult a header option file associated with the channel and to trim the headers on messages queued to that destination channel accordingly, *after the original message headers are processed*. The `noheadertrim` keyword bypasses header trimming. The keyword `noheadertrim` is the default.

The `innertrim` keyword instructs the MTA to perform header trimming on inner message parts, that is, embedded MESSAGE/RFC822 parts, as well. The `noinnertrim` keyword, which is the default, tells the MTA not to perform any header trimming on inner message parts.

The `headerread` keyword instructs the MTA to consult a header option file associated with the channel and to trim the headers on messages enqueued by that source channel accordingly, *before the original message headers are processed*. Note that `headertrim` header trimming, on the other hand, is applied after the messages have been processed and is the destination channel, rather than the source channel. The `noheaderread` keyword bypasses message enqueue header trimming. `noheaderread` is the default.

Unlike the `headeromit` and `headerbottom` keywords, the `headertrim` and `headerread` keywords may be applied to any channel whatsoever. Note, however, that stripping away vital header information from messages may cause improper operation of the MTA. Be extremely careful when selecting headers to remove or limit. This facility exists because there are occassional situations where selected header lines must be removed or otherwise limited.

CAUTION Stripping away header information from messages may cause improper operation of the MTA. Be careful when selecting headers to remove or limit. These keywords are provided for the rare situations where selected header lines must be removed or limited. Before trimming or removing any header line, you must understand the usage of that header line and have considered the possible implications of its removal.

Header options files for the `headertrim` and `innertrim` keywords have names of the form channel_`headers.opt` with channel, the name of the channel with which the header option file is associated. Similarly, header options files for the `headerread` keyword have names of the form `channel_read_headers.opt`. These files are stored in the MTA configuration directory, server_root`/msg-instance/imta/config/`.

Generating/Removing X-Envelope-to: Header Lines

Keywords: x_env_to, nox_env_to

The x_env_to and nox_env_to keywords control the generation or suppression of X-Envelope-to header lines on copies of messages queued to a specific channel. On channels that are marked with the single keyword, the x_env_to keyword enables generation of these headers while the nox_env_to removes such headers from enqueued messages. The default is nox_env_to.

The x_env_to keyword also requires the single keyword in order to take effect.

Converting Date to Two- or Four-Digits

Keywords: datefour, datetwo

The original RFC 822 specification called for two-digit years in the date fields in message headers. This was later changed to four digits by RFC 1123. However, some older mail systems cannot accommodate four-digit dates. In addition, some newer mail systems can no longer tolerate two-digit dates.

NOTE Systems that cannot handle both formats are in violation of the standards.

The datefour and datetwo keywords control the MTA's processing of the year field in message header dates. The keyword datefour, the default, instructs the MTA to expand all year fields to four digits. Two- digit dates with a value less than 50 have 2000 added, while values greater than 50 have 1900 added.

CAUTION The keyword datetwo instructs the MTA to remove the leading two digits from four-digit dates. This is intended to provide compatibility with incompliant mail systems that require two digit dates; it should never be used for any other purpose.

Specifying Day of Week in Date

Keywords: dayofweek, nodayofweek

The RFC 822 specification allows for a leading day of the week specification in the date fields in message headers. However, some systems cannot accommodate day of the week information. This makes some systems reluctant to include this information, even though it is quite useful information to have in the headers.

The `dayofweek` and `nodayofweek` keywords control the MTA's processing of day of the week information. The keyword `dayofweek`, the default, instructs the MTA to retain any day of the week information and to add this information to date and time headers if it is missing.

CAUTION The keyword `nodayofweek` instructs the MTA to remove any leading day of the week information from date and time headers. This is intended to provide compatibility with incompliant mail systems that cannot process this information properly; it should never be used for any other purpose.

Automatic Splitting of Long Header Lines

Keywords: `maxheaderaddrs`, `maxheaderchars`

Some message transfers, notably some sendmail implementations, cannot process long header lines properly. This often leads not just to damaged headers but to erroneous message rejection. Although this is a gross violation of standards, it is nevertheless a common problem.

The MTA provides per-channel facilities to split (break) long header lines into multiple, independent header lines. The `maxheaderaddrs` keyword controls how many addresses can appear on a single line. The `maxheaderchars` keyword controls how many characters can appear on a single line. Both keywords require a single integer parameter that specifies the associated limit. By default, no limit is imposed on the length of a header line nor on the number of addresses that can appear.

Header Alignment and Folding

Keywords: `headerlabelalign`, `headerlinelength`

The `headerlabelalign` keyword controls the alignment point for message headers enqueued on this channel; it takes an integer-valued argument. The alignment point is the margin where the contents of headers are aligned. For example, sample header lines with an alignment point of 10 might look like this:

```
To:       joe@siroe.com
From:     mary@siroe.com
Subject:  Alignment test
```

The default `headerlabelalign` is 0, which causes headers not to be aligned. The `headerlinelength` keyword controls the length of message header lines enqueued on this channel. Lines longer than this are folded in accordance with RFC 822 folding rules.

These keywords only control the format of the headers of the message in the message queue; the actual display of headers is normally controlled by the user agent. In addition, headers are routinely reformatted as they are transferred across the Internet, so these keywords may have no visible effect even when used in conjunction with simple user agents that do not reformat message headers.

Specifying Maximum Length Header

Keywords: `maxprocchars`

Processing of long header lines containing lots of addresses can consume significant system resources. The `maxprocchars` keyword is used to specify the maximum length header that the MTA can process and rewrite. Messages with headers longer than this are still accepted and delivered; the only difference is that the long header lines are not rewritten in any way. A single integer argument is required. The default is processing headers of any length.

Sensitivity Checking

Keywords: `sensitivitynormal`, `sensitivitypersonal`, `sensitivityprivate` `sensitivitycompanyconfidential`

The sensitivity checking keywords set an upper limit on the sensitivity of messages that can be accepted by a channel. The default is `sensitivitycompanyconfidential`; messages of any sensitivity are allowed through. A message with no `Sensitivity:` header is considered to be of normal, that is, the lowest, sensitivity. Messages with a higher sensitivity than that specified by such a keyword is rejected when enqueued to the channel with an error message:

```
message too sensitive for one or more paths used
```

Note that the MTA does this sort of sensitivity checking at a per-message, not per-recipient, level: if a destination channel for one recipient fails the sensitivity check, then the message bounces for all recipients, not just for those recipients associated with the sensitive channel.

Setting Default Language in Headers

Keywords: `language`

Encoded words in headers can have a specific language. The `language` keyword specifies the default language.

Attachments and MIME Processing

This section describes keywords that deal with attachments and MIME processing. It consists of the following sections:

Ignoring the Encoding: Header Line

Keywords: `ignoreencoding`, `interpretencoding`

The MTA can convert various nonstandard message formats to MIME using the Yes CHARSET-CONVERSION. In particular, the RFC 1154 format uses a nonstandard Encoding: header line. However, some gateways emit incorrect information on this header line, with the result that sometimes it is desirable to ignore this header line. The ignoreencoding keyword instructs the MTA to ignore any Encoding: header line.

NOTE Unless the MTA has a CHARSET-CONVERSION enabled, such headers are ignored in any case. The interpretencoding keyword instructs the MTA to pay attention to any Encoding: header line, if otherwise configured to do so, and is the default.

Automatic Defragmentation of Message/Partial Messages

Keywords: defragment, nodefragment

The MIME standard provides the message / partial content type for breaking up messages into smaller parts. This is useful when messages have to traverse networks with size limits, or traverse unreliable networks where message fragmentation can provide a form of “checkpointing,” allowing for less subsequent duplication of effort when network failures occur during message transfer. Information is included in each part so that the message can be automatically reassembled after it arrives at its destination.

The defragment channel keyword and the defragmentation channel provide the means to reassemble messages in the MTA. When a channel is marked defragment, any partial messages queued to the channel are placed in the defragmentation channel queue instead. After all the parts have arrived, the message is rebuilt and sent on its way. The nodefragment disables this special processing. The keyword nodefragment is the default.

Automatic Fragmentation of Large Messages

Keywords: maxblocks, maxlines

Some email systems or network transfers cannot handle messages that exceed certain size limits. The MTA provides facilities to impose such limits on a channel-by-channel basis. Messages larger than the set limits are automatically split (fragmented) into multiple, smaller messages. The content type used for such fragments is `message/partial`, and a unique ID parameter is added so that parts of the same message can be associated with one another and, possibly, be automatically reassembled by the receiving mailer.

The `maxblocks` and `maxlines` keywords are used to impose size limits beyond which automatic fragmentation are activated. Both of these keywords must be followed by a single integer value. The keyword `maxblocks` specifies the maximum number of blocks allowed in a message. An MTA block is normally 1024 bytes; this can be changed with the `BLOCK_SIZE` option in the MTA option file. The keyword `maxlines` specifies the maximum number of lines allowed in a message. These two limits can be imposed simultaneously if necessary.

Message headers are, to a certain extent, included in the size of a message. Because message headers cannot be split into multiple messages, and yet they themselves can exceed the specified size limits, a rather complex mechanism is used to account for message header sizes. This logic is controlled by the `MAX_HEADER_BLOCK_USE` and `MAX_HEADER_LINE_USE` options in the MTA option file.

`MAX_HEADER_BLOCK_USE` is used to specify a real number between 0 and 1. The default value is 0.5. A message's header is allowed to occupy this much of the total number of blocks a message can consume (specified by the `maxblocks` keyword). If the message header is larger, the MTA takes the product of `MAX_HEADER_BLOCK_USE` and `maxblocks` as the size of the header (the header size is taken to be the smaller of the actual header size and `maxblocks`) * `MAX_HEADER_BLOCK_USE`.

For example, if `maxblocks` is 10 and `MAX_HEADER_BLOCK_USE` is the default, 0.5, any message header larger than 5 blocks is treated as a 5-block header, and if the message is 5 or fewer blocks in size it is not fragmented. A value of 0 causes headers to be effectively ignored insofar as message-size limits are concerned.

A value of 1 allows headers to use up all of the size that's available. Each fragment always contains at least one message line, regardless of whether or not the limits are exceeded by this. `MAX_HEADER_LINE_USE` operates in a similar fashion in conjunction with the `maxlines` keyword.

Imposing Message Line Length Restrictions

Keywords: `linelength`

The SMTP specification allows for lines of text containing up to 1000 bytes. However, some transfers may impose more severe restrictions on line length. The `linelength` keyword provides a mechanism for limiting the maximum permissible message line length on a channel-by-channel basis. Messages queued to a given channel with lines longer than the limit specified for that channel are automatically encoded.

The various encodings available in the MTA always result in a reduction of line length to fewer than 80 characters. The original message may be recovered after such encoding is done by applying an appropriating decoding filter.

NOTE Encoding can only reduce line lengths to fewer than 80 characters. Specification of line length values less than 80 may not actually produce lines with lengths that comply with the stated restriction.

The `linelength` keyword causes encoding of data to perform “soft” line wrapping for transport purposes. The encoding is normally decoded at the receiving side so that the original “long” lines are recovered. For “hard” line wrapping, see the “Record, text” CHARSET-CONVERSION.

Size Limits on Messages, User Quotas and Privileges

This section describes keywords that set size limits on messages, user quotas, and privileges. It consists of the following sections:

Specifying Absolute Message Size Limits

Keywords: `blocklimit`, `noblocklimit`, `linelimit`, `nolinelimit`, `sourceblocklimit`

Although fragmentation can automatically break messages into smaller pieces, it is appropriate in some cases to reject messages larger than some administratively defined limit, (for example, to avoid service denial attacks).

The blocklimit, linelimit, and sourceblocklimit keywords are used to impose absolute size limits. Each of these keywords must be followed by a single integer value.

The keyword blocklimit specifies the maximum number of blocks allowed in a message. The MTA rejects attempts to queue messages containing more blocks than this to the channel. An MTA block is normally 1024 bytes; this can be changed with the BLOCK_SIZE option in the MTA option file.

The keyword sourceblocklimit specifies the maximum number of blocks allowed in an incoming message. The MTA rejects attempts to submit a message containing more blocks than this to the channel. In other words, blocklimit applies to destination channels; sourceblocklimit applies to source channels. An MTA block is normally 1024 bytes; this can be changed with the BLOCK_SIZE option in the MTA option file.

The keyword linelimit specifies the maximum number of lines allowed in a message. The MTA rejects attempts to queue messages containing more than this number of lines to the channel. The keywords, blocklimit and linelimit, can be imposed simultaneously, if necessary.

The MTA options LINE_LIMIT and BLOCK_LIMIT can be used to impose similar limits on all channels. These limits have the advantage that they apply across all channels. Therefore, the MTA servers can make them known to mail clients prior to obtaining message recipient information. This simplifies the process of message rejection in some protocols.

The nolinelimit and noblocklimit channel keywords are the default and mean that no limits are imposed, other than any global limits imposed via the LINE_LIMIT or BLOCK_LIMIT MTA options.

Handling Mail Delivery to Over Quota Users

Keywords: holdexquota, noexquota

The noexquota and holdexquota keywords control the handling of messages addressed to Berkeley mailbox users (UNIX), that is, users delivered to uid the native channel, who have exceeded their disk quotas.

noexquota tells the MTA to return messages addressed to over quota users to the message's sender. holdexquota tells the MTA to hold messages to over quota users; such messages remain in the MTA queue until they can either be delivered or they time out and are returned to their sender by the message return job.

File Creation in the MTA Queue

This section describes keywords that allow you to control disk resources by specifying file creation in the MTA queue. It consists of the following sections:

Controlling How Multiple Addresses on a Message are Handled

Keywords: `multiple`, `addrsperfile`, `single`, `single_sys`

The MTA allows multiple destination addresses to appear in each queued message. Some channel programs may only be able to process messages with one recipient, or with a limited number of recipients, or with a single destination system per message copy. For example, the SMTP channels master program establishes a connection only to a single remote host in a given transaction, so only addresses to that host can be processed (this, despite the fact, that a single channel is typically used for all SMTP traffic).

Another example is that some SMTP servers may impose a limit on the number of recipients they can handle at one time, and they may not be able to handle this type of error.

The keywords `multiple`, `addrsperfile`, `single`, and `single_sys` can be used to control how multiple addresses are handled. The keyword `single` means that a separate copy of the message should be created for each destination address on the channel. The keyword `single_sys` creates a single copy of the message for each destination system used. The keyword `multiple`, the default, creates a single copy of the message for the entire channel.

NOTE At least one copy of each message is created for each channel the message is queued to, regardless of the keywords used.

The `addrsperfile` keyword is used to put a limit on the maximum number of recipients that can be associated with a single message file in a channel queue, thus limiting the number of recipients that are processed in a single operation. This keyword requires a single-integer argument specifying the maximum number of

recipient addresses allowed in a message file; if this number is reached, the MTA automatically creates additional message files to accommodate them. (The default `multiple` keyword corresponds in general to imposing no limit on the number of recipients in a message file, however the SMTP channel defaults to 99.)

Spreading a Channel Message Queue Across Multiple Subdirectories

Keywords: `subdirs`

By default, all messages queued to a channel are stored as files in the directory `/imta/queue/`*channel-name*, where *channel-name* is the name of the channel. However, a channel that handles a large number of messages and tends to build up a large store of message files waiting for processing, for example, a TCP/IP channel, may get better performance out of the file system if those message files are spread across a number of subdirectories. The `subdirs` channel keyword provides this capability: it should be followed by an integer that specifies the number of subdirectories across which to spread messages for the channel, for example:

```
tcp_local single_sys smtp subdirs 10
```

Configuring Logging and Debugging

This section describe logging and debugging keywords.

Logging Keywords

Keywords: `logging`, `nologging`

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 13, “Logging and Log Analysis.”

Debugging Keywords

Keywords: `master_debug`, `slave_debug`, `nomaster_debug`, `noslave_debug`

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`.

On UNIX, when `master_debug` and `slave_debug` are enabled for the `l` channel, users then receive `imta_sendmail.log-`*uniqueid* files in their current directory (if they have write access to the directory; otherwise, the debug output goes to stdout.) containing MTA debug information.

Setting Loopcheck

Keywords: `loopcheck`, `noloopcheck`

The `loopcheck` keyword places a string into the SMTP EHLO response banner in order for the MTA to check if it is communicating with itself. When `loopcheck` is set, the SMTP server advertises an XLOOP extension.

When it communicates with an SMTP server supporting XLOOP, the MTA’s SMTP client compares the advertised string with the value of its MTA and immediately bounce the message if the client is in fact communicating with the SMTP server.

Miscellaneous Keywords

This section describes miscellaneous keywords. It consists of the following sections:

Channel Operation Type

Keywords: `submitsubmit`

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.

Pipe Channel

Keywords: `user`

The `user` keyword is used on pipe channels to indicate under what user name to run.

Note that the argument to `user` is normally forced to lowercase, but original case is preserved if the argument is quoted.

Specifying Mailbox Filter File Location

Keywords: `filter`, `nofilter`, `channelfilter`, `nochannelfilter`, `destinationfilter` `nodestinationfilter`, `sourcefilter`, `nosourcefilter`, `fileinto`, `nofileinto`)

The `filter` keyword may be used on the native and ims-ms channels to specify the location of user filter files for that channel. It takes a required URL argument describing the filter file location. `nofilter` is the default and means that a user mailbox filters are not enabled for the channel.

The `sourcefilter` and `destinationfilter` keywords may be used on general MTA channels to specify a channel-level filter to apply to incoming and outgoing messages, respectively. These keywords take a required URL argument describing the channel filter file location. `nosourcefilter` and `nodestinationfilter` are the defaults and mean that no channel mailbox filter is enabled for either direction of the channel.

The obsolete `channelfilter` and `nochannelfilter` keywords are synonyms for `destinationfilter` and `nodestinationfilter`, respectively.

The `fileinto` keyword, currently supported only for ims-ms channels, specifies how to alter an address when a mailbox filter `fileinto` operator is applied. For ims-ms channels, the usual usage is:

```
fileinto $U+$S@$D
```

The above specifies that the folder name should be inserted as a sub-address into the original address, replacing any originally present sub-address.

Chapter 9

Using Pre-defined Channels

When you first install iPlanet Messaging Server, several channels are already defined (see Table 9-1). This chapter describes how to use pre-defined channel definitions in the MTA.

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:

Table 9-1 Predefined Channels

Channel	Definition
`l`	**UNIX only.** Used to make routing decisions and for submitting 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 "To Deliver Messages to Programs Using the Pipe Channel" on page 275.

Table 9-1 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.

`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 smarthost / 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.

To Deliver Messages to Programs Using the Pipe Channel

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
```

To Configure the Native (/var/mail) Channel

An option file may be used to control various characteristics of the native channel. This native channel option file must be stored in the MTA configuration directory and named native_option (for example, *server_root*/msg-*instance*/imta/config/native_option).

Option files consist of several lines. Each line contains the setting for one option. An option setting has the form:

```
option=value
```

The *value* may be either a string or an integer, depending on the option's requirements.

Table 9-2 Local Channel Options

Options	Descriptions
FORCE_CONTENT_LENGTH (0 or 1; UNIX only)	If FORCE_CONTENT_LENGTH=1, then the MTA adds a Content-length: header line to messages delivered to the native channel, and causes the channel not to use the ">From" syntax when "From" is at the beginning of the line. This makes local UNIX mail compatible with Sun's newer mail tools, but potentially incompatible with other UNIX mail tools.

Table 9-2 Local Channel Options *(Continued)*

Options	Descriptions
FORWARD_FORMAT (string)	Specifies the location of the users' .forward files. The string %u indicates that it is substituted in each user id. The string %h indicates that it is substituted in each user's home directory. The default behavior, if this option is not explicitly specified, corresponds to: FORWARD_FORMAT=%h/.forward
REPEAT_COUNT (integer) SLEEP_TIME (integer)	In case the user's new mail file is locked by another process when the MTA tries to deliver the new mail, these options provide a way to control the number and frequency of retries the native channel program should attempt. If the file can not be opened after the number of retries specified, the messages remain in the native queue and the next run of the native channel attempts to deliver the new messages again. The REPEAT_COUNT option controls how many times the channel programs attempt to open the mail file before giving up. REPEAT_COUNT defaults to 30, (30 attempts). The SLEEP_TIME option controls how many seconds the channel program waits between attempts. SLEEP_TIME defaults to 2 (two seconds between retries).
SHELL_TIMEOUT (integer)	Controls the length of time in seconds the channel waits for a user's shell command in a .forward to complete. Upon such time-outs, the message are returned to the original sender with an error message resembling "Time-out waiting for *user*'s shell command *command* to complete." The default is 600 (10 minutes).
SHELL_TMPDIR (directory-specific)	Controls the location where the local channel creates its temporary files when delivering to a shell command. By default, such temporary files are created in users' home directories. Using this option, the administrator may instead choose the temporary files to be created in another (single) directory. For example: SHELL_TMPDIR=/tmp

To Temporarily Hold Messages Using the Hold Channel

The `hold` channel is used to hold the messages of a recipient temporarily prevented from receiving new messages. Messages may be held 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 by 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.

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.

The Conversion Channel

The `conversion` channel allows you to perform arbitrary body part-by-body part processing on specified messages flowing through the MTA. (Note that a body part is different than a message in that a message can contain multiple body parts as, for instance, in an attachment.) This processing can be done by any site-supplied programs or command procedures and may do such things such as convert text or images from one format to another, virus scanning, language translation and so forth. Various message types of the MTA traffic are selected for conversion, and specific processes and programs can be specified for each type of message body part.

The prerequisite for using this chapter is understanding the concept of channels (see "Channels" on page 100). For supplemental information on virus scanning using the `conversion` channel, refer to the iPlanet Messaging Server Technical Notes at the bottom of the iPlanet Messaging Server Documentation website.

Implementing the conversion channel consists of A) selecting message traffic for processing, and B) specifying how different messages will be processed. These procedures will discussed in further detail.

NOTE A default conversion channel is automatically created in the MTA configuration file (`imta.cnf`). This channel can be used as is and requires no modification.

MIME Overview

The conversion channel makes extensive use of the MIME (Multipurpose Internet Mail Extensions) header lines. Knowledge of message construction and MIME header fields is required. For complete information on MIME, refer to RFCs 1806, 2045 through 2049, and 2183. A short overview of MIME is presented here for convenience.

Message Construction

A simple message consists of a header and a body. The header is at the top of the message and contains certain control information such as date, subject, sender, and recipient. The body is everything after the first blank line after the header. MIME specifies a way to construct more complex messages which can contain multiple body parts, and even body parts nested within body parts. Messages like these are called multi-part messages, and, as mentioned earlier, the conversion channel performs body part-by-body part processing of messages.

MIME Headers

The MIME specification defines a set of header lines for body parts. These include `MIME-Version`, `Content-type`, `Content-Transfer-Encoding`, `Content-ID`, and `Content-disposition`. The conversion channel uses the `Content-type` and `Content-disposition` headers most frequently. An example of some MIME header lines is shown below:

```
Content-type: APPLICATION/wordperfect5.1;name=Poem.wpc
Content-transfer-encoding: BASE64
Content-disposition: attachment; filename=Poem.wpc
Content-description: "Project documentation Draft1 wordperfect format"
```

`Content-type` Header

The MIME `Content-Type` header describes the content of the body-part. The `Content-Type` header format (with an example) is shown below:

`Content-type:` *type*/*subtype*; *parameter1*=*value*; *parameter2*=*value*...

type describes the type of content of the body part. Examples of type are `Text`, `Multipart`, `Message`, `Application`, `Image`, `Audio`, and `Video`.

subtype further describes content type. Each `Content-type` has its own set of subtypes. For examples: `text/plain`, `application/octet-stream`, and `image/jpeg`. Content Subtypes for MIME mail are assigned and listed by the IANA (Internet Assigned Numbers Authority). A copy of the list is at http://www.isi.edu/in-notes/iana/assignments/media-types/media-types

parameter is specific to Content-type/subtype pairs. For example, the `charset` and the `name` parameters are shown below:

```
Content-type: text/plain; charset=us-ascii
Content-type: application/msword; name=temp.doc
```

The `charset` parameter specifies a character set for a textual message. The `name` parameter gives a suggested file name to be used if the data were to be written to a file.

NOTE `Content-Type` values, `subtypes`, and parameter names are case-insensitive.

Content-disposition Header

The MIME `Content-disposition` header provides presentation information for the body-part. It is often added to attachments specifying whether the attachment body part should be displayed (`inline`) or presented as a file name to be copied (`attachment`). The `Content-disposition` header has the following format:

`Content-disposition:` *disposition_type; parameter1=value;parameter2=value...*

disposition_type is usually `inline` (display the body part) or `attachment` (present as file to save.) `Attachment` usually has the parameter `filename` with a value specifying the suggested name for the saved file.

For details on the `Content-disposition` header, refer to RFC2183.

Selecting Traffic for Conversion Processing

Unlike other MTA channels, the conversion channel is not normally specified in an address or MTA rewrite rule. Instead, messages are sent to the conversion channel using the `CONVERSIONS` mapping table (specified by the parameter `IMTA_MAPPING_FILE` in the `imta_tailor` file). Entries to the table have the following format:

`IN-CHAN=`*source-channel*`;OUT-CHAN=`*destination-channel*`;CONVERT` *Yes/No*

As the MTA processes each message it probes the CONVERSIONS mapping table (if one is present). If the *source-channel* is the channel from which the message is coming and *destination-channel* is the channel to which the message is going, then the action following CONVERT is taken (Yes means the MTA diverts the message from its *destination-channel* to the conversion channel; if no match is found, the message will be queued to the regular destination channel).

NOTE An address of the form user@conversion.*localhostname* or user@conversion will be routed through the conversion channel, regardless of the CONVERSIONS mapping table.

The following example routes all non-internal messages—messages originating from, or destined to, the Internet—to the conversion channel.

```
CONVERSIONS

   IN-CHAN=tcp_local;OUT-CHAN=*;CONVERT   Yes
   IN-CHAN=*;OUT-CHAN=tcp_local;CONVERT     Yes
```

The first line specifies that messages coming from the tcp_local channel will be processed. The second line specifies that messages going to the tcp_local channel will also be processed. The tcp_local channel handles all messages going to and coming from the Internet. Since the default is to not go through the conversion channel, any other messages won't go through the conversion channel.

Note that this is a very basic table, and that it might not be sufficient for a site with a more customized configuration, for example, one using multiple outbound-to-the-Internet tcp_* channels, or using multiple inbound-from-the-Internet tcp_* channels.

To Control Conversion Processing

When a message is sent to the conversion channel, it is processed body part-by-body part. Processing is controlled by the MTA conversions file, which is specified by the IMTA_CONVERSION_FILE option in the imta_tailor file (default: *server_root*/msg-*instance*/imta/conversions). The conversions file consists of entries that control which types of body parts will be processed, and how they will be processed.

Each entry consists of one or more lines containing one or more name=*value* parameter clauses. The values in the parameter clauses conform to MIME conventions. 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 back slash (\) continuation character. Entries are terminated either by a line that does not end in a semicolon, one or more blank lines, or both.

Below is a simple example of a conversion file entry:

Code Example 9-1 conversion File Entry

```
out-chan=ims-ms; in-type=application; in-subtype=wordperfect5.1;
  out-type=application; out-subtype=msword; out-mode=block;
  command="/usr/bin/convert -in=wordp -out=msword 'INPUT_FILE' \
'OUTPUT_FILE'"
```

The clauses out-chan=ims-ms; in-type=application; in-subtype=wordperfect5.1 qualify the body part. That is, they specify the type of part to be converted. The header of each part is read and its Content-Type: and other header information is extracted. The entries in the conversion file are then scanned in order from first to last; any in-* parameters present, and the OUT-CHAN parameter, if present, are checked. If all of these parameters match the corresponding information for the body part being processed, then the conversion specified by the command= or delete= clause is performed, and the out-* parameters are set.

If no match occurs, then the part is matched against the next conversions file entry. Once all body parts have been scanned and processed (assuming there is a qualifying match), then the message is sent onwards to the next channel. If there are no matches, no processing occurs, and the message is sent to the next channel.

out-chan=ims-ms specifies that only message parts destined for the ims-ms channel will be converted. in-type=application and in-subtype=wordperfect5.1 specifies that the MIME Content-type header for the message part must be application/wordperfect5.1.

Message parts can be further qualified with additional `in-*` parameters. (See Table 9-5.) The entry above will trigger conversion actions on a message part which has the following MIME header lines:

```
Content-type: APPLICATION/wordperfect5.1;name=Draft1.wpc
Content-transfer-encoding: BASE64
Content-disposition: attachment; filename=Draft1.wpc
Content-description: "Project documentation Draft1 wordperfect format"
```

After the three `conversion` file qualifying parameters in Code Example 9-1, the next two parameters, `out-type=application` and `out-subtype=msword`, specify replacement MIME header lines to be attached to the “processed” body part. `out-type=application` and `out-subtype=msword` specify that the MIME `Content-type/subtype` of the outgoing message be `application/msword`.

Note that since the `in-type` and `out-type` parameters are the same, `out-type=application` is not necessary since the conversion channel defaults to the original MIME labels for outgoing body parts. Additional MIME labels for outgoing body parts can be specified with additional output parameters.

`out-mode=block` (Code Example 9-1) specifies the file type that the site-supplied program will return. In other words, it specifies how the file will be stored and how the conversion channel should be read back in the returned file. For example, an html file is stored in text mode, while an `.exe` program or a zip file is stored in block / binary mode. Mode is a way of describing that the file being read is in a certain storage format.

The final parameter in Code Example 9-1,

```
command="/usr/bin/convert -in=wordp -out=msword ‘INPUT_FILE’
‘OUTPUT_FILE’"
```

specifies the action to take on the body part. The `command=` parameter specifies that a program will execute on the body part. `/usr/bin/convert` is the hypothetical command name; `-in=wordp` and `-out=msword` are hypothetical command line arguments specifying the format of the input text and output text; `INPUT_FILE` and `OUTPUT_FILE` are conversion channel environmental parameters (see “To Use Conversion Channel Environmental Variables” on page 284) specifying a file containing the original body part and a file where the program should store its converted body part.

Instead of executing a command on the body part, the message part can simply be deleted by substituting `DELETE=1` in place of the `command` parameter.

Conversion Channel Information Flow

The flow of information is as follows: a message containing body parts comes into the conversion channel. The conversion channel parses the message, and processes the parts one by one. The conversion channel then qualifies the body part, that is, it determines if it should be processed or not by comparing its MIME header lines to the *qualifying parameters*. If the body part qualifies, the conversion processing commences. If MIME or body part information is to be passed to the conversion script, it is stored in an environmental variable (Table 9-3) as specified by *information passing parameter*s.

At this point, an action specified by an *action parameter*, is taken on the body part. Typically the action is that the body part be deleted or that it be passed to a program wrapped in a script. The script processes the body part and then sends it back to the conversion channel for reassembling into the post-processed message. The script can also send information to the conversion channel by using the conversion channel *output options*. This can be information such as new MIME header lines to add to the output body part, error text to be returned to the message sender, or special directives instructing the MTA to initiate some action such as bounce, delete, or hold a message.

Finally, the conversion channel replaces the header lines for the output body part as specified by the *output parameters*.

To Use Conversion Channel Environmental Variables

When operating on message body parts, it is often useful to pass MIME header line information, or entire body parts, to and from the site-supplied program. For example, a program may require `Content-type` and `Content-disposition` header line information as well as a message body part. Typically a site-supplied program's main input is a message body part which is read from a file. After processing the body part, the program will need to write it to a file from which the conversion channel can read it. This type of information passing is done by using conversion channel environmental variables.

Environmental variables can be created in the `conversions` file using the `parameter-symbol-*` parameter or by using a set of pre-defined conversion channel environmental variables (see Table 9-4 on page 288).

The following `conversions` file entry and incoming header show how to pass MIME information to the site-supplied program using environment variables.

conversions file entry:

```
in-channel=*; in-type=application; in-subtype=*;
  parameter-symbol-0=APPARENT_NAME; parameter-copy-0=*;
  dparameter-symbol-0=APPARENT_FILENAME; dparameter-copy-0=*;
  message-header-file=2; original-header-file=1;
  override-header-file=1; override-option-file=1;
  command="/bin/viro-scan500.sh ‘INPUT_FILE’ ‘OUTPUT_FILE’"
```

Incoming header:

```
Content-type: APPLICATION/msword; name=Draft1.doc
Content-transfer-encoding: BASE64
Content-disposition: attachment; filename=Draft1.doc
Content-description: "Project documentation Draft1 msword format"
```

`in-channel=*; in-type=application; in-subtype=*` specify that a message body part from any input channel of type `application` will be processed.

`parameter-symbol-0=APPARENT_NAME` specifies that the first `Content-type` parameter value (`Draft1.doc` in our example) be stored in an environment variable called `APPARENT_NAME`.

`parameter-copy-0=*` specifies that all `Content-type` parameters of the input body part be copied to the output body part.

`dparameter-symbol-0=APPARENT_FILENAME` specifies that the first `Content-disposition` parameter value (`Draft1.doc` in our example) be stored in an environment variable called `APPARENT_FILENAME`.

`dparameter-copy-0=*` specifies that all `Content-disposition` parameters of the input body part be copied to the output body part.

`message-header-file=2` specifies that the original header of the message as a whole (the outermost message header) be written to the file specified by the environment variable `MESSAGE_HEADERS`.

`original-header-file=1` specifies that the original header of the enclosing MESSAGE/RFC822 part are written to the file specified by the environment variable `INPUT_HEADERS`.

`override-header-file=1` specifies that MIME headers are read from the file specified by environmental variable `OUTPUT_HEADERS`, overriding the original header in the enclosing MIME part. `$OUTPUT_HEADERS` is an on-the-fly temporary file created at the time conversion runs. A site-supplied program would use this file to store headers changed during the conversion process. The conversion channel would then read the header lines from this file when it re-assembles the body part.

`override-option-file=1` specifies that the conversion channel read *conversion channel options* from the file named by the `OUTPUT_OPTIONS` environmental variable. See "To Use Conversion Channel Output Options" on page 287.

`command="SERVER_ROOT/msg-INSTANCE/bin/viro-scan500.sh"` specifies the command to execute on the message body part.

Table 9-3 Conversion Channel Environment Variables

Environment Variable	Description
`INPUT_ENCODING`	Encoding originally present on the body part.
`INPUT_FILE`	Name of the file containing the original body part. The site-supplied program should read this file.
`INPUT_HEADERS`	Name of the file containing the original header lines for the body part. The site-supplied program should read this file.
`INPUT_TYPE`	MIME `Content-type` of the input message part.
`INPUT_SUBTYPE`	MIME content subtype of the input message part.
`INPUT_DESCRIPTION`	MIME `content-description` of the input message part.
`INPUT_DISPOSITION`	MIME `content-disposition` of the input message part.
`MESSAGE_HEADERS`	Name of the file containing the original outermost header for an enclosing message (not just the body part) or the header for the part's most immediately enclosing MESSAGE/RFC822 part. The site-supplied program should read this file.
`OUTPUT_FILE`	Name of the file where the site-supplied program should store its output. The site-supplied program should create and write this file.
`OUTPUT_HEADERS`	Name of the file where the site-supplied program should store MIME header lines for an enclosing part. The site-supplied program should create and write this file. Note that file should contain actual header lines (not `option=value` lines) followed by a blank line as its final line.
`OUTPUT_OPTIONS`	Name of the file from which the site-supplied program should read conversion channel options. See "To Use Conversion Channel Output Options" on page 287."

To Use Conversion Channel Output Options

Conversion channel output options (Table 9-4) are dynamic variables used to pass information and special directives from the conversion script to the conversion channel. For example, during body part processing the script may want to send a special directive asking the conversion channel to bounce the message and to add some error text to the returned message stating that the message contained a virus.

The output options are initiated by setting `OVERRIDE-OPTION-FILE=1` in the desired conversion entry. Output options are then set by the script as needed and stored in the environmental variable file, `OUTPUT_OPTIONS`. When the script is finished processing the body part, the conversion channel reads the options from the `OUTPUT_OPTIONS` file.

The `OUTPUT_OPTION` variable is the name of the file from which the conversion channel reads options. Typically it is used as an on-the-fly temporary file to pass information. The example below shows a script that uses output options to return an error message to a sender who mailed a virus.

```
/usr/local/bin/viro_screen2k $INPUT_FILE   # run the virus screener

if [ $? -eq 1 ]; then
   echo "OUTPUT_DIAGNOSTIC='Virus found and deleted.'" > $OUTPUT_OPTIONS
   echo "STATUS=178029946" >> $OUTPUT_OPTIONS
else
   cp $INPUT_FILE $OUTPUT_FILE # Message part is OK
fi
```

In this example, the system diagnostic message and status code are added to the file defined by `$OUTPUT_OPTIONS`. If you read the `$OUTPUT_OPTIONS` temporary file out you would see something like:

```
OUTPUT_DIAGNOSTIC="Virus found and deleted."
STATUS=178029946
```

The line `OUTPUT_DIAGNOSTIC='Virus found and deleted'` tells the conversion channel to add the text `Virus found and deleted` to the message.

`178029946` is the `PMDF__FORCERETURN` status per the `pmdf_err.h` file which is found in the `server-root/bin/msg/imtasdk/include/pmdf_err.h`. This status code directs the conversion channel to bounce the message back to the sender. (For more information on using special directives refer to “To Bounce, Delete, or Hold Messages Using the Conversion Channel Output” on page 290.)

A complete list of the output options is shown below.

Table 9-4 Conversion Channel Output Options

Option	Description
OUTPUT_TYPE	MIME content type of the output message part.
OUTPUT_SUBTYPE	MIME content subtype of the output message part.
OUTPUT_DESCRIPTION	MIME content description of the output message part.
OUTPUT_DIAGNOSTIC	Text to include as part of the message sent to the sender if a message is forcibly bounced by the conversion channel.
OUTPUT_DISPOSITION	MIME `content-disposition` of the output message part.
OUTPUT_ENCODING	MIME content transfer `encoding` to use on the output message part.
OUTPUT_MODE	MIME `Mode` with which the conversion channel should write the output message part, hence the mode with which recipients should read the output message part.
STATUS	Exit status for the converter. This is typically a special directive initiating some action by the conversion channel. A complete list of directives can be viewed in `server-root/bin/msg/imtasdk/include/pmdf_err.h`

Headers in an Enclosing MESSAGE/RFC822 Part

When performing conversions on a message part, the conversion channel has access to the header in an enclosing `MESSAGE/RFC822` part, or to the message header if there is no enclosing `MESSAGE/RFC822` part. Information in the header may be useful for the site-supplied program.

If an entry is selected that has `ORIGINAL-HEADER-FILE=1`, then all the original header lines of the enclosing `MESSAGE/RFC822` part are written to the file represented by the `OUTPUT_HEADERS` environment variable. If `OVERRIDE-HEADER-FILE=1`, then the conversion channel will read and use as the header on that enclosing part the contents of the file represented by the `OUTPUT_HEADERS` environment variable.

To Call Out to a Mapping Table from a Conversion Entry

out-parameter-* values may be stored and retrieved in an arbitrarily named mapping table. This feature is useful for renaming attachments sent by clients that send all attachments with a generic name like att.dat regardless of whether they are postscript, msword, text or whatever. This is a generic way to relabel the part so that other clients (Outlook for example) will be able to open the part by reading the extension.

The syntax for retrieving a parameter value from a mapping table is as follows:

'mapping-table-name:mapping-input[*$Y*, *$N*]'

$Y returns a parameter value. If there is no match found or the match returns $N, then that parameter in the conversions file entry is ignored or treated as a blank string. Lack of a match or a $N does not cause the conversion entry itself to be aborted.

Consider the following mapping table:

```
X-ATT-NAMES

   postscript       temp.PS$Y
   wordperfect5.1   temp.WPC$Y
   msword           temp.DOC$Y
```

The following conversion entry for the above mapping table results in substituting generic file names in place of specific file names on attachments:

```
out-chan=tcp_local; in-type=application; in-subtype=*;
   in-parameter-name-0=name; in-parameter-value-0=*;
   out-type=application; out-subtype='INPUT-SUBTYPE';
   out-parameter-name-0=name;
   out-parameter-value-0="'X-ATT-NAMES:\\'INPUT_SUBTYPE\\''";
   command="cp  'INPUT_FILE'  'OUTPUT_FILE'"
```

In the example above, out-chan=tcp_local; in-type=application; in-subtype=* specifies that a message to be processed must come from the tcp_local channel with the content-type header of application/* (* specifies that any subtype would do).

`in-parameter-name-0=name; in-parameter-value-0=*` additionally specifies that the message must have as its first paramater type `name=*` (again, `*` specifies that any parameter value would do.)

`out-type=application;` specifies that the MIME `Content-type` parameter for the post-processing message be `application`.

`out-subtype='INPUT-SUBTYPE';` specifies that the MIME `subtype` parameter for the post-processing body part be the `INPUT-SUBTYPE` environmental variable, which is the original value of the input `subtype`. Thus, if you wanted change

```
Content-type: application/xxxx; name=foo.doc
```

to

```
Content-type: application/msword; name=foo.doc
```

then you would use

```
out-type=application; out-subtype=msword
```

`out-parameter-name-0=name;` specifies that the first MIME Content-type parameter of the output body part be of type `name=` .

`out-parameter-value-0='X-ATT-NAMES:\\'INPUT_SUBTYPE\\'';` says to take the first MIME `subtype` parameter value and search the mapping table `X-ATT-NAMES` for a `subtype` match. If a match is found, the `name` parameter receives the new value specified in the `X-ATT-NAMES` mapping table. Thus, if the parameter is of type `msword`, the `name` parameter will be `temp.DOC`.

To Bounce, Delete, or Hold Messages Using the Conversion Channel Output

This section describes how to use the conversion channel options to bounce, delete, or hold messages. The basic procedure is as follows:

1. Set `OVERRIDE-OPTION-FILE=1` in the appropriate conversions file entry. This tells the conversion channel to read the output options from the `OUTPUT_OPTIONS` file.
2. Use the conversion script to determine what action is required on a particular message body part.
3. In the script, specify the special directive for that action by writing the `STATUS=`*directive_code* option in the `OUTPUT_OPTIONS` file.

A complete listing of special directives can be found in `server_root/bin/msg/imtasdk/include/pmdf_err.h`. The ones commonly used by the conversion channel are:

NAME	Hex Value	Decimal Value
PMDF__FORCEHOLD	0x0A9C86AA	178030250
PMDF__FORCERETURN	0x0A9C857A	178029946
PMDF__FORCEDELETE	0x0A9C8662	178030178

We will explain the functions of these directives using examples.

To Bounce Messages

To bounce a message using the conversion channel set `OVERRIDE-OPTION-FILE=1` in the appropriate `conversions` file entry and add the following line to your conversion script:

```
echo "STATUS=178029946" >> $OUTPUT_OPTIONS
```

If you wish to add a short text string to the bounced message add the following line to the conversion script:

```
echo OUTPUT_DIAGNOSTIC=text-string >> $OUTPUT_OPTIONS
```

where text string is something like: “`The message sent from your machine contained a virus which has been removed. Be careful about executing email attachments.`”

To Conditionally Delete Message Parts

It may be useful to delete parts conditionally, depending on what they contain. This can be done using the output options. By contrast, the `DELETE=1` conversion parameter clause unconditionally deletes a message part.

To delete a message part using the output options, set `OVERRIDE-OPTION-FILE=1` in the appropriate conversions file entry and add the following line to your conversion script:

```
echo "STATUS=178030178" >> $OUTPUT_OPTIONS
```

To Hold a Message

It may be useful to hold messages conditionally, depending on what they contain. To delete a message part using the output options, set `OVERRIDE-OPTION-FILE=1` in the appropriate conversions file entry and add the following line to your conversion script:

```
echo "STATUS=178030250" >> $OUTPUT_OPTIONS
```

This requests that the conversion channel hold the message as a `.HELD` file in the conversion channel queue.

Conversion Channel Example

The `CONVERSIONS` mapping and set of conversion rules seen in examples below cause GIF, JPEG, and BITMAP files sent to the hypothetical channel `tcp_docuprint` to be converted into PostScript automatically. Several of these conversions use the hypothetical `/usr/bin/ps-converter.sh` to make that transformation. An additional rule that converts WordPerfect 5.1 files into Microsoft Word files is included.

```
CONVERSIONS

      IN-CHAN=*;OUT-CHAN=tcp_docuprint;CONVERT    Yes
```

```
!
out-chan=ims-ms; in-type=application; in-subtype=wordperfect5.1;
  out-type=application; out-subtype=msword; out-mode=block;
  command="/bin/doc-convert -in=wp -out=msw   'INPUT_FILE'  'OUTPUT_FILE'"

out-chan=tcp_docuprint; in-type=image; in-subtype=gif;
  out-type=application; out-subtype=postscript; out-mode=text;
  command="/bin/ps-convert -in=gif -out=ps   'INPUT_FILE'  'OUTPUT_FILE'"

out-chan=tcp_docuprint; in-type=image; in-subtype=jpeg;
  out-type=application; out-subtype=postscript; out-mode=text;
  command="/bin/ps-convert -in=jpeg -out=ps   'INPUT_FILE'  'OUTPUT_FILE'"

out-chan=tcp_docuprint; in-type=image; in-subtype=bitmap;
  out-type=application; out-subtype=postscript; out-mode=text;
  command="/bin/ps-convert -in=bmp -out=ps   'INPUT_FILE'  'OUTPUT_FILE'"
```

Table 9-5 Conversion Parameters

Parameter	Description
Qualifying Parameters (Specifies the parameters for which the message must match before it will be converted.)	
`OUT-CHAN`, `OUT-CHANNEL`	Output channel to match for conversion (wildcards allowed). The conversion specified by this entry is performed only if the message is destined for this specified channel.
`IN-CHAN`, `IN-CHANNEL`	Input channel to match for conversion (wildcards allowed). The conversion specified by this entry is only performed if the message is coming from the specified channel.
`IN-TYPE`	Input MIME type to match for conversion (wildcards allowed). The conversion specified is performed only if this field matches the MIME type of the body part.
`IN-SUBTYPE`	Input MIME subtype to match for conversion (wildcards allowed). The conversion specified by this entry is performed only if this field matches the MIME subtype of the body part.
`IN-PARAMETER-NAME-`*n*	Input MIME `Content-Type` parameter name that must match for conversion; *n* = 0, 1, 2.... This parameter can be used with `IN-PARAMETER-VALUE-`*n* to distinctly identify a parameter by its name and the value that it holds.

Table 9-5 Conversion Parameters *(Continued)*

Parameter	Description
IN-PARAMETER-VALUE-*n*	Input MIME Content-Type parameter value of corresponding IN-PARAMETER-NAME that must match for conversion. The conversion specified by this entry is performed only if this field matches the corresponding parameter in the body part's Content-Type parameter list. Wildcards allowed.
IN-PARAMETER-DEFAULT-*n*	Input MIME Content-Type parameter value default if parameter is not present. This value is used as a default for the IN-PARAMETER-VALUE-*n* test when no such parameter is specified in the body part.
IN-DISPOSITION	Input MIME Content-Disposition to match for conversion.
IN-DPARAMETER-NAME-*n*	Input MIME Content-Disposition parameter name that must match for conversion; *n* = 0, 1, 2.... This parameter can be used with IN-DPARAMETER-VALUE-*n* to distinctly identify a parameter by its name and the value that it holds.
IN-DPARAMETER-VALUE-*n*	Input MIME Content-Disposition parameter value of corresponding IN-DPARAMETER-NAME that must match for conversion. The conversion specified by this entry is performed only if this field matches the corresponding parameter in the body part's Content-Disposition: parameter list. Wildcards allowed.
IN-DPARAMETER-DEFAULT-*n*	Input MIME Content-Disposition parameter value default if parameter is not present. This value is used as a default for the IN-DPARAMETER-VALUE-*n* test when no such parameter is specified in the body part.
IN-DESCRIPTION	Input MIME Content-Description to match for conversion.
IN-SUBJECT	Input Subject from enclosing MESSAGE/RFC822 part.
Output Parameters (Specify the body part's post-conversion output settings.)	
OUT-TYPE	Output MIME type if it is different than the input type.
OUT-SUBTYPE	Output MIME subtype if it is different than the input subtype.
OUT-PARAMETER-NAME-*n*	Output MIME Content-Type parameter name; *n* = 0, 1, 2...
OUT-PARAMETER-VALUE-*n*	Output MIME Content-Type parameter value corresponding to OUT-PARAMETER-NAME-*n*.
PARAMETER-COPY-*n*	A list of the Content-Type parameters to copy from the input body part's Content-Type parameter list to the output body part's Content-Type: parameter list; *n*=0, 1, 2... Uses the same name of the MIME parameter to copy, as matched by an IN-PARAMETER-NAME-*n* clause.

Table 9-5 Conversion Parameters *(Continued)*

Parameter	Description
OUT-DISPOSITION	Output MIME Content-Disposition if it is different than the input MIME Content-Disposition.
OUT-DPARAMETER-NAME-*n*	Output MIME Content-Disposition parameter name; *n*=0, 1, 2...
OUT-DPARAMETER-VALUE-*n*	Output MIME Content-Disposition parameter value corresponding to OUT-DPARAMETER-NAME-*n*.
DPARAMETER-COPY-*n*	A list of the Content-Disposition: parameters to copy from the input body part's Content-Disposition: parameter list to the output body part's Content-Disposition: parameter list; *n* = 0, 1, 2,... Takes as argument the name of the MIME parameter to copy, as matched by an IN-PARAMETER-NAME-*n* clause. Wildcards may be used in the argument. In particular, an argument of * means to copy all the original Content-Disposition: parameters.
OUT-DESCRIPTION	Output MIME Content-Description if it is different than the input MIME Content-Description.
OUT-MODE	Mode in which to read and store the converted file. This should be BLOCK (binaries and executables) or TEXT.
OUT-ENCODING	Encoding to apply to the converted file when the message is reassembled.
Action Parameters (Specify an action to take on a message part.)	
COMMAND	Command to execute to perform conversion. Command to execute to perform conversion. This parameter is required; if no command is specified, the entry is ignored.
DELETE	0 or 1. If this flag is set, the message part is deleted. (If this is the only part in a message, then a single empty text part is substituted.)
RELABEL	RELABEL=1 will relabel the MIME label to whatever is specified by the Output parameters. Relabel=0 does nothing. Usually relabelling is done on mislabeled parts (example: from Content-type: application/octet-stream to Content-type: application/msword) so users can "doubleclick" to open a part, rather than having to save the part to a file and open it with a program.
SERVICE-COMMAND	SERVICE-COMMAND=command will execute a site-supplied procedure that will operate on entire MIME message (MIME headers and content body part). Also, unlike other CHARSET-CONVERSION operations or conversion channel operations, the service-command are expected to do their own MIME disassembly, decoding, re-encoding, and reassembly. Note that this flag causes an entry to be ignored during conversion channel processing; SERVICE-COMMAND entries are instead performed during character set conversion processing.

Table 9-5 Conversion Parameters *(Continued)*

Parameter	Description
TAG	Input tag, as set by a mail list CONVERSION_TAG parameter.
Information Passing Parameters (Used to pass information to and from the site-supplied program.)	
DPARAMETER-SYMBOL-*n*	Environment variable into which the Content-disposition parameter value, if present, will be stored; *n* = 0, 1, 2,... Each DPARAMETER-SYMBOL-*n* is extracted from the Content-Disposition: parameter list in order (n=0 is first parameter, n=2 second, etc.) and placed in the specified environment variable prior to executing the site-supplied program.
PARAMETER-SYMBOL-*n*	Environment variable into which the Content-Type parameter value, if present, will be stored; *n* = 0, 1, 2... Each PARAMETER-SYMBOL-*n* is extracted from the Content-Type: parameter list in order (n=0 is first parameter, n=2 second, etc.) and placed in an environment variable of the same name prior to executing the site-supplied program. Takes as argument the variable name into which the MIME parameter to convert, as matched by an IN-PARAMETER-NAME-*n* clause.
MESSAGE-HEADER-FILE	Writes all, part, or none of the original header of a message to the file specified by the environmental variable MESSAGE_HEADERS. If set to 1, the original header of the immediately enclosing body part are written to the file specified by the environmental variable MESSAGE_HEADERS. If set to 2, the original header of the message as a whole (the outermost message header) are written to the file.
ORIGINAL-HEADER-FILE	0 or 1. If set to 1, the original header of the enclosing MESSAGE/RFC822 part (not just the body part) are written to the file represented by the environmental variable OUTPUT_HEADERS.
OVERRIDE-HEADER-FILE	0 or 1. If set to 1, then MIME header lines are read by the conversion channel from the environmental variable OUTPUT_HEADERS, overriding the original header lines in the enclosing MIME part.
OVERRIDE-OPTION-FILE	If OVERRIDE-OPTION-FILE=1, the conversion channel reads options from the OUTPUT_OPTIONS environmental variable.
PART-NUMBER	Dotted integers: *a. b. c...* The part number of the MIME body part.

Character Set Conversion and Message Reformatting

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 9-6 lists the keywords.

Table 9-6 CHARSET-CONVERSION Mapping Table Keywords

Keyword	Description
`Always`	Force conversion even the message is going to be passed through the conversion channel before going to *out-channel*.
`Appledouble`	Convert other MacMIME formats to Appledouble format.
`Applesingle`	Convert other MacMIME formats to Applesingle format.
`BASE64`	Switch MIME encodings to BASE64.

Table 9-6 CHARSET-CONVERSION Mapping Table Keywords

Keyword	Description
`Binhex`	Convert other MacMIME formats, or parts including Macintosh type and Mac creator information, to Binhex format.
`Block`	Extract just the data fork from MacMIME format parts.
`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 10

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
- To Test Access Control Mappings
- To Add SMTP Relaying
- Configuring SMTP Relay Blocking
- Handling Large Numbers of Access Entries

- Access Control Mapping Table Flags

Controlling Access with Mapping Tables

You can control access to your mail services by configuring certain mapping tables. These mapping tables (Table 10-1) 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 file, see the *Messaging Server Reference Manual*.

Table 10-1 lists the mapping tables described in this section.

Table 10-1 Access Control Mapping Tables

Mapping Table	Description
SEND_ACCESS (See page 307.)	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.
ORIG_SEND_ACCESS (See page 307.)	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.
MAIL_ACCESS (See page 309.)	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.
ORIG_MAIL_ACCESS (See page 309.)	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.
FROM_ACCESS (See page 310.)	Used to filter mail based on envelope From addresses. Use this table if the To address is irrelevant.
PORT_ACCESS (See page 313.)	Used to block incoming connections based on IP number.

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 "Access Control Mapping Table Flags," on page 330.

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 10-1 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 10-1 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 10-2.

Figure 10-2 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 (that is, 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“Access Control Mapping Table Flags,” on page 330.

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 10-3. 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 10-3 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 10-2 lists the available flags.

Table 10-2 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

Table 10-2 PORT_ACCESS Mapping Flags

Flag	Description
$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.

To Limit 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 too frequently from particular IP addresses. 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 library 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.

To Test 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 10-4 shows a sample `SEND_ACCESS` mapping table and the resulting probe.

Figure 10-4 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:
```

To Add 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 `INTERNAL_IP` mapping table.

How the MTA Differentiates Between Internal and External Mail

In order to block mail relaying activities, the MTA 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, and is set by default.

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 MTA configuration described below is setup by default so that the server can differentiate between your internal and external message traffic.

- In the configuration file, immediately before the local channel, is a `defaults` channel with the `noswitchchannel` keyword:

  ```
  ! final rewrite rules
  defaults noswitchchannel
  ! Local store
  ims-ms ...
  ```

- The incoming TCP/IP channel specifies the `switchchannel` and `remotehost` keywords; for example:

```
tcp_local smtp single_sys mx switchchannel remotehost
TCP-DAEMON
```

- After the incoming TCP/IP channel definition is a similar channel with a different name; for example:

  ```
  tcp_intranet smtp single_sys mx allowswitchchannel routelocal
  tcp_intranet-daemon
  ```

  The `routelocal` channel keyword causes the MTA, when rewriting an address to the channel, to attempt to "short circuit" any explicit routing in the address through this channel, thereby blocking possible attempts to relay by means of looping through internal SMTP hosts via explicitly source routed addresses.

With the above configuration settings, SMTP mail generated within your domain will come in via the `tcp_intranet` channel. All other SMTP mail will come in via the `tcp_local` channel. Mail is distinguished between internal and external based upon which channel it comes in on.

How does this work? The key is the `switchchannel` keyword. 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 the source IP address matches an IP address or subnet in your `INTERNAL_IP` mapping table, the rewrite rule which calls out to that mapping table causes the address to rewrite to the `tcp_intranet` channel.

Since the `tcp_intranet` channel is marked with the `allowswitchchannel` keyword, the message is switched to the `tcp_intranet` channel and comes in on that channel. If the message comes in from a system whose IP address is not in the `INTERNAL_IP` mapping table, the reverse-pointing envelope rewrite will either rewrite to the `tcp_local` or, perhaps to some other channel. However, it will not rewrite to the `tcp_intranet` channel and since all other channels are marked `noswitchchannel` by default, 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_intranet`" 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.

To Use 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.so,\
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.so,\
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_verify.so,\
dns_verify_domain,$1,rbl.maps.vix.com.]$E
```

Support for DNS-based Databases

Starting with iPlanet Messaging Server 5.2 , the `dns_verify` program now supports DNS-based databases used to determine incoming SMTP connections that might send unsolicited bulk mail. Some of the publicly available DNS databases do not contain TXT records that are typically used for this purpose. Instead, they only contain `A` records.

In a typical setup, the `TXT` record found in the DNS for a particular IP address contains an error message suitable to return to the SMTP client when refusing a message. But, if a `TXT` record is not found and an `A` record is found, then versions of `dns_verify` prior to iPlanet Messaging Server 5.2 returned the message "*No error text available.*"

`dns_verify` now supports an option that specifies a default text that is used in the event that no `TXT` record is available. For example, the following `PORT_ACCESS` mapping table shows how to enable this option:

```
PORT_ACCESS

    *|*|*|*|* $C$|INTERNAL_IP;$3|$Y$E
    TCP|*|25|*|*
$C$[/export/home/iplanet/server51/msg/bin/imta/lib/dns_verify.so
,dns_verify_domain_port,$1,dnsblock.siroe.com,Your$ host$ ($1)$
found$ on$ dnsblock$ list]$E
    * $YEXTERNAL
```

In this example, if the remote system is found in a query in the domain `dnsblock.siroe.com`, but no `TXT` record is available, then the following message is returned, "*Your host a.b.c.d found on dnsblock list.*"

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 10-5.

Figure 10-5 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 10-6, which shows sample general database entries and a sample `ORIG_SEND_ACCESS` mapping table.

Figure 10-6 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.

Access Control Mapping Table Flags

Table 10-3 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 10-2).

Table 10-3 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+	
$J*address*	Replace original envelope From: address with specified *address*.*
$K*address*	Replace original Sender: address with specified *address*.* ++
$I*user* | *identifier*	Check specified user for group ID.
$<*string*	Send *string* to syslog (UNIX, user.notice facility and severity) or to the event log (NT) if probe matches.+++
$>*string*	Send *string* to syslog (UNIX, user.notice facility and severity) or to the event log (NT) if access is rejected. +++
$D*delay*	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).
$T*tag*	Prefix with *tag*.
$A*header*	Add the header line *header* to the message.
$X*error-code*	Issue the specified *error-code* extended SMTP error code if rejecting the message.
$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*.

Table 10-3 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
- To Create Per-User Filters
- To Create Channel-Level Filters
- To Create MTA-Wide Filters
- To Debug 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.

To Create 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 10-7 shows a sample template.

Figure 10-7 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 10-8. In the example, brackets indicate a drop-down list.

Figure 10-8 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.

To Create 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:///usr/tmp/filters/`*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 10-4 on page 337.

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 10-4 Substitution Tags (Case-insensitive)

Tag	Meaning
*	Perform group expansion. See “Processing Group Entries,” on page 556
**	Expand the attribute `mailForwardingAddress`. This can be a multivalued attribute resulting in several delivery addresses being produced.
$$	Substitute in the $ character
$\	Force subsequent text to lower case
$^	Force subsequent text to upper case
$_	Perform no case conversion on subsequent text
$~	Substitute in the file path for the home directory associated with the local part of the address
$1S	As `$S`, but if no subaddress is available just insert nothing
$2S	As `$S`, but if no subaddress is available insert nothing and delete the preceding character
$3S	As `$S`, but if no subaddress is available insert nothing and ignore the following character
$A	Substitute in the address, local-part@ host.domain
$D	Substitute in host.domain
$E	Insert the value of the second spare attribute, `LDAP_SPARE_1`
$F	Insert the name of the delivery file (`mailDeliveryFileURL` attribute)
$G	Insert the value of the second spare attribute, `LDAP_SPARE_2`
$H	Substitute in host
$I	Insert the hosted domain (part of UID to the right of the separator specified by `domainUidSeparator`). Fail if no hosted domain is available
$1I	As `$I`, but if no hosted domain is available just insert nothing
$2I	As `$I`, but if no hosted domain is available insert nothing and delete the preceding character
$3I	As `$I`, but if no hosted domain is available insert nothing and ignore the following character
$L	Substitute in local-part
$M	Insert the UID, stripped of any hosted domain
$P	Insert the program name (`mailProgramDeliveryInfo` attribute)

Table 10-4 Substitution Tags (Case-insensitive)

Tag	Meaning
$S	Insert the subaddress associated with the current address. The subaddress is that part of the user part of the original address after the subaddress separator, usually +, but can be specified by the MTA option SUBADDRESS_CHAR. Fail if no subaddress is given
$U	Insert the mailbox part of the current address. This is either the whole of the address to the left of the @ sign, or that part of the left hand side of the address before the subaddress separator, +.

To Create 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`.

To Debug 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:...
  ```

  This error may tell you exactly what is wrong with the filter.

- 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 11

Managing the Message Store

This chapter describes the message store and the message store administration interface. This chapter contains the following sections:

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. There are two ways to integrate this additional disk space into your system. The easiest way is to add additional partitions. Optionally, you can also add additional Messaging Server instances, each responsible for a particular message store. However, this approach is more complex.

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. You can also expand the message store by adding more partitions.

To manage the message store, iPlanet Messaging Server provides a set of command-line utilities in addition to the iPlanet Console interface. Table 11-1 describes these command-line utilities. For information about using these utilities, see “Performing Maintenance and Recovery Procedures” on page 362 and the *Messaging Server Reference Manual*.

Table 11-1 Message Store Command-line Utilities

Utility	Description
`configutil`	Sets and modifies configuration parameters for the store.
`deliver`	Delivers mail directly to the message store accessible by IMAP or POP mail clients.
`hashdir`	Identifies the directory that contains the message store for a particular user.
`iminitquota`	Reinitializes the quota limit from the LDAP directory and recalculates the disk space being used.
`imsasm`	Handles the saving and recovering of user mailboxes.
`imsbackup`	Backs up stored messages.
`imsexport`	Exports Certificate Management System mailboxes into UNIX `/var/mail` format folders.
`imsrestore`	Restores messages that have been backed up.
`imscripter`	The IMAP server protocol scripting tool. Executes a command or sequence of commands.
`mboxutil`	Lists, creates, deletes, renames, or moves mailboxes; reports quota usage.

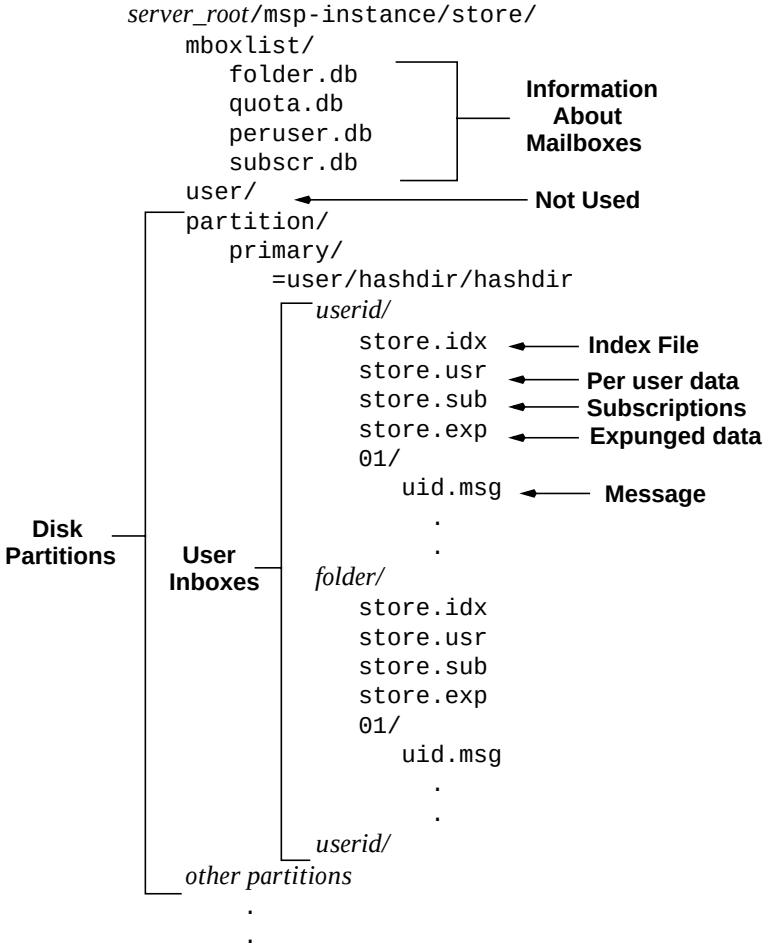
Table 11-1 Message Store Command-line Utilities

Utility	Description
`mkbackupdir`	Creates and synchronizes the backup directory with the information in the message store.
`MoveUser`	Moves a user's account from one messaging server to another.
`quotacheck`	Calculates the total mailbox size for each user in the message store and compares the size with their assigned quota.
`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.

Message Store Directory Layout

Figure 11-1 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 11-2.

Figure 11-1 Message Store Directory Layout



For example, a sample directory path might be:

server_root/msg-*instance*/store/partition/primary/=user/53/53/=mack1

Table 11-2 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/	Not used.
.../store/partition/	Contains the default primary partition. You can also place any other subpartitions you define in this directory.
/*partition*/=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.
/=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.

Table 11-2 Message Store Directory Description

Location	Content/Description
/*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	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	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 Messages

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.

Messages can also be erased by setting the **expire** option. The server deletes messages based on aging policies defined by `configutil`. At expiration, messages are expunged, but will not be physically removed until cleanup. (See "To Specify Aging Policies," on page 356.)

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:

- To Add an Administrator
- To Modify an Administrator Entry
- To Delete 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.

To Add 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 space-separated list of administrator IDs. If you specify more than one administrator, you must enclose the list in quotes.

To Modify 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"
```

To Delete 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 a user's messages exceed their quota, incoming messages remain in the MTA queue until one of the following occurs:

(1) The size or number of the user's messages no longer exceeds the quota, at which time the MTA delivers the messages to the user. (2) The undelivered message remains in the MTA queue longer than the specified grace period. See "To Set a Grace Period," on page 355.

Disk space becomes available when a user deletes and expunges messages or when the server deletes messages according to the aging policies you have established.

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.

Exceptions for Telephony Application Servers

To support unified messaging requirements, Messaging Server provides the ability to override quota limitations imposed by the message store. This guarantees 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:

- To Specify a Default User Quota
- To Enabling Quota Enforcement and Notification
- To Set 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.

To Specify 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 number of messages.

To 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 11-3.

Table 11-3 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. Define the quota warning messages

   See “Defining a Quota Warning Message,” on page 354.
4. 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.

To Set a Grace Period

The grace period specifies how long the mailbox can be over the quota (disk space or number of messages) before messages are bounced back to sender. Messages are accepted by the MTA, but remain in the MTA queue and are not delivered to the message store until one of the following occurs:

- The mailbox no longer exceeds the quota, at which time messages are delivered to the mailbox.
- The user has remained over quota longer than the specified grace period, at which time the server will bounce all messages including those in the queue.
- The message has remained in the queue longer than the maximum message queue time.

For example, if your grace period is set for two days, and you exceed quota for one day, new messages will continue to be received and held in the queue, and delivery attempts will continue. After the second day, messages bounce.

NOTE	Grace period is NOT how long the message will held in the queue, it's how long the mailbox is over quota before all incoming messages, including those in the queue, are bounced.

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.

To Specify 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. Note that this describes only the most frequently used `store.expire*` options. For a complete list refer to the *iPlanet Messaging Server Reference Manual*.

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.

To Specify Expiration Time and Day

To specify the expiration time and day:

`configutil -o store.expirestart -v` *`time`* (example: 23 is 11:00PM)
`configutil -o local.store.expire.workday -v` *`day`* (0-6, 0 is Sunday)

Setting `local.store.expire.workday` to -1 or a value larger than 6 will disable expire/cleanup. `stored` will check this configuration variable at the time specified by `store.expirestart` everyday. If `local.store.expire.workday` is not set, then the default is to run every day. There is no need to restart `stored` after changing this variable.

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.

To Add 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
```

To Move 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 (“To Add a Partition,” on page 360) and move mailboxes to the new partitions

If possible, we recommend adding additional disk space 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 the ProvisioningUsers Chapter in the *iPlanet Messaging Server Provisioning Guide*.

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 13, "Logging and Log Analysis."

This section contains the following:

- To Manage Mailboxes

- To Monitor Quota Limits
- To Monitor Disk Space
- Using the stored Utility
- Repairing Mailboxes and the Mailboxes Database

To Manage 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 “To Monitor Quota Limits” on page 366.

Table 11-4 lists the mboxutil commands. For detailed syntax and usage requirements, see the *Messaging Server Reference Manual*.

Table 11-4 `mboxutil` Options

Option	Description
`-a`	Lists all user quota information.
`-c` *mailbox*	Creates the specified mailbox.
`-d` *mailbox*	Deletes the specified mailbox.
`-f` *file*	Creates, deletes, or locks the mailbox or mailboxes listed in the specified data file.
`-g` *group*	Lists quota information for the specified group.

Table 11-4 `mboxutil` Options

Option	Description
`-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
```

To Monitor 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
```

To Monitor 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 11-5.

Table 11-5 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* and “Monitoring Disk Space,” on page 502

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 (see “stored,” on page 511).

- 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.

Table 11-6 lists the `stored` options. Some common usage examples follow the table. For detailed syntax and usage requirements, see the *Messaging Server Reference Manual*.

Table 11-6 `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.
`-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 become 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 11-7 lists the `reconstruct` options. For detailed syntax and usage requirements, see the *Messaging Server Reference Manual*.

Table 11-7 `reconstruct` Options

Option	Description
`-f`	Forces `reconstruct` to perform a fix on the mailbox or mailboxes.
`-m`	Repairs and performs a consistency check of the mailboxes database. This option examines every mailbox it finds in the spool area, adding or removing entries from the mailboxes database as appropriate. The utility prints a message to the standard output file whenever it adds or removes an entry from the database.

`-n`	Checks the message store only, without performing a fix on the mailbox or mailboxes. The `-n` option cannot be used by itself, unless a mailbox name is provided. When a mailbox name is not provided, the `-n` option must be used with the `-r` option; the `-r` option may be combined with the `-p` option. For example, any of the following commands are valid:
	`reconstruct -n user/dulcinea/INBOX`
	`reconstruct -n -r`
	`reconstruct -n -r -p primary`
	`reconstruct -n -r user/dulcinea/`

Table 11-7 `reconstruct` Options

Option	Description
`-o`	Checks for orphaned accounts. This option searches for inboxes in the current messaging server host which do not have corresponding entries in LDAP. For example, the `-o` option finds 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; do not use a full path name. If this option is not specified, `reconstruct` defaults to all partitions.
`-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*]	Repairs and performs a consistency check of the partition area of the specified mailbox or mailboxes. The `-r` option also repairs all sub-mailboxes within the specified mailbox. If you specify `-r` with no mailbox argument, the utility repairs the spool areas of all mailboxes within the user partition directory.

To Rebuild 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 372.) 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.

To Remove 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.

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:

- Move user mailboxes from one server to another
- 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 377.

This section contains the following subsections:

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

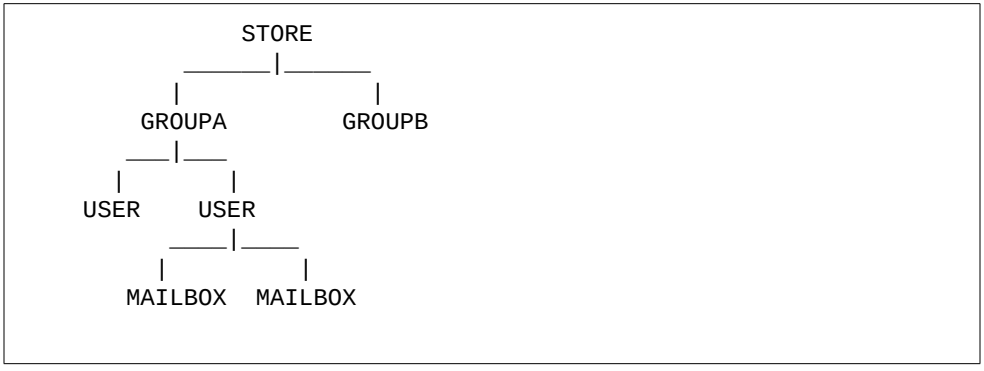
When user data is stored on multiple disks, you can back up user groups in parallel if you wish. Depending on system resources, parallel backups can speed up the overall backup procedure. However, you might want to use serial backups, for example, if you do not want to impact the server's performance. 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.

To Create 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 would represent a backup group while users whose last names begin with B 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 “To Create Backup Groups” on page 375.

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:

server_root`/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 comprehensive 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 “To Use Legato Networker” on page 379.

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.

To Use 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 *server_root*`/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 "To Create Backup Groups" on page 375.
   b. To verify your configuration, run `mkbackupdir.sh`.

      Look at the directory structure in *server_root*`/backup`. The structure should look similar to that shown in Figure 11-2.

   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 11-3 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 *server_root*`/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`. Therefore, you should use a short path name for the `-p` option of `mkbackupdir.sh`. For example the following command will create the backup image under the directory `/backup`:

```
mkbackupdir.sh -p /backup
```

Important: The backup directory must be writable by the message store owner (example: `mailsrv`).

Figure 11-2 shows a sample backup groups directory structure.

Figure 11-2 Backup Group Directory Structure

```
siroe-groupA-a1
            -a2
     -groupB-b1
            -b2
     -groupC-c1
            -c2
```

Figure 11-3 shows a sample `res` file named `IMS.res` in the `/nsr/res` directory:

Figure 11-3 Sample res File

```
type: savepnpc
precmd: "echo mkbackupdir started",
   "/usr/siroe/server5/msg-siroe/bin/mkbackupdir.sh -p /backup"
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 *server_root*`/backup`

      For parallel backups, use *server_root*`/backup/`*server/group*

      Be sure you've already created *group* as defined in "To Create Backup Groups" on page 375.

      You must also set the parallelism to the number of backup sessions.

      See "Example. Creating A Backup Client in Networker" on page 381.
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:/backup/siroe/groupA
   /backup/siroe/groupB
   /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 /backup/siroe/groupA/a1/INBOX
```

The next example recovers the entire message store:

```
recover -a -f -s siroe /backup/siroe
```

To Use a Third Party Backup Software (Besides Legato)

iPlanet Messaging Server provides two message store backup solutions, the command line `imsbackup` and the Solstice Backup (Legato Networker). A large message store running a single `imbackup` to backup the entire message store can take a significant amount of time. The Legato solution supports concurrent backup sessions on multiple backup devices. Concurrent backup can shorten backup time dramatically (backups of 25GB of data per hour have been achieved).

If you are using another third party concurrent backup software (for example, Netbackup), you may use the following method to integrate your backup software with the iPlanet Messaging Server.

1. Divide your users into groups (see "To Create Backup Groups," on page 375) and create a `backup-groups.conf` file under the directory *server_root*`/msg-<instance>/config/`.

   For example, to group users by UID, use the following definitions in `/usr/iplanet/server5/msg-siroe/config/backup-groups.conf`:

   ```
   groupA=a*
   groupB=b*
   groupC=c*
   . . .
   ```

   NOTE This backup solution requires additional disk space. To backup all the groups concurrently, the disk space requirement is 2 times the message store size. If you do not have that much disk space, divide your users into smaller groups, and then backup a set of groups at a time. For example group1 - group5, group6 - group10. Remove the group data files after backup.

2. Run `imsbackup` to backup each group into files under a staging area.

   The command is `imsbackup -f <device> /<instance>/<group>`

   You can run multiple `imsbackup` processes simultaneously. For example:

   ```
   # imsbackup -f- /siroe/groupA > /bkdata/groupA &
   # imsbackup -f- /siroe/groupB > /bkdata/groupB &

   . . .
   ```

`imsbackup` does not support large files, if the backup data is larger than 2 GB, you need to use the `-f-` option to write the data to `stdout` and then pipe the output to a file.

3. Use your third party backup software to backup the group data files in the staging area (in our example that is `/bkdata`).

4. To restore a user, identify the group filename of the user, restore that file from tape, and then use `imsrestore` to restore the user from the data file.

   Note that `imsrestore` does not support large files. If the data file is larger than 2GB. Use the following command:

   # **`cat /bkdata/groupA | imsrestore -f- /siroe/groupA/andy`**

Troubleshooting the Message Store

This section provides guidelines for pro-actively maintaining your message store. In addition, this section describes other message store recovery procedures you can use if the message store becomes corrupted or unexpectedly shuts down. Note that the section on these additional message store recovery procedures is an extension of “Repairing Mailboxes and the Mailboxes Database,” on page 369.

Prior to reading this section, it is strongly recommended that you review this chapter as well as the command-line utility and `configutil` chapters in the *iPlanet Messaging Server Reference Manual*. Topics covered in this section include:

- “Standard Message Store Monitoring Procedures,” on page 383
- “Common Problems and Solutions,” on page 386
- “Message Store Recovery Procedures,” on page 389

Standard Message Store Monitoring Procedures

This section outlines standard monitoring procedures for the message store. These procedures are helpful for general wellness checks, testing, and standard maintenance.

For additional information, see “Monitoring the Message Store,” on page 510.

Check Hardware Space

A message store should have enough additional disk space and hardware resources. When the message store is near the maximum limit of disk space and hardware space, problems might occur within the message store.

Inadequate disk space is one of the most common causes of the mail server problems and failure. Without space to write to the message store, the mail server will fail. In addition, when the available disk space goes below a certain threshold, there will be problems related to message delivery, logging, and so forth. Disk space can be rapidly depleted when the clean up function of the `stored` process fails and deleted messages are not expunged from the message store.

For information on monitoring disk space, see “To Monitor Disk Space,” on page 367 and “Monitoring the Message Store,” on page 510.

Check Log Files

Check the log files to make sure the message store processes are running as configured. Messaging Server creates a separate set of log files for each of the major protocols, or services, it supports: SMTP, IMAP, POP, and HTTP. You can look at the log files through the Console or in directory *server-root*`/msg-`*instance*`/log/`. You should monitor the log files on a routine basis.

Be aware that logging can impact server performance. The more verbose the logging you specify, the more disk space your log files will occupy for a given amount of time. You should define effective but realistic log rotation, expiration, and backup policies for your server. For information about defining logging policies for your server, see Chapter 13, “Logging and Log Analysis."

Check stored Processes

The `stored` function performs a variety of important tasks such as deadlock and transaction operations of the message database, enforcing aging policies, and expunging and erasing messages stored on disk. If `stored` stops running, Messaging Server will eventually run into problems. If `stored` doesn’t start when `start-msg` is run, no other processes will start.

- Check that the `stored` process is running. A `pid` file is created and updated by `stored` (*server-root*`/msg-`*instance*`/config/store.pid`).

- Check that the time stamps of the following files (in directory *server-root*/msg-*instance*/config/) are updated whenever one of the following functions are attempted by the stored process:

Table 11-8 stored Operations

stored Operation	Function
stored.ckp	Touched when a database checkpoint was initiated. Stamped approximately every 1 minute.
stored.lcu	Touched at every database log cleanup. Time stamped approximately every 5 minutes.
stored.per	Touched at every spawn of peruser db writeout. Time stamped once an hour.

- Check for the log file build up in *server-root*/msg-*instance*/store/mailboxlist.
- Check for stored messages in the default log file *server-root*/msg-*instance*/log/default/default

For more information on the stored process, see “Using the stored Utility,” on page 367 and the stored utility in the Messaging Server Command-line Utilities chapter of the *iPlanet Messaging Server Reference Manual*.

For additional information on monitoring the stored function, see “Monitoring the Message Store,” on page 510.

Check Database Log Files

Database log files refer to sleepycat transaction checkpointing log files (in directory *server-root*/msg-*instance*/store/mboxlist). If log files accumulate, then database checkpointing is not occurring. In general, there are two or three database log files during a single period of time. If there are more files, it could be a sign of a problem.

Check User Folders

If you want to check the user folders, you might run the command reconstruct -r -n (recursive nofix) which will review any user folder and report errors. For more information on the reconstruct command, see “Repairing Mailboxes and the Mailboxes Database,” on page 369.

Check for Core Files

Core files only exist when processes have unexpectedly terminated. It is important to review these files, particularly when you see a problem in the message store.

Common Problems and Solutions

This section lists common message store problems and solutions:

- "User Mailbox Directory Problems," on page 386
- "Global Store Problems," on page 387

User Mailbox Directory Problems

A user mailbox problem exists when the damage to the message store is limited to a small number of users, and there is no global damage to the system. The following guidelines suggest a process for identifying, analyzing, and resolving a user mailbox directory problem:

1. Review the log files, the error messages, or any unusual behavior that the user observes.
2. To keep debugging information and history, copy the entire *server-root*/msg-*instance*/store/mboxlist/ user directory to another location outside the message store.
3. To find the user folder that might be causing the problem, you should run the command `reconstruct -r -n`. If you are unable to find the folder using `reconstruct`, the folder might not exist in the `folder.db`.

   If you are unable to find the folder using the `reconstruct -r -n` command, use the `hashdir` command to determine the location. For more information on `hashdir`, see "The hashdir Utility," on page 365 and the `hashdir` utility in the Messaging Server Command-line Utilities chapter of the *iPlanet Messaging Server Reference Manual*.
4. Once you find the folder, examine the files, check permissions, and verify the proper file sizes.
5. Use `reconstruct -r` (without the `-n` option) to rebuild the mailbox.
6. If `reconstruct` does not detect a problem that you observe, you can force the reconstruction of your mail folders by using the `reconstruct -r -f` command.

7. If the folder does not exist in the `mboxlist` directory (*server-root*`/msg-`*instance*`/store/mboxlist`), but exists in `partition` directory *server-root*`/msg-`*instance*`/store/partition`), there might be a global inconsistency. In this case, you should run the `reconstruct -m` command.

8. If the previous steps do not work, you can remove the `store.idx` file and run the `reconstruct` command again.

CAUTION You should only remove the `store.idx` file if you are sure there is a problem in the file that the `reconstruct` command is unable to find.

9. If the issue is limited to a problematic message, you should copy the message file to another location outside of the message store and run the command `reconstruct -r` on the `mailbox/` directory.

10. If you determine the folder exists on the disk (*server-root*`/msg-`*instance*`/store/mboxlist/partition/` directory), but is apparently not in the database (*server-root*`/msg-`*instance*`/store/mboxlist/` directory), run the command `reconstruct -m` to ensure message store consistency.

For more information on the `reconstruct` command, see “Repairing Mailboxes and the Mailboxes Database” on page 369.

Global Store Problems

If you determine that the message store failure is a problem that is affecting all users or is a result of a global damage to a system, you can use the following guidelines to recover the system:

1. Stop the message store processes.

   a. Once you have verified that the message store processes have been stopped, restart the message store processes.

   b. Run the `stored` process to recover the database.

   In many instances, the database can automatically recover from a failure. This process occurs because when `stored` starts, it initiates a database recovery that analyzes database log files against cache files and database files. It attempts to put the database in a consistent state.

2. If the `msg-start` command unexpectedly stops while the `stored` process command is attempting to start the message store, `stored` either failed or is trying to recover the store.

   If this process abnormally ends while `stored` is attempting to start the message store, the `stored` process might be reviewing large log files in order to restore the database.

   a. Check the *server-root*`/msg-`*instance*`/log/default/` directory to review the information that `stored` has been analyzing.

   b. In addition, you can review the configuration and `pidfile.store` files.

      The `pidfile.store` file shows the `pid` as well as the state of the `stored` process. The `pidfile` shows an `init` state when recovering and a `ready` state when the `stored` process has finished repairing the database.

3. If the `pidfile` indicates a `ready` state, then the database has recovered, and the rest of the message store can restart.

   a. Start the store processes and run the `reconstruct -m` command. For more information on `reconstruct`, see “Repairing Mailboxes and the Mailboxes Database” on page 369.

   b. Determine if user mailbox directories are valid by monitoring test accounts and reviewing log files.

      If individual user mailboxes are damaged, run the `reconstruct -r` command.

   c. If damage to the message store is extensive, it might be necessary to repair with the message store processes stopped. See “Message Store Recovery Procedures” on page 389.

4. If the `pidfile` cannot change to the `ready` state, then the `stored` process is either reviewing the `mboxlist` log files, or the database cannot recover.

   a. If there are a number of database log files in the *server-root*`/msg-`*instance*`/store/mboxlist` directory, the `stored` process might not go beyond the `init` state. In addition, the database might take too long to recover (For example, twenty to thirty log files can take too long to process on most machines.). If this scenario occurs, you should stop the `stored` process, remove the files in the *server-root*`/msg-`*instance*`/store/mboxlist` directory, and initiate snapshot or fast recovery processes.

b. If the `stored` process cannot recover the message store, the database is most probably corrupted. You will then need to restore a snapshot copy of the database or initiate fast recovery techniques. For more information, see “Message Store Recovery Procedures” on page 389.

CAUTION You should never terminate a process while it is accessing the database. If you terminate the `stored` process while it is in the `init` state, you will not be able to recover the database from the existing `mboxlist` data. Consequently, the data must be removed. If you terminate another process that is accessing the database, the database might be left in an inconsistent state, and you will need to shut down the entire message store and restart it.

Message Store Recovery Procedures

This section describes recovery procedures to rebuild or repair the message store.

- **To Perform Fast Recovery.** Use fast recovery when the database is corrupted beyond standard repair. (See “Repairing Mailboxes and the Mailboxes Database” on page 369 for information on standard mailbox repair.) In addition, fast recovery allows the message store to be brought up immediately. As with the standard message store recovery procedure (See “Repairing Mailboxes and the Mailboxes Database” on page 369), you will also need to use the `reconstruct` command in the fast recovery process.
- **To Create Database Snapshot Backups and To Recover the Message Store with Database Snapshots.** If the database becomes damaged, a previous version of the database can be implemented, so that a high percentage of user folders can be immediately restored. After performing the restoration, you can use the fast recovery procedure with the `reconstruct` command to replace and rebuild the database.

To Perform Fast Recovery

When the database is inconsistent, you will use the `reconstruct` utility during a standard recovery. (See “Repairing Mailboxes and the Mailboxes Database” on page 369.)

If the database is corrupted beyond standard repair, you can use the `reconstruct` utility for fast recovery by following these procedures:

1. Stop the message store processes.

2. Verify all store processes have been stopped.

3. Copy the *server-root*/msg-*instance*/store/mboxlist/* files to a safe location to review at a later point.
4. Remove all of the files in the *server-root*/msg-*instance*/store/mboxlist/ directory.
5. Start the message store processes such as stored, imapd, popd, and mshttpd.
6. Run the utility reconstruct -m to rebuild the folder.db.

To Create Database Snapshot Backups

You can pro-actively anticipate message store corruptions by creating backups of the mailboxes database and log files (referred to as snapshots). In the event that the database becomes corrupt, you can use the snapshot to replace the database without having to reconstruct the database. The snapshot facility makes consistent copies of the database over time and can be recovered. Be sure you have enough disk space to keep these backups.

NOTE Unless otherwise specified, the database snapshot parameters listed in Table 11-9 should only be used with iPlanet Messaging Server 5.2.

Table 11-9 describes the three configutil parameters that are used to create database snapshots. These database snapshots are then invoked by the stored process during recovery:

Table 11-9 configutil Database Snapshot Parameters

Database Snapshot Parameter	Description
local.store.snapshotpath	Specifies the path in which to copy the mboxlist directory. Permissions are set for the message store owner. Snapshots will be placed in subdirectories.
local.store.snapshotinterval	Interval of time between snapshots. Unit of time is in minutes. It is recommended that you perform this procedure at least once a day.
local.store.snapshotdirs	Number of separate snapshots to store on disk. Minimum is 2, default is 3, recommend enough to be sure you have a good database back by the time you figure out the current one is beyond repair.

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.

CAUTION Database snapshots utilities from earlier Messaging Server releases do not function in the same way as these utilities. Therefore, older Messaging Server versions of snapshot utilities are not recommended for use with Messaging Server 5.2.

To Recover the Message Store with Database Snapshots

In order to recover the database utilizing database snapshots, it is imperative that you are familiar with the message store layout. For more information, see “Message Store Directory Layout” on page 344.

After the database snapshots are created (as explained in “To Create Database Snapshot Backups” on page 390), they are stored in `src` subdirectories. These files eventually are moved to the `dst` *server-root*`/msg-`*instance*`/store/mboxlist/` directory where the recovered database resides. In addition to the snapshot files, there are control files that are created while the snapshots are created. Table 11-10 describes the database snapshot control files. Note that these files are owned by the message store owner:

Table 11-10 Database Snapshot Control Files

Control File	Description
`dst/.nosnap`	Disables the database snapshot process even if configuration data is not refreshed.
`dst/.snaprst`	Marks all previous snapshots as invalid. This file is removed after the first new snapshot.
`dst/.catrecov`	Triggers the `stored` process to initiate a catastrophic recovery that restores the snapshot to a usable format.
`src/.snaptime`	Indicates a valid snapshot is in the directory. The time stamp of this file indicates when the snapshot completed.

The following steps explain how to perform a manual recovery by using database snapshots, control files, `src/`, and `dst/` directories:

1. Be sure that you are the message store owner prior to performing the recovery.
2. Stop the message store processes and verify all processes have been stopped.
3. Copy the files in *server-root*`/msg-`*instance*`/store/mboxlist/` directory to a safe location to review at a later time.
4. Review the snapshots you took to determine which, if any, can replace the message store. For more information, see "To Create Database Snapshot Backups" on page 390.
   a. Use the `*.snaptime` files to determine the validity and time of backup. If a snapshot has too many corresponding log files, review a different snapshot.
   b. Pick the latest valid snapshot that did not capture the database problem.

      If no snapshot is available, follow the fast recovery procedures. For more information, see "To Perform Fast Recovery" on page 389.
5. Remove all of the files in the *server-root*`/msg-`*instance*`/store/mboxlist/` directory, since they are corrupted.
6. Copy the corresponding snapshot files from the chosen snapshot to the *server-root*`/msg-`*instance*`/store/mboxlist/`directory, but be sure not to copy the `*.snaptime` files.
7. Use the `touch` command to create the `.catrecov` file in directory *server-root*`/msg-`*instance*`/store/mboxlist/`.

   A `.catrecov` file signals to the message store that a catastrophic recovery needs to be performed.
8. Start the message store processes.
9. Monitor the `stored` process. The `stored` process should recover.
10. Make sure that the *server-root*`/msg-`*instance*`/store/mboxlist/.catrecov` file has been removed after the `stored` process has recovered, otherwise the message store will assume it needs to do a catastrophic recovery whenever it starts up.
11. Run `reconstruct -m` to fix any differences between the `snaptime` file and the database failure.

Chapter 12

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

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 authenticated 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 11, "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 Appendix D, “Managing Mail Users and Mailing Lists.”
- **Filtering unsolicited bulk email (UBE):** See Chapter 10, “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 both user ID/password authentication and client certificate authentication. 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.)

The following techniques are used to improve the security of HTTP sessions:

- The session IDs are bound to a specific IP address.
- 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 3, “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 400.

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 400.

- **DIGEST-MD5** - A challenge / response authentication mechanism 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.

To Configure 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.

To Configure Directory Server to Store Passwords

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".

NOTE This change only impacts users created in the future. Existing users will have to transition or have their password reset after this change.

To 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 these challenge / response 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).

To Transition 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, internal 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 55.

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 “To Enable SSL and Selecting Ciphers,” on page 406), 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 “To Set Up Certificate-Based Login,” on page 408.

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, SASL, and TLS” on page 232.

You can use Authenticated SMTP with or without SSL encryption.

Configuring Encryption and Certificate-Based Authentication

This section contains the following subsections:

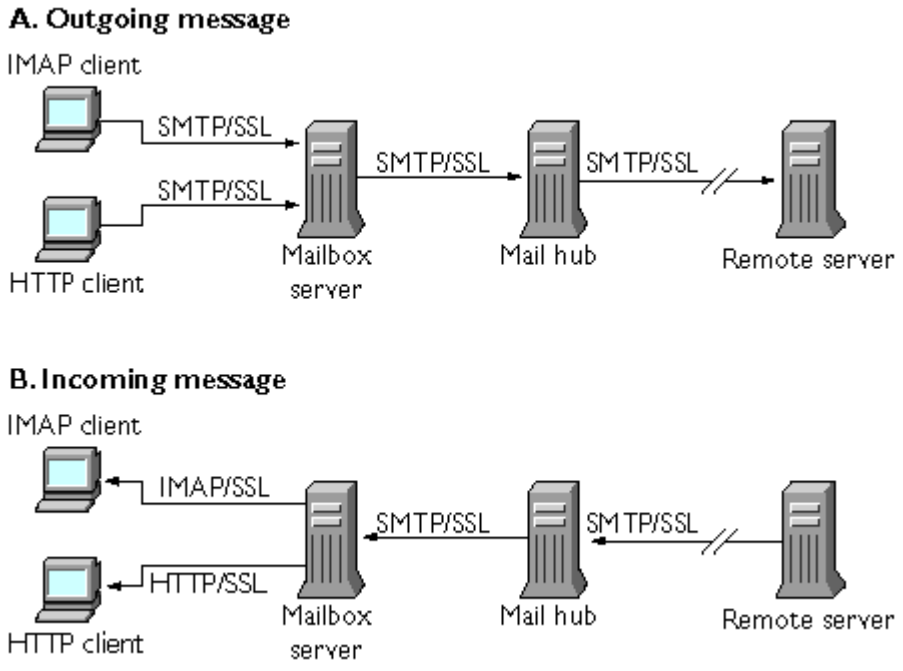
iPlanet Messaging Server uses the Transport Layer Security (TLS) protocol, otherwise known as the Secure Sockets Layer (SSL) protocol, for encrypted communications and for certificate-based authentication of clients and servers. iPlanet Messaging Server supports SSL versions 3.0 and 3.1. TLS is fully 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 12-1, for both outgoing and incoming messages.

Figure 12-1 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 many 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.

To Manage 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.

To Request 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*.)

To Install 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.

To Install 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.

To Enable 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 12-1 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 12-1 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.

To Set 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 402.)

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 “To Install Certificates of Trusted CAs,” on page 404.)

   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 “To Enable SSL and Selecting Ciphers,” on page 406.)

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*.

How to Optimize SSL Performance Using the SMTP Proxy

Most sites should not use the SMTP proxy as it adds additional latency to the SMTP protocol. However, a large-scale site which makes heavy use of SSL to protect SMTP connections may wish to maximize their investment in SSL accelerator hardware by performing all SSL operations for all protocols on a server which does nothing other than SSL and proxy. The SMTP proxy allows SSL to be processed by a front end proxy server while the mail queues are on a separate MTA machine. This way hardware optimized for each task can be separately configured and purchased.

See “To Install the SMTP Proxy,” on page 424 for instructions on how to install the SMTP Proxy.

Configuring Administrator Access to Messaging Server

This section contains the following subsections:

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 “To Restrict Access to Specific Tasks,” on page 412.

To Provide 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.

To Restrict 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

This section contains the following subsections:

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 10, "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 420 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 "To Create Access Filters for Services," on page 421 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 416.)

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 418 and “Client User-Name Specification” on page 418.)

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 417.

Wildcard Names

You can use the following wildcard names to represent service names, host names or addresses, or user names:

Table 12-2 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.

Table 12-2 Wildcard Names

Wildcard Name	Explanation
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`.
- `identd` is not supported by most modern client machines and thus provides little added value in modern deployments. We are considering removal of `identd` support in a future version, so please inform iPlanet if this feature is of value to your site.

- 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.

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.

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.

To Create Access Filters for Services

You can create Allow and Deny filters for the IMAP, POP, or HTTP services. You can also create them for SMTP services, but they have little value because they only apply to authenticated SMTP sessions. See Chapter 10, “Mail Filtering and Access Control” for how to control access to unauthenticated SMTP sessions.

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 415. For additional examples, see “Filter Examples” on page 419.

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 415.

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 415.

To Create 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 347.) 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 415.

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 415.

Enabling POP Before SMTP

SMTP Authentication, or *SMTP Auth* (RFC 2554) is the preferred method of providing SMTP relay server security. SMTP Auth allows only authenticated users to send mail through the MTA. However, some legacy clients only provide support for *POP before SMTP*. If this is the case for your system, you may enable POP before SMTP as described below. If possible, however, encourage your users to upgrade their POP clients rather than using POP before SMTP. Once POP before SMTP is deployed at a site users will become dependent on clients which fail to follow Internet security standards, putting end users at greater risk of hacking and slowing your site with the unavoidable performance penalty because of the necessity of having to track and coordinate IP addresses of recent successful POP sessions.

The iPlanet Messaging Server implementation of POP before SMTP is completely different from either SIMS or Netscape Messaging Server. POP before SMTP is supported by configuring a Messaging Multiplexor (MMP) to have both a POP and SMTP proxy. When an SMTP client connects to the SMTP proxy, the proxy will check an in-memory cache of recent POP authentications. If a POP authentication from the same client IP address is found, the SMTP proxy will inform the SMTP server that it should permit messages directed to both local and non-local recipients.

To Install the SMTP Proxy

1. Install an iPlanet Messaging Multiplexor (MMP) as described in the *iPlanet Messaging Server Installation Guide.*

2. Enable the SMTP proxy on the MMP.

   Add the string:

   server_root/bin/msg/mmp/lib/SmtpProxyAService@25|587

   to the ServiceList option in the *server_root*/mmp-*hostname*/AService.cfg file. That option is one long line and can't contain line breaks.

NOTE When the MMP is upgraded, four new files which correspond to the existing four configuration files for the MMP. The new files are:

AService-def.cfg, ImapProxyAService-def.cfg, PopProxyAService-def.cfg, and SmtpProxyAService-def.cfg

These files are created by the installer, the four configuration files described in the docs are not created or affected by the install process. When the MMP starts up, it will look for the normal configuration file (as currently documented). If it doesn't find the normal configuration file, it will attempt to copy the respective *AService-def.cfg file to the corresponding *AService.cfg file name.

---

3. Set the PROXY_PASSWORD option in the SMTP channel option file tcp_local_option at each SMTP relay server.

   When the SMTP proxy connects to the SMTP server, it has to inform the SMTP server of the real client IP address and other connection information so that the SMTP server can correctly apply relay blocking and other security policy (including POP before SMTP authorization). This is a security sensitive operation and must be authenticated. The proxy password configured on both the MMP SMTP Proxy and the SMTP server assures that a third party cannot abuse the facility.

   Example: PROXY_PASSWORD *A_Password*

4. Configure the SMTP proxy to Support POP before SMTP.

   a. Edit the *server_root*/mmp-*instance*/SmtpProxyAService.cfg configuration file.

The following SMTP proxy options operate identically to the same options for the IMAP and POP proxies (see the appendix entitled, “Installing the Messaging Multiplexor” in the *iPlanet Messaging Server Installation Guide* and the description of these options in the Encryption (SSL) Option section in the *iPlanet Messaging Server Reference Manual* or more information):

`LdapURL`, `LogDir`, `LogLevel`, `BindDN`, `BindPass`, `Timeout`, `Banner`, `SSLEnable`, `SSLSecmodFile`, `SSLCertFile`, `SSLKeyFile`, `SSLKeyPasswdFile`, `SSLCipherSpecs`, `SSLCertNicknames`, `SSLCacheDir`, `SSLPorts`, `CertMapFile`, `CertmapDN`, `ConnLimits`, `TCPAccess`

Other MMP options not listed above (including the `BacksidePort` option) do not currently apply to the SMTP Proxy.

Add the following five options:

`SmtpRelays` is a space-separated list of SMTP relay server hostnames (with optional port) to use for round-robin relay. These relays must support the `XPROXYEHLO` extension. This option is mandatory with no default.
Example: `default:SmtpRelays  manatee:485 gonzo mothra`

`SmtpProxyPassword` is a password used to authorize source channel changes on the SMTP relay servers. This option is mandatory with no default and must match the `PROXY_PASSWORD` option on the SMTP servers.
Example: `default:SmtpProxyPassword` *A_Password*

`EhloKeywords` option provides a list of EHLO extension keywords for the proxy to pass through to the client, in addition to the default set. The MMP will remove any unrecognized EHLO keywords from the EHLO list returned by an SMTP relay. `EhloKeywords` specifies additional EHLO keywords which should not be removed from the list. The default is empty, but the SMTP proxy will support the following keywords, so there is no need to list them in this option: `8BITMIME`, `PIPELINING`, `DSN`, `ENHANCEDSTATUSCODES`, `EXPN`, `HELP`, `XLOOP`, `ETRN`, `SIZE`, `STARTTLS`, `AUTH`

The following is an example that might be used by a site which uses the rarely used “TURN” extension:
Example: `default:EhloKeywords  TURN`

`PopBeforeSmtpKludgeChannel` option is set to the name of an MTA channel to use for POP before SMTP authorized connections. The default is empty and the typical setting for users who want to enable POP before SMTP is `tcp_intranet`. This option is not required for optimizing SSL performance (see “How to Optimize SSL Performance Using the SMTP Proxy,” on page 410).
Example: `default:PopBeforeSmtpKludgeChannel  tcp_intranet`

`ClientLookup` option defaults to `no`. If set to `yes`, a DNS reverse lookup on the client IP address will be performed unconditionally so the SMTP relay server doesn't have to do that work This option may be set on a per hosted domain domain basis.
Example: `default:ClientLookup  yes`

b. Set the `PreAuth` option and the `AuthServiceTTL` option in `PopProxyAService.cfg` configuration file. This option is not required for optimizing SSL performance. (See "How to Optimize SSL Performance Using the SMTP Proxy," on page 410.)

NOTE `AuthServiceTTL` must **not** be set in IMAP or SMTP proxy configuration files in order for POP before SMTP to work.

These options specify how long in seconds a user is authorized to submit mail after a POP authentication. The typical setting is 900 to 1800 (15-30 minutes).

Example:

```
default:PreAuth    yes
default:AuthServiceTTL    900
```

c. You may optionally specify how many seconds the MMP will wait for an SMTP Relay to respond before trying the next one in the list.

The default is 10 (seconds). If a connection to an SMTP Relay fails, the MMP will avoid trying that relay for a number of minutes equivalent to the failover time-out (so if the failover time-out is 10 seconds, and a relay fails, the MMP won't try that relay again for 10 minutes).
Example: `default:FailoverTimeout  10`

Configuring Client Access to SMTP Services

For information about configuring client access to SMTP services, see Chapter 10, "Mail Filtering and Access Control."

Chapter 13

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," on page 427

"PART 2: Service Logs (Message Store, Administration Server, and MTA)," on page 429

"PART 3: Service Logs (MTA)," on page 440

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. It consists of the following sections:

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 13-1 lists the services that can be logged, and describes the log files for each service.

Table 13-1 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, Administration Server, and MTA)

This section describes logging for the following services: POP, IMAP, HTTP, MTA, Admin, and Default (see Table 13-1).

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. For additional information on service logs for the MTA see “PART 3: Service Logs (MTA),” on page 440.

Part 2 contains the following sections:

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 by setting the `logfile.`*service*`.loglevel` configuration parameter (see “Defining and Setting Logging Options” on page 434). You can also use logging levels to filter searches for log events. Table 13-2 describes the available levels. These logging levels are a subset of those defined by the UNIX `syslog` facility.

Table 13-2 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.
Information	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.
Debug	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 434.

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 13-3 lists the categories that Messaging Server recognizes for logging purposes.

Table 13-3 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 438.

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 13-4 lists the message store log filename conventions.

Table 13-4 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).

Table 13-4 Filename Conventions for Store and Administration Logs

Component	Definition
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 434.

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 434.

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 13-5 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 13-5 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 13-3 on page 431).
logLevel	The level of logging that the event represents: for example, `Notice` (see Table 13-2 on page 430).
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 438.

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.)

To Set 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`, `imta`, 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`, `imta`, 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 429). 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 430). 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 432) 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

To Specify 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. It also provides dispatcher error and debugging output. Part 3 contains the following sections:

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.

To Enable 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`

In addition, you can also set a number of configuration parameters such as directory path for log files, log levels, an so on. See “PART 2: Service Logs (Message Store, Administration Server, and MTA),” on page 429.

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 (UNIX) or event log (Windows NT) by setting the `LOG_MESSAGES_SYSLOG` option to 1. 0 is the default and indicates that syslog (event log) logging is not performed.

To Specify 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 13-1.

Figure 13-1 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 13-6.
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 13-6 describes the logging entry codes.

Table 13-6 Logging Entry Codes

Entry	Description
D	Successful dequeue
DA	Successful dequeue with SASL (authentication)

Table 13-6 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 13-2. (The sample log entry line has been wrapped for typographic reasons; the actual log entry would appear on one physical line.)

Figure 13-2 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 228. 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 “To Specify Additional MTA Logging Options” on page 442.

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 13-3 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 13-3 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 13-4 shows a logging entry similar to that shown in Figure 13-3, 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 13-4 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 13-5 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 13-5 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 13-6 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 13-6 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 13-7 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 13-6, 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 13-7 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 13-8 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 320) 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 13-8 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 13-6 and Figure 13-7).
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 13-9 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 13-9 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 13-6, 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 13-10 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 13-10 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 13-11 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 13-11 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 13-12 illustrates log output for an incoming SMTP message when connection logging is enabled, via `LOG_CONNECTION=3`.

Figure 13-12 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.

Dispatcher Debugging and Log Files

Dispatcher error and debugging output (if enabled) are written to the file `dispatcher.log` in the MTA log directory.

Debugging output may be enabled using the option `DEBUG` in the Dispatcher configuration file, or on a per-process level, using the `IMTA_DISPATCHER_DEBUG` environment variable (UNIX).

The `DEBUG` option or `IMTA_DISPATCHER_DEBUG` environment variable (UNIX) defines a 32-bit debug mask in hexadecimal. Enabling all debugging is done by setting the option to -1, or by defining the logical or environment variable system-wide to the value `FFFFFFFF`. The actual meaning of each bit is described in Table 13-7.

Table 13-7 Dispatcher Debugging Bits

Bit	Hexadecimal value	Decimal value	Usage
0	x 00001	1	Basic Service Dispatcher main module debugging.
1	x 00002	2	Extra Service Dispatcher main module debugging.
2	x 00004	4	Service Dispatcher configuration file logging.
3	x 00008	8	Basic Service Dispatcher miscellaneous debugging.
4	x 00010	16	Basic service debugging.

Table 13-7 Dispatcher Debugging Bits *(Continued)*

Bit	Hexadecimal value	Decimal value	Usage
5	x 00020	32	Extra service debugging.
6	x 00040	64	Process related service debugging.
7	x 00080	128	Not used.
8	x 00100	256	Basic Service Dispatcher and process communication debugging.
9	x 00200	512	Extra Service Dispatcher and process communication debugging.
10	x 00400	1024	Packet level communication debugging.
11	x 00800	2048	Not used.
12	x 01000	4096	Basic Worker Process debugging.
13	x 02000	8192	Extra Worker Process debugging.
14	x 04000	16384	Additional Worker Process debugging, particularly connection hand-offs
15	x 08000	32768	Not used.
16	x 10000	65536	Basic Worker Process to Service Dispatcher I/O debugging.
17	x 20000	131072	Extra Worker Process to Service Dispatcher I/O debugging.
20	x 100000	1048576	Basic statistics debugging.
21	x 200000	2097152	Extra statistics debugging.
24	x 1000000	16777216	Log PORT_ACCESS denials to the dispatcher.log file.

System Parameters on Solaris

The system's heap size (`datasize`) must be enough to accommodate the Dispatcher's thread stack usage. For each Dispatcher service compute `STACKSIZE*MAX_CONNS`, and then add up the values computed for each service. The system's heap size needs to be at least twice this number.

The Dispatcher services offered in the Dispatcher configuration file affects requirements for various system parameters.

To display the heap size (that is, default `datasize`), use the `csh` command:

```
# limit
```

or the `ksh` command

```
# ulimit -a
```

or the utility

```
# sysdef
```

Chapter 14

Troubleshooting the MTA

This chapter describes common tools, methods, and procedures for troubleshooting the Message Transfer Agent (MTA). It consists of the following sections:

A related topic, monitoring procedures can be found in Chapter 15, "Monitoring the iPlanet Messaging Server."

NOTE Prior to reading this chapter, you should review Chapters 6 through 10 in this guide and the MTA configuration and command-line utility chapters in the *iPlanet Messaging Server Reference Manual.*

Troubleshooting Overview

One of the first steps in troubleshooting the MTA is to determine where to begin the diagnosis. Depending on the problem, you might look for error messages in log files. In other situations, you might check all of the standard MTA processes, review the MTA configuration, or start and stop individual channels. Whatever

approach you use, consider the following questions when troubleshooting the MTA:

- Did configuration or environmental problems prevent messages from being accepted (for example, disk space or quota problems)?
- Were MTA services such as the Dispatcher and the Job Controller present at the time the message entered the message queue?
- Did network connectivity or routing problems cause messages to be stuck or misrouted on a remote system?
- Did the problem occur before or after a message entered into the message queue?

This chapter will address these questions in the subsequent sections.

Standard MTA Troubleshooting Procedures

This section outlines standard troubleshooting procedures for the MTA. Follow these procedures if a problem does not generate an error message, if an error message does not provide enough diagnostic information, or if you want to perform general wellness checks, testing, and standard maintenance of the MTA.

- “Check the MTA Configuration,” on page 467
- “Check the Message Queue Directories,” on page 467
- “Check the Ownership of Critical Files,” on page 467
- “Check That the Job Controller and Dispatcher are Running,” on page 468
- “Check the Log Files,” on page 470
- “Run a Channel Program Manually,” on page 471
- “Starting and Stopping Individual Channels,” on page 471
- “An MTA Troubleshooting Example,” on page 473

Check the MTA Configuration

Test your address configuration by using the `imsimta test -rewrite` utility. With this utility, you can test the MTA's address rewriting and channel mapping without actually having to send a message. Refer to the MTA Command-line Utilities chapter in the *iPlanet Messaging Server Reference Manual* for more information.

The utility will normally show address rewriting that will be applied as well as the channel to which messages will be queued. However, syntax errors in the MTA configuration will cause the utility to issue an error message. If the output is not what you expect, you may need to correct your configuration.

Check the Message Queue Directories

Check if messages are present in the MTA message queue directory, typically */server-root*`/msg-`*instance*`/imta/queue/`. Use command-line utilities like `imsimta qm` to check for the presence of expected message files under the MTA message queue directory. For more information on `imsimta qm`, refer to the MTA command-line utilities chapter in the *iPlanet Messaging Server Reference Manual* and "imsimta qm counters," on page 520.

If the `imsimta test -rewrite` output looks correct, check that messages are actually being placed in the MTA message queue subdirectories. To do so, enable message logging (For more information on MTA logging, see "PART 3: Service Logs (MTA)," on page 440). You should then look at the `mail.log_current` file in the directory */server-root*`/msg-`*instance*`/log/imta/`. You can track a specific message by its message ID to ensure that it is being placed in the MTA message queue subdirectories. If you are unable to find the message, you may have a problem with file disk space or directory permissions.

Check the Ownership of Critical Files

You should have selected a mail server user account (nobody by default) when you installed iPlanet Messaging Server. The following directories, subdirectories, and files should be owned by this account:

/server-root`/msg-`*instance*`/imta/queue/`
/server-root`/msg-`*instance*`/log/imta/`
/service-root`/msg-`*instance*`/imta/tmp`

Commands, like the ones in the following UNIX system example, may be used to check the protection and ownership of these directories:

```
ls -l -p -d /usr/iplanet/server5/msg-budgie/imta/queue
drwx------ 6 nobody bin 512 Feb 7 09:32
/usr/iplanet/server5/msg-budgie/imta/queue

ls -l -p -d /usr/iplanet/server5/msg-budgie/log/imta
drwx------ 2 nobody bin 1536 Mar 10 09:00
/usr/iplanet/server5/msg-budgie/log/imta

ls -l -p -d /usr/iplanet/server5/msg-budgie/imta/tmp
drwx------ 2 nobody bin 512 Feb 7 10:00
/usr/iplanet/server5/msg-budgie/imta/tmp
```

Check that the files in */server-root*/msg-*instance*/imta/queue are owned by the MTA account by using commands like in the following UNIX system example:

```
ls -l -p -R /usr/iplanet/server5/msg-budgie/imta/queue
```

Check That the Job Controller and Dispatcher are Running

The MTA Job Controller handles the execution of the MTA processing jobs, including most outgoing (master) channel jobs.

Some MTA channels, such as the MTA's multi-threaded SMTP channels, include resident server processes that process incoming messages. These servers handle the slave (incoming) direction for the channel. The MTA Dispatcher handles the creation of such MTA servers. Dispatcher configuration options control the availability of the servers, the number of created servers, and how many connections each server can handle.

To check that the Job Controller and Dispatcher are present, and to see if there are MTA servers and processing jobs running, use the command `imsimta process`. Under idle conditions the command should result in `job_controller` and `dispatcher` processes. For example:

```
imsimta process

USER      PID S   VSZ    RSS    STIME     TIME   COMMAND
mailsrv  9567  S  18416   9368   02:00:02    0:00 /opt/iplanet/
  server5/bin/msg/imta/bin/tcp_smtp_server

mailsrv  6573  S  18112   5720   Jul_13      0:00 /opt/iplanet/
  server5/bin/msg/imta/bin/job_controller

mailsrv  9568 S  18416   9432   02:00:02   0:00 /opt/iplanet/
  server5/bin/msg/imta/bin/tcp_smtp_server

mailsrv  6574 S  17848   5328   Jul_13     0:00 /opt/iplanet/
  server5/bin/msg/imta/bin/dispatcher
```

If the Job Controller is not present, the files in the */server-root/msg-*`instance/imta/queue` directory will get backed up and messages will not be delivered. If you do not have a Dispatcher, then you will be unable to receive any SMTP connections.

For more information on `imsimta process`, refer to the *iPlanet Messaging Server Reference Manual*.

If neither the Job Controller nor the Dispatcher is present, you should review the `dispatcher.log-*` or `job_controller.log-*` file in the */server-root/*`msg-`*instance*`/log/imta/`directory.

If the log files do not exist or do not indicate an error, start the processes by using the `imsimta start` command. For more information, refer to the MTA command-line utilities chapter in the *iPlanet Messaging Server Reference Manual*.

NOTE You should not see multiple instances of the Dispatcher or Job Controller when you run `imsimta process`.

Check the Log Files

If MTA processing jobs run properly but messages stay in the message queue directory, you can examine the log files to see what is happening. All MTA log files are created in the directory */server-root*/msg-*instance*/log/imta. Log file name formats for various MTA processing jobs are shown in Table 14-1.

Table 14-1 MTA Log Files

File Name	Log File Contents
*channel*_master.log-*uniqueid*	Output of master program (usually client) for *channel.*
*channel*_slave.log-*uniqueid*	Output of slave program (usually server) for *channel.*
dispatcher.log-*uniqueid*	Dispatcher debugging. This log is created regardless if the Dispatcher DEBUG option is set. However, to get detailed debugging information, you should set the DEBUG option to a non-zero value.
imta	ims-ms channel error messages when there is a problem in delivery.
job_controller.log-*uniqueid*	Job controller logging. This log is created regardless if the Job Controller DEBUG option is set. However, to get detailed debugging information, you should set the DEBUG option to a non-zero value.
tcp_smtp_server.log-*uniqueid*	Debugging for the tcp_smtp_server. The information in this log is specific to the server, not to messages.
return.log-*uniqueid*	Debug output for the periodic MTA message bouncer job; this log file is created if the return_debug option is used in the option.dat

NOTE Each log file is created with a unique ID (*uniqueid*) to avoid overwriting an earlier log created by the same channel. To find a specific log file, you can use the imsimta view utility. You can also purge older log files by using the imsimta purge command. For more information, see the MTA command-line utilities chapter in the *iPlanet Messaging Server Reference Manual*.

The *channel*_master.log-*uniqueid* and *channel*_slave.log-*uniqueid* log files will be created in any of the following situations:

- There are errors in your current configuration.

- The `master_debug` or `slave_debug` keywords are set on the channel in the `imta.cnf` file.
- If `mm_debug` is set to a non-zero value (`mm_debug > 0`) in your `option.dat` file (in the directory: `/`*server-root*`/`*msg*`-instance/imta/config/`).

For more information on debugging channel master and slave programs, see the *iPlanet Messaging Server Reference Manual*.

Run a Channel Program Manually

When diagnosing an MTA delivery problem it is helpful to manually run an MTA delivery job, particularly after you enable debugging for one or more channels.

The command `imsimta submit` will notify the MTA Job Controller to run the channel. If debugging is enabled for the channel in question, `imsimta submit` will create a log file in directory `/`*server-root*`/msg-`*instance*`/log/imta` as shown in Table 14-1.

The command `imsimta run` will perform outbound delivery for the channel under the currently active process, with output directed to your terminal. This may be more convenient than submitting a job, particularly if you suspect problems with job submission itself.

NOTE In order to manually run channels, the Job Controller must be running.

For information on syntax, options, parameters, examples of `imsimta submit` and `imsimta run` commands, refer to the MTA command-line utility chapter in the *iPlanet Messaging Server Reference Manual*.

Starting and Stopping Individual Channels

In some cases, stopping and starting individual channels may make message queue problems easier to diagnose and debug. Stopping a message queue allows you to examine queued messages to determine the existence of loops or spam attacks.

To Stop Outbound Processing (dequeueing) for a Specific Channel

1. Use the `imsimta qm stop` command to stop a specific channel. Doing so prevents you from having to stop the Job Controller and having to recompile the configuration. In the following example, the `conversion` channel is stopped:

   `imsimta qm stop conversion`

2. To resume processing, use the `imsimta qm start` command to restart the channel. In the following example, the `conversion` channel is started:

   `imsimta qm start conversion`

For more information on the `imsimta qm start` and `imsimta qm stop` commands, see the chapter on MTA command-line utilities in the *iPlanet Messaging Server Reference Manual*.

To Stop Inbound Processing from a Specific Domain or IP Address (enqueuing to a channel)

You can run one of the following processes if you want to stop inbound message processing for a specific domain or IP address, while returning temporary SMTP errors to client hosts. By doing so, messages will not be held on your system. Refer to the “PART 1. MAPPING TABLES,” on page 305.

- To stop inbound processing for a specific host or domain name, add the following access rule to the ORIG_SEND_ACCESS mapping table in the MTA mappings file (typically */server-root*`/msg-`*instance*`/imta/config/mappings`):

```
ORIG_SEND_ACCESS

    *|*@sesta.com|*|*              $X4.2.1|$NHost$ blocked
```

  By using this process, the sender’s remote MTA will hold messages on their systems, continuing to resend them periodically until you restart inbound processing.

- To stop inbound processing for a specific IP address, add the following access rule to the PORT_ACCESS mapping table in the MTA mappings file (typically */server-root*/msg-*instance*/imta/config/mappings):

```
PORT_ACCESS

    TCP|*|25|IP_address_to_block|*                    $N500$ unable$ to$ \
    connect$ at$ this$ time
```

When you want to restart inbound processing from the domain or IP address, be sure to remove these rules from the mapping tables and recompile your configuration. In addition, you may want to create unique error messages for each mapping table. Doing so will enable you to determine which mapping table is being used.

An MTA Troubleshooting Example

This section explains how to troubleshoot a particular MTA problem step-by-step. In this example, a mail recipient did not receive an attachment to an email message. Note: In keeping with MIME protocol terminology, the “attachment” is referred to as a “message part” in this section. The aforementioned troubleshooting techniques are used to identify where and why the message part disappeared (See “Standard MTA Troubleshooting Procedures,” on page 466). By using the following steps, you can determine the path the message took through the MTA. In addition, you can determine if the message part disappeared before or after the message entered the message queue. To do so, you will need to manually stop and run channels, capturing the relevant files.

NOTE The Job Controller must be running when you manually run messages through the channels.

Identify the Channels in the Message Path

By identifying which channels are in the message path, you can apply the `master_debug` and `slave_debug` keywords to the appropriate channels. These keywords generate debugging output in the channels’ master and slave log files; in turn, the master and slave debugging information will assist in identifying the point where the message part disappeared.

1. Add `log_message_id=1` in the `option.dat` file in directory */server-root*`/msg-`*instance*`/imta/config`. With this parameter, you will see message ID: header lines in the `mail.log_current` file.
2. Run `imsimta cnbuild` to recompile the configuration.
3. Run `imsimta restart dispatcher` to restart the SMTP server.
4. Have the end user resend the message with the message part.
5. Determine the channels that the message passes through.

   While there are different approaches to identifying the channels, the following approach is recommended:

   a. On UNIX platforms, use the `grep` command to search for message ID: header lines in the `mail.log_current` file in directory */server-root*`/msg-`*instance*`/log/imta/`. On Windows NT platforms, use the `find` command.

   b. Once you find the message ID: header lines, look for the E (enqueue) and D (dequeue) records to determine the path of the message. Refer to "MTA Log Entry Format," on page 443 for more information on logging entry codes. See the following E and D records for this example:

```
29-Aug-2001 10:39:46.44 tcp_local conversion     E 2 ...

29-Aug-2001 10:39:46.44 conversion tcp_intranet  E 2 ...

29-Aug-2001 10:39:46.44 tcp_intranet             D 2 ...
```

   The channel on the left is the source channel, and the channel on the right is the destination channel. In this example, the E and D records indicate that the message's path went from the `tcp_local` channel to the `conversion` channel and finally to the `tcp_intranet` channel.

Manually Start and Stop Channels to Gather Data

This section describes how to manually start and stop channels. See "Starting and Stopping Individual Channels," on page 471 for more information. By manually starting and stopping the channels in the message's path, you are able to save the message and log files at different stages in the MTA process. These files are later used to "Identify the Point of Message Breakdown," on page 477.

1. Set the `mm_debug=5` in the `option.dat` file in directory */server-root*`/msg-`*instance*`/imta/config` in order to provide substantial debugging information.
2. Add the `slave_debug` and `master_debug` keywords to the appropriate channels in the `imta.cnf` file in directory */server-root*`/msg-`*instance*`/imta/config`.
   a. Use the `slave_debug` keyword on the inbound channel (or any channel where the message is switched to during the initial dialog) from the remote system that is sending the message with the message part. In this example, the `slave_debug` keyword is added to the `tcp_local` channel.
   b. Add the `master_debug` keyword to the other channels that the message passed through and were identified in "Identify the Channels in the Message Path," on page 473. In this example, the `master_debug` keyword would be added to the `conversion` and `tcp_intranet` channels.
   c. Run the command `imsimta restart dispatcher` to restart the SMTP server.
3. Use the `imsimta qm stop` and `imsimta qm start` commands to manually start and stop specific channels. For more on information by using these keywords, see "Starting and Stopping Individual Channels," on page 471.
4. To start the process of capturing the message files, have the end user resend the message with the message part.
5. When the message enters a channel, the message will stop in the channel if it has been stopped with the `imsimta qm stop` command. For more information, see Step 3.
   a. Copy and rename the message file before you manually run the next channel in the message's path. See the following UNIX platform example:

      `# cp ZZ01K7LXW76T7O9TD0TB.00 ZZ01K7LXW76T7O9TD0TB.KEEP1`

      The message file typically resides in directory similar to */server-root*`/msg-`*instance*`/imta/queue/`*destination_channel*`/001`. The *destination_channel* is the next channel that the message passes through (such as: `tcp_intranet`). If you want to create subdirectories (like `001`, `002`, and so on) in the *destination_channel* directory, add the `subdirs` keyword to the channels.
   b. It is recommended that you number the extensions of the message each time you trap and copy the message in order to identify the order in which the message is processed.

6. Resume message processing in the channel and enqueue to the next destination channel in the message's path. To do so, use the `imsimta qm start` command.

7. Copy and save the corresponding channel log file (for example: `tcp_intranet_master.log-*`) located in directory */server-root*`/msg-`*instance*`/log/imta/`. Choose the appropriate log file that has the data for the message you are tracking. Make sure that the file you copy matches the timestamp and the subject header for the message as it comes into the channel. In the example of the `tcp_intranet_master.log-*`, you might save the file as `tcp_intranet_master.keep` so the file is not deleted.

8. Repeat steps 5 - 7 until the message has reached its final destination.

   The log files you copied in Step 7 should correlate to the message files that you copied in Step 5. If, for example, you stopped all of the channels in the missing message part scenario, you would save the `conversion_master.log-*` and the `tcp_intranet_master.log-*` files. You would also save the source channel log file `tcp_local_slave.log-*`. In addition, you would save a copy of the corresponding message file from each destination channel: `ZZ01K7LXW76T7O9TD0TB.KEEP1` from the `conversion` channel and `ZZ01K7LXW76T7O9TD0TB.KEEP2` from the `tcp_intranet` channel.

9. Remove debugging options once the message and log files have been copied.

   a. Remove the `slave_debug` and the `master_debug` keywords from the appropriate channels in the `imta.cnf` file in directory */server-root*`/msg-`*instance*`/imta/config`.

   b. Reset the `mm_debug=0`, and remove `log_message_id=1` in the `option.dat` file in directory */server-root*`/msg-`*instance*`/imta/config`.

   c. Recompile the configuration by using `imsimta cnbuild`.

   d. Run the command `imsimta restart dispatcher` to restart the SMTP server.

Identify the Point of Message Breakdown

1. By the time you have finished starting and stopping the channel programs, you should have the following files with which you can use to troubleshoot the problem:

   a. All copies of the message file (for example: `ZZ01K7LXW76T7O9TD0TB.KEEP1`) from each channel program

   b. A `tcp_local_slave.log-*` file

   c. A set of *channel*`_master.log-*` files for each destination channel

   d. A set of `mail.log_current` records that show the path of the message

   All files should have timestamps and message ID values that match the message ID: header lines in the `mail.log_current` records. Note that the exception is when messages are bounced back to the sender; these bounced messages will have a different message ID value than the original message.

2. Examine the `tcp_local_slave.log-*` file to determine if the message had the message part when it entered the message queue.

   Look at the SMTP dialog and data to see what was sent from the client machine.

   If the message part did not appear in the `tcp_local_slave.log-*` file, then the problem occurred before the message entered the MTA. As a result, the message was enqueued without the message part. If this the case, the problem could have occurred on the sender's remote SMTP server or in the sender's client machine.

3. Investigate the copies of the message files to see where the message part was altered or missing.

   If any message file showed that the message part was altered or missing, examine the previous channel's log file. For example, you should look at the `conversion_master.log-*` file if the message part in the message entering the `tcp_intranet` channel was altered or missing.

4. Look at the final destination of the message.

   If the message part looks unaltered in the `tcp_local_slave.log`, the message files (for example: `ZZ01K7LXW76T7O9TD0TB.KEEP1`), and the *channel*`_master.log-*` files, then the MTA did not alter the message and the message part is disappearing at the next step in the path to its final destination.

   If the final destination is the `ims-ms` channel (the Message Store), then you might download the message from the server to a client machine to determine if the message part is being dropped during or after this transfer. If the destination channel is a `tcp_*` channel, then you need to go to the MTA in the message's path. Assuming it is an iPlanet Messaging Server MTA, you will need to repeat the entire troubleshooting process (See "Identify the Channels in the Message Path," on page 473, "Manually Start and Stop Channels to Gather Data," on page 474, and this section). If the other MTA is not under your administration, then the user who reported the problem should contact that particular site.

Common MTA Problems and Solutions

This sections lists common problems and solutions for MTA configuration and operation.

Changes to Configuration Files or MTA Databases Do Not Take Effect

If changes to your configuration, mapping, conversion, security, option, or alias files are not taking effect, check to see if you have performed the following steps:

1. Recompile the configuration (by running `imsimta cnbuild`).
2. Restart the appropriate processes (like `imsimta restart dispatcher`).
3. Re-establish any client connections.

The MTA Sends Outgoing Mail but Does Not Receive Incoming Mail

Most MTA channels depend upon a slave or channel program to receive incoming messages. For some transport protocols that are supported by the MTA (like TCP/IP and UUCP), you need to make sure that the transport protocol activates the MTA slave program rather than its standard server. Replacing the native `sendmail` SMTP server with the MTA SMTP server is performed as a part of the iPlanet Messaging Server installation. Refer to the *iPlanet Messaging Server Installation Guide for UNIX* for more information.

For the multi-threaded SMTP server, the startup of the SMTP server is controlled by the Dispatcher. If the Dispatcher is configured to use a `MIN_PROCS` value greater than or equal to one for the SMTP service, then there should always be at least one SMTP server process running (and potentially more, according to the `MAX_PROCS` value for the SMTP service). The `imsimta process` command may be used to check for the presence of SMTP server processes. See the chapter on MTA command-line utilities in the *iPlanet Messaging Server Reference Manua*l for more information.

Timeouts on Incoming SMTP connections

Timeouts on incoming SMTP connections are most often related to system resources and their allocation. The following techniques can be used to identify the causes of timeouts on incoming SMTP connections:

1. Check how many simultaneous incoming SMTP connections you allow. This is controlled by the `MAX_PROCS` and `MAX_CONNS` Dispatcher settings for the SMTP service; the number of simultaneous connections allowed is `MAX_PROCS*MAX_CONNS`. If you can afford the system resources, consider raising this number if it is too low for your usage.

2. Another technique you can use is to open a TELNET session. In the following example, the user connects to `127.0.0.1` port 25. Once connected, 220 banner is returned. For example:

```
telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
220 budgie.sesta.com -- Server ESMTP (iPlanet Messaging Server
5.1 (built May  7 2001))
```

   If you are connected and receive a 220 banner, but additional commands (like `ehlo` and `mail from`) do not illicit a response, then you should run `imsimta test -rewrite` to ensure that the configuration is correct. If you are using the `imsimta dirsync` command, make sure the command has recently run. On occasion, if `dirsync` fails, the commands in the SMTP server do not receive responses. In this case, running `imsimta dirsync -F` will clear the issue as long as the `dirsync` lock files are first removed.

3. If the response time of the 220 banner is slow, and if running the `pstack` command on the SMTP server shows the following `iii_res*` functions (These functions indicate that a name resolution lookup is being performed.):

```
febe2c04 iii_res_send (fb7f4564, 28, fb7f4de0, 400, fb7f458c,
fb7f4564) + 142c
febdfdcc iii_res_query (0, fb7f4564, c, fb7f4de0, 400, 7f) + 254
```

   then it is likely that the host has to do reverse name resolution lookups, even on a common pair like `localhost/127.0.0.1`. To prevent such a performance slowdown, you should reorder your host's lookups in the `/etc/nsswitch.conf` file. To do so, change the following line in the `/etc/nsswitch.conf` file from:

```
hosts: dns nis [NOTFOUND=return] files
```

   to:

```
hosts: files dns nis [NOTFOUND=return]
```

   Making this change in the `/etc/nsswitch.conf` file can improve performance. Fewer SMTP servers have to handle messages instead of multiple SMTP servers having to perform unnecessary lookups.

4. You can also put the `slave_debug` keyword on the channels handling incoming SMTP over TCP/IP mail, usually `tcp_local` and `tcp_intranet`. After doing so, review the most recent `tcp_local_slave.log-`*uniqueid* files to identify any particular characteristics of the messages that time out. For example, if incoming messages with large numbers of recipients are timing out, consider using the `expandlimit` keyword on the channel.

Remember that if your system is overloaded and overextended, timeouts will be difficult to avoid entirely.

Messages are Not Dequeued

Errors encountered during TCP/IP delivery are often transient; the MTA will generally retain messages when problems are encountered and retry them periodically. It is normal on large networks to experience periodic outages on certain hosts while other host connections work fine. To verify the problem, examine the log files for errors relating to delivery attempts. You may see error messages such as, “Fatal error from smtp_open.” Such errors are not uncommon and are usually associated with a transient network problem. To debug TCP/IP network problems, use utilities like PING, TRACEROUTE, and NSLOOKUP.

The following example shows the steps you might use to see why a message is sitting in the queue awaiting delivery to `xtel.co.uk`. To determine why the message is not being dequeued, you can recreate the steps the MTA uses to deliver SMTP mail on TCP/IP.

```
% nslookup -query=mx xtel.co.uk (1)

Server: LOCALHOST
Address: 127.0.0.1

Non-authoritative answer:
XTEL.CO.UK  preference = 10, mail exchanger = nsfnet-relay.ac.uk
(2)

% telnet nsfnet-relay.ac.uk 25 (3)
Trying... [128.86.8.6]
telnet: Unable to connect to remote host: Connection refused
```

1. Use the NSLOOKUP utility to see what MX records, if any, exist for this host. If no MX records exist, then you should try connecting directly to the host. If MX records do exist, then you must connect to the designated MX relays. The MTA honors MX information preferentially, unless explicitly configured not to do so. See also “TCP/IP MX Record Support,” on page 230.
2. In this example, the DNS (Domain Name Service) returned the name of the designated MX relay for `xtel.co.uk`. This is the host to which the MTA will actually connect. If more than one MX relay is listed, the MTA will try each MX record in succession, with the lowest preference value tried first.

3. If you do have connectivity to the remote host, you should check if it is accepting inbound SMTP connections by using TELNET to the SMTP server port 25.

NOTE If you use TELNET without specifying the port, you will discover that the remote host accepts normal TELNET connections. This does not indicate that it accepts SMTP connections; many systems accept regular TELNET connections but refuse SMTP connections and vice versa. Consequently, you should always do your testing against the SMTP port.

In the previous example, the remote host is refusing connections to the SMTP port. This is why the MTA fails to deliver the message. The connection may be refused due to a misconfiguration of the remote host or some sort of resource exhaustion on the remote host. In this case, nothing can be done to locally to resolve the problem. Typically, you should let the MTA continue to retry the message.

If you are running iPlanet Messaging Server on a TCP/IP network that does not use DNS, you can skip steps (1) and (2). Instead, you can use TELNET to directly access the host in question. Be careful to use the same host name that the MTA would use. Look at the relevant log file from the MTA's last attempt to determine the host name. If you are using host files, you should make sure that the host name information is correct. It is strongly recommended that you use DNS instead of host names.

Note that if you test connectivity to a TCP/IP host and encounter no problems using interactive tests, it is quite likely that the problem has simply been resolved since the MTA last tried to deliver the message. You can re-run the `imsimta submit tcp_channel` on the appropriate channel to see if messages are being dequeued.

MTA Messages are Not Delivered

In addition to message transport problems, there are two common problems which can result in unprocessed messages in the message queues:

1. The queue cache is not synchronized with the messages in the queue directories. Message files in the MTA queue subdirectories that are awaiting delivery are entered into an in-memory queue cache. When channel programs run, they consult this queue cache to determine which messages to deliver in their queues. There are circumstances where there are message files in the queue, but there is no corresponding queue cache entry.

   a. To check if a particular file is in the queue cache, you can use the `imsimta cache -view` utility; if the file is not in the queue cache, then the queue cache needs to be synchronized.

      The queue cache is normally synchronized every four hours. If required, you can manually resynchronize the cache by using the command `imsimta cache -sync`. Once synchronized, the channel programs will process the originally unprocessed messages after new messages are processed. If you want to change the default (4 hours), you should modify the `job_controller.cnf` file in directory */server-root*`/msg-`*instance*`/imta/config` by adding `sync_time=`*timeperiod* where *timeperiod* reflects how often the queue cache is synchronized. Note that the *timeperiod* must be greater than 30 minutes. In the following example, the queue cache synchronization is modified to 2 hours by adding the `sync_time=02:00` to the global defaults section of the `job_controller.cnf`:

```
! VERSION=5.0
!IMTA job controller configuration file
!
!Global defaults
tcp_port=27442
secret=N1Y9[HzQKW
slave_command=NULL
sync_time=02:00
```

      You can run `imsimta submit` *channel* to clear out the backlog of messages after running `imsimta cache -sync`. It is important to note that clearing out the channel may take a long time if the backlog of messages is large (greater than 1000).

For summarized queue cache information, run `imsimta qm -maint dir -database -total`.

b. If after synchronizing the queue cache, messages are still not being delivered, you should restart the Job Controller. To do so, use the `imsimta restart job_controller` command.

Restarting the Job Controller will cause the message data structure to be rebuilt from the message queues on disk.

CAUTION Restarting the Job Controller is a drastic step and should only be performed after all other avenues have been thoroughly exhausted.

Refer “The Job Controller,” on page 105 for more information on the Job Controller.

2. Channel processing programs fail to run because they cannot create their processing log file. Check the access permissions, disk space and quotas.

Messages are Looping

If the MTA detects that a message is looping, that message will be sidelined as a .HELD file. See “Diagnosing and Cleaning up .HELD Messages,” on page 486. Certain cases can lead to message loops which the MTA can not detect.

The first step is to determine why the messages are looping. You should look at a copy of the problem message file while it is in the MTA queue area, MTA mail log entries (if you have the `logging` channel keyword enabled in your MTA configuration file for the channels in question) relating to the problem message, and MTA channel debug log files for the channels in question. Determining the From: and To: addresses for the problem message, seeing the Received: header lines, and seeing the message structure (type of encapsulation of the message contents), can all help pinpoint which sort of message loop case you are encountering.

Some of the more common cases include:

1. A postmaster address is broken.

The MTA requires that the postmaster address be a functioning address that can receive email. If a message to the postmaster is looping, check that your configuration has a proper postmaster address pointing to an account that can receive messages.

2. Stripping of Received: header lines is preventing the MTA from detecting the message loop.

   Normal detection of message loops is based on Received: header lines. If Received: header lines are being stripped (either explicitly on the MTA system itself, or on another system like a firewall), it can interfere with proper detection of message loops. In these scenarios, check that no undesired stripping of Received: header lines is occurring. Also, check for the underlying reason why the messages are looping. Possible reasons include: a problem in the assignment of system names or a system not configured to recognize a variant of its own name, a DNS problem, a lack of authoritative addressing information on the system in question, or a user address forwarding error.

3. Incorrect handling of notification messages by other messaging systems are generating reencapsulated messages in response to notification messages.

   Internet standards require that notification messages (reports of messages being delivered, or messages bouncing) have an empty envelope From: address to prevent message loops. However, some messaging systems do not correctly handle such notification messages. When forwarding or bouncing notification messages, these messaging systems may insert a new envelope From: address. This can then lead to message loops. The solution is to fix the messaging system that is incorrectly handling the notification messages.

Diagnosing and Cleaning up .HELD Messages

If the MTA detects that messages are bouncing between servers or channels, delivery is halted and the messages are stored in a file with the suffix .HELD in */server-root*/msg-*instance*/imta/queue/*channel*. Typically, a message loop occurs because each server or channel thinks the other is responsible for delivery of the message.

For example, an end user may set an option to forward messages on two separate mail hosts to one another. On his `sesta.com` account, the end-user enables mail forwarding to his `varrius.com` account. And, forgetting that he has enabled this setting, he sets mail forwarding on his `varrius.com` account to his `sesta.com` account.

A loop can also occur with a faulty MTA configuration. For example, MTA Host X thinks that messages for `mail.sesta.com` go to Host Y. However, Host Y thinks that Host X should handle messages for `mail.sesta.com`; as a result, Host Y returns the mail to Host X.

In these cases, the message is ignored by the MTA and no further delivery is attempted. When such a problem occurs, look at the header lines in the message to determine which server or channel is bouncing the message. Fix the entry as needed.

You can also retry the .HELD message by following these steps:

1. Rename the .HELD extension to any 2 digit number other than 00. For example, .HELD to .06.

NOTE Before renaming the .HELD file, be sure that the message has stopped looping.

2. Run `imsimta cache -sync`. Running this command will update the cache.
3. Run `imsimta submit` *channel* or `imsimta run` *channel*.

It may be necessary to perform these steps multiple times, since the message may again be marked as .HELD, because the Received: header lines accumulate.

Received Message is Encoded

Messages sent by the MTA are received in an encoded format. For example:

```
Date: Wed, 04 Jul 2001 11:59:56 -0700 (PDT)
From: "Desdemona Vilalobos" <Desdemona@sesta.com>
To: santosh@varrius.com
Subject: test message with 8bit data
MIME-Version: 1.0
Content-type: TEXT/PLAIN; CHARSET=ISO-8859-1
Content-transfer-encoding: QUOTED-PRINTABLE

2=00So are the Bo=F6tes Void and the Coal Sack the same?=
```

These messages appear unencoded when read with the MTA decoder command `imsimta decode`. Refer to the *iPlanet Messaging Server Reference Manual* for more information.

The SMTP protocol only allows the transmission of ASCII characters (a seven-bit character set) as set forth by RFC 821. In fact, the unnegotiated transmission of eight-bit characters is illegal via SMTP, and it is known to cause a variety of problems with some SMTP servers. For example, SMTP servers can go into compute bound loops. Messages are sent over and over again. Eight-bit characters can crash SMTP servers. Finally, eight-bit character sets can wreak havoc with browsers and mailboxes that cannot handle eight-bit data.

An SMTP client used to only have three options when handling a message containing eight-bit data: return the message to the sender as undeliverable, encode the message, or send it in direct violation of RFC 821. But with the advent of MIME and the SMTP extensions, there are now standard encodings which may be used to encode eight-bit data by using the ASCII character set.

In the previous example, the recipient received an encoded message with a MIME content type of TEXT/PLAIN. The remote SMTP server (to which the MTA SMTP client transferred the message) did not support the transfer of eight-bit data. Since the original message contained eight-bit characters, the MTA had to encode the message.

Server-Side Rules (SSR) Are Not Working

A filter consists of one or more conditional actions to apply to a mail message. Since the filters are stored and evaluated on the server, they are often referred to as server-side rules (SSR).

The directory synchronization command (`imsimta dirsync`) updates the MTA's SSR database with information about users' filters. The SSR database stores short filters (less than 1016 bytes) and an LDAP DN is used for long filters. Note that the MTA does not recognize changes to a user's filters until after the `imsimta dirsync` command has updated the directory server. For more information on SSR, refer to "PART 2. MAILBOX FILTERS," on page 331.

This section includes information on the following SSR topics:

Testing Your SSR Rules

- To check the MTA's user filters, use the command:

```
# imsimta test -rewrite -debug -filter user@domain
```

In the output, look for the following information:

```
mmc_open_url called to open ssrf:user@ims-ms
   URL with quotes stripped: ssrd:user@ims-ms
Determined to be a SSRD URL.
   Identifier: user@ims-ms-daemon
Filter successfully obtained.
```

- In addition, you can add the `slave_debug` keyword to the `tcp_local` channel to see how a filter is applied. The results are displayed in the `tcp_local_slave.log` file. Be sure to add `mm_debug=5` in the `option.dat` file in directory */server-root*`/msg-`*instance*`/imta/config` in order to get sufficient debugging information.

Troubleshooting Procedures

When diagnosing SSR problems be sure that you have verified the following procedures.

- If you are using the `imsimta dirsync` command, make sure that the `ims-ms` channel is marked filter
  `ssrd:$a`
  and
  `fileinto $u+$s@$d`
  in the `imta.cnf` file in directory */server-root*`/msg-`*instance*`/imta/config`.

- If you are using the `imsimta dirsync` command, make sure that the `imsimta dirsync` command properly synchronizes filter information. To do so, run the following command from directory */server-root*`/msg-`*instance*`/`. Be sure you are running the command as the messaging server user:

```
# configutil -l -o service.imta.ssrenabled -v true
OK SET
# configutil | fgrep ssr
local.imta.ssrenabled = yes
service.imta.ssrenabled = true
```

Common Syntax Problems

- If there is a syntax problem with the filter, then look for the following message in the `tcp_local_slave.log-*` file:
  `Error parsing filter expression:...`
  - If the filter is good, then filter information will be at the end of the output.
  - If the filter is bad, then the following error will be at the end of the output:
    ```
    Address list error -- 4.7.1 Filter syntax error:
        desdaemona@sesta.com
    ```
    Also, if the filter is bad, then the SMTP RCPT TO command will return a temporary error response code:

```
RCPT TO:user@domain
452 4.7.1 Filter syntax error
```

General Error Messages

When the MTA fails to start, general error messages appear at the command line. In this section, common general error messages will be described and diagnosed.

NOTE To diagnose your own MTA configuration, use the `imsimta test -rewrite -debug` utility to examine your MTA's address rewriting and channel mapping process. By using this utility allows you to check the configuration without actually sending a message. See "Check the MTA Configuration," on page 467.

MTA subcomponents might also issue other error messages that are not described in this chapter. You should refer to the chapters on MTA command-line utilities and configuration in the *iPlanet Messaging Server Reference Manual* and chapters 6 through 10 for more information on each subcomponent. This section includes the following types of errors:

Errors in mm_init

An error in mm_init generally indicates an MTA configuration problem. If you run the `imsimta test -rewrite` utility, these errors will be displayed. Other utilities like `imsimta cnbuild`, a channel, a server, or a browser might also return such an error.

Commonly encountered mm_init errors include:

• “official host name is too long,” on page 494

bad equivalence for alias. . .

The right hand side of an alias file entry is improperly formatted.

cannot open alias include file. . .

A file included into the alias file cannot be opened.

duplicate aliases found. . .

Two alias file entries have the same left hand side. You will need to find and eliminate the duplication. Look for an error message that says `error line #XXX` where `XXX` is a line number. You can fix the duplicated alias on the line.

duplicate host in channel table. . .

This error message indicates that you have two channel definitions in the MTA configuration that both have the same official host name.

Note that an extraneous blank line in the rewrite rules (upper portion) of your MTA configuration file (`imta.cnf`) causes the MTA to interpret the remainder of the configuration file as channel definitions. Make sure that the very first line of the file is not a blank. Since there are often multiple rewrite rules with the same pattern (left-hand side), this then causes MTA to interpret them as channel definitions with non-unique official host names. Check your MTA configuration for any channel definitions with duplicate official host names and for any improper blank lines in the upper (rewrite rules) portion of the file.

duplicate mapping name found. . .

This message indicates that two mapping tables have the same name, and one of the duplicate mapping tables needs to be removed. However, formatting errors in the mapping file may cause the MTA to wrongly interpret something as a mapping table name. For example, failure to properly indent a mapping table entry will cause the MTA to think that the left hand side of the entry is actually a mapping table name. Check your mapping file for general form and check the mapping table names.

NOTE A blank line should precede and follow any line with a mapping table name. However, no blank lines should be interspersed among the entries of a mapping table.

mapping name is too long. . .

This error means that a mapping table name is too long and needs to be shortened. Formatting errors in the mapping file may cause the MTA to wrongly interpret something as a mapping table name. For example, failure to properly indent a mapping table entry will cause the MTA to think that the left hand side of the entry is actually a mapping table name. Check your mapping file and mapping table names.

error initializing ch_ facility: compiled character set version mismatch

If you see this message, you need to recompile and reinstall your compiled character set tables through the command `imsimta chbuild`. See the *iPlanet Messaging Server Reference Manual* for more information.

error initializing ch_ facility: no room in. . .

This error message generally means that you need to resize your MTA character set internal tables and then rebuild the compiled character set tables with the following commands:

```
imsimta chbuild -noimage -maximum -option
imsimta chbuild
```

Verify that nothing else needs to be recompiled or restarted before making this change. Refer to the MTA command-line utilities chapter in the *iPlanet Messaging Server Reference Manual* for more information on `imsimta chbuild`.

local host alias or proper name too long for system. . .

This error indicates that a local host alias or proper name is too long (the optional right hand side in the second or subsequent names in a channel block). However, certain syntax errors earlier in the MTA configuration file (an extraneous blank line in the rewrite rules, for instance) may cause MTA to wrongly interpret something as a channel definition. Aside from checking the indicated line of the configuration file, also check above that line for other syntax errors. In particular, if the line in which MTA issues this error is intended as a rewrite rule, then be sure to check for extraneous blank lines above it.

no equivalence addresses for alias. . .

An entry in the alias file is missing a right hand side (translation value).

no official host name for channel. . .

This error indicates that a channel definition block is missing the required second line (the official host name line). See the chapters on MTA configuration and command-line utilities in the *iPlanet Messaging Server Reference Manual* and Chapter 8, “Configuring Channel Definitions.” for more information on channel definition blocks. A blank line is required before and after each channel definition block, but a blank line must not be present between the channel name and official host name lines of the channel definition. Also note that blank lines are not permitted in the rewrite rules portion of the MTA configuration file.

official host name is too long

The official host name for a channel (second line of the channel definition block) is limited to forty octets in length. If you are trying to use a longer official host name on a channel, shorten it to a place holder name, and then use a rewrite rule to match the longer name to the short official host name. You may see this scenario if you work with the `l` (local) channel host name. For example:

```
Original l Channel:
!delivery channel to local /var/mail store
l subdirs 20 viaaliasrequired maxjobs 7 pool LOCAL_POOL
newt.salamander.lizard.gecko.komododragon.com

Create Place Holder:
!delivery channel to local /var/mail store
l subdirs 20 viaaliasrequired maxjobs 7 pool LOCAL_POOL
newt

Create Rewrite Rule:
newt.salamander.lizard.gecko.komododragon.com   $U%$D@newt
```

Note that when using the `l` (local) channel, you will need to use a REVERSE mapping table. Refer to the MTA configuration chapter in the *iPlanet Messaging Server Reference Manual* for information on usage and syntax.

Certain syntax errors earlier in the MTA configuration file (for example, an extraneous blank line in the rewrite rules) may cause the MTA to wrongly interpret something as a channel definition. This could result in an intended rewrite rule being interpreted as an official host name. Besides checking the indicated line of the configuration file, also check above that line for other syntax errors. In particular, if the line on which the MTA issues this error is intended as a rewrite rule, be sure to check for extraneous blank lines above it.

Compiled Configuration Version Mismatch

One of the functions of the `imsimta cnbuild` utility is to compile MTA configuration information into an image that can be quickly loaded. The compiled format is quite rigidly defined and often changes substantially between different versions of the MTA. Minor changes might occur as part of patch releases.

When such changes occur, an internal version field is also changed so that incompatible formats can be detected. The MTA components will halt with the above error when an incompatible format is detected. The solution to this problem is to generate a new, compiled configuration with the command `imsimta cnbuild`.

It is also a good idea to use the `imsimta restart` command to restart any resident MTA server processes, so they can obtain updated configuration information.

Swap Space Errors

To ensure proper operation, it is important to configure enough swap space on your messaging system. The amount of required swap space will vary depending on your configuration. A general tuning recommendation is that the amount of swap space should be at least three times the amount of main memory.

An error message such as the following indicates a lack of swap space:

```
jbc_channels: chan_execute [1]: fork failed: Not enough space
```

You might see this error in the Job Controller log file. Other swap space errors will vary depending on your configuration.

Use the following commands to determine how much swap space you have left and determine how much you have used:

- Solaris systems: `swap -s` (at the time MTA processes are busy), `ps -elf`, or `tail /var/adm/messages`
- HP-UX systems: `swapinfo` or `tail /var/adm/syslog/syslog.log`
- Windows NT systems: You can set a paging file size on drives other than the default hard drive (for example `C:\`), if you need more room or if you have a faster drive elsewhere. To check the available space or to set a new paging file size, follow these steps:
  - Go to the Control Panel and click System Properties (or System).
  - Click the Performance Tab.

- Click Change in the Virtual Memory section.
- The Virtual Memory window will provide the paging file size for each drive.

File open or create errors

In order to send a message, the MTA reads configuration files and creates message files in the MTA message queue directories. Configuration files must be readable by the MTA or any program written against the MTA's SDKs. During installation, proper permissions are assigned to these files. The MTA utilities and procedures which create configuration files also assign permissions. If the files are protected by the system manager, other privileged user, or through some site-specific procedure, the MTA may not be able to read configuration information. This will result in "File open" errors or unpredictable behavior. The `imsimta test -rewrite` utility reports additional information when it encounters problems reading configuration files. See the `imsimta test -rewrite` documentation in the MTA chapters of the *iPlanet Messaging Server Reference Manual*.

If the MTA appears to function from privileged accounts but not from unprivileged accounts, then file permissions in the MTA table directory are likely the cause of the problem. Check the permissions on configuration files and their directories. See "Check the Ownership of Critical Files," on page 467.

"File create" errors usually indicate a problem while creating a message file in an MTA message queue directory. See "Check the Message Queue Directories," on page 467 to diagnose file creation problems.

Illegal Host/Domain Errors

You may see this error when an address is provided to the MTA through a browser. Or, the error may be deferred and returned as part of an error return mail message. In both cases, this error message indicates that the MTA is not able to deliver mail to the specified host. To determine why the mail is not being sent to the specified host, you should follow these troubleshooting procedures:

- Verify that the address in question is not misspelled, is not transcribed incorrectly, or does not use the name of a host or domain that no longer exists.

- Run the address in question through the `imsimta test -rewrite` utility. If this utility also returns an "illegal host/domain" error on the address, then MTA has no rules in the `imta.cnf` file and related files to handle the address. Verify that you have configured MTA correctly, that you answered all configuration questions appropriately, and that you have kept your configuration information up to date.
- If `imsimta test -rewrite` does not encounter an error on the address, then MTA is able to determine how to handle the address, but the network transport will not accept it. You can examine the appropriate log files from the delivery attempt for additional details. Transient network routing or name service errors should not result in returned error messages, though it is possible for badly misconfigured domain name servers to cause these problems.
- If you are on the Internet, check that you have properly configured your TCP/IP channel to support MX record lookups. Many domain addresses are not directly accessible on the Internet and require that your mail system correctly resolve MX entries. If you are on the Internet and your TCP/IP is configured to support MX records, you should have configured the MTA to enable MX support; see TCP/IP Connection and DNS Lookup Support "TCP/IP Connection and DNS Lookup Support," on page 224 for more information. If your TCP/IP package is not configured to support MX record lookups, then you will not be able to reach MX-only domains.

Errors in SMTP channels: os_smtp_* errors

Errors such as the following are not necessarily MTA errors: os_smtp_* errors like os_smtp_open, os_smtp_read, and os_smtp_write errors. These errors are generated when the MTA reports a problem encountered at the network layer. For example, an os_smtp_open error means that the network connection to the remote side could not be opened. The MTA may be configured to connect to an invalid system because of addressing errors or channel configuration errors. The os_smtp_* errors are commonly due to DNS or network connectivity problems, particularly if this was a previously working channel or address. os_smtp_read or os_smtp_write errors are usually an indication that the connection was aborted by the other side or due to network problems.

Network and DNS problems are often transient in nature. The occasional os_smtp_* error is usually nothing to be concerned about. However, if you are consistently seeing these errors, it may be an indication of an underlying network problem.

To obtain more information about a particular os_smtp_* error, enable debugging on the channel in question. Investigate the debug channel log file that will show details of the attempted SMTP dialogue. In particular, look at the timing of when a network problem occurred during the SMTP dialogue. The timing may suggest the type of network or remote side issue. In some cases, you may also want to perform network level debugging (for example, TCP / IP packet tracing) to determine what was sent or received.

Chapter 15

Monitoring the iPlanet Messaging Server

In most cases, a well-planned, well-configured server will perform without extensive intervention from an administrator. As an administrator, however, it is your job to monitor the server for signs of problems. This chapter describes the monitoring of the iPlanet Messaging Server. It consists of the following sections:

Troubleshooting procedures can be found in Chapter 14, “Troubleshooting the MTA.

Daily Monitoring Tasks

The most important tasks you should perform on a daily basis are checking postmaster mail, monitoring the log files, and setting up the `stored` utility. These tasks are described below.

Checking postmaster Mail

Messaging Server has a predefined administrative mailing list set up for postmaster email. Any users who are part of this mailing list will automatically receive mail addressed to postmaster.

The rules for postmaster mail are defined in RFC822, which requires every email site to accept mail addressed to a user or mailing list named postmaster and that mail sent to this address be delivered to a real person. All messages sent to postmaster@host.domain are sent to a postmaster account or mailing list.

Typically, the postmaster address is where users should send email about their mail service. As postmaster, you might receive mail from local users about server response time, from other server administrators who are encountering problems sending mail to your server, and so on. You should check postmaster mail daily.

You can also configure the server to send certain error messages to the postmaster address. For example, when the MTA cannot route or deliver a message, you can be notified via email sent to the postmaster address. You can also send exception condition warnings (low disk space, poor server response) to postmaster.

Monitoring and Maintaining the Log Files

iPlanet Messaging Server creates a separate set of log files for each of the major protocols, or services, it supports: SMTP, IMAP, POP, and HTTP. You should monitor the log files on a routine basis--especially if you are having problems with the server.

Be aware that logging can impact server performance. The more verbose the logging you specify, the more disk space your log files will occupy for a given amount of time. You should define effective but realistic log rotation, expiration, and backup policies for your server. For information about defining logging policies for your server, see Chapter 13, “Logging and Log Analysis.”

Setting Up the `stored` Utility

The `stored` utility performs automatic monitoring and maintenance tasks for the server, such as:

- 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 midnight. For further information see “stored,” on page 511.

Monitoring System Performance

This chapter focuses on iPlanet Messaging Server monitoring, however, you will also need to monitor the system on which the server resides. A well-configured server cannot perform well on a poorly-tuned system, and symptoms of server failure may be an indication that the hardware is not powerful enough to serve the email load. This chapter does not provide all the details for monitoring system performance as many of these procedures are platform specific and may require that you refer to the platform specific system documentation. The following procedures are described here for performance monitoring:

- “Monitoring End-to-end Message Delivery Times,” on page 501
- “Monitoring Disk Space,” on page 502
- “Monitoring CPU Usage,” on page 503

Monitoring End-to-end Message Delivery Times

Email needs to be delivered on time. This may be a service agreement requirement, but also it is good policy to have mail delivered as quickly as possible. Slow end-to-end times could indicate a number of things. It may be that the server is not working properly, or that certain times of the day experience overwhelming message loads, or that the existing hardware resources are being pushed beyond their capacity.

Symptoms of Poor End-to-end Message Delivery Times

Mail takes a longer period of time to be delivered than normal.

To Monitor End-to-end Message Delivery Times

Use any facility that sends a message and receives it. Compare the headers times between server hops, and times between point of origin and retrieval.

Monitoring Disk Space

Inadequate disk space is one of the most common causes of the mail server problems and failure. Without space to write to the MTA queues or to the message store, the mail server will fail. In addition, unless log files are monitored and cleaned up, they can grow uncontrollably filling up all disk space.

Disk space can be rapidly depleted when the clean up function of `stored` fails and deleted messages are not expunged from the message store. Other causes of running out of disk space are the MTA message queues growing too large, the message store outgrowing the available disk space, and unmonitored log files growing uncontrollably. (Note that there are a number of log files such as LDAP, MTA, and Message Access, and that each of these log files can be stored on different disks.)

Symptoms of Disk Space Problems

Different symptoms can occur depending on which disk or partition is running out of space. MTA queues can overflow and reject SMTP connections, messages might remain in the ims_master queue and not be not delivered to the message store, and log files can overflow.

To Monitor Disk Space

Depending upon the system configuration you may need to monitor various disks and partitions. For example, MTA queues may reside on one disk/partition, message stores may reside on another, and log files may reside on yet another. Each of these spaces will require monitoring and the methods to monitor these spaces may differ.

Monitoring the Message Store

It is recommended that message store disk usage not exceed 75% capacity. You can monitor message store disk usage by configuring the following alarm attributes using the `configutil` utility:

- `alarm.diskavail.msgalarmstatinterval`
- `alarm.diskavail.msgalarmthreshold`
- `alarm.diskavail.msgalarmwarninginterval`

By setting these parameters, you can specify how often the system should monitor disk space and under what circumstances the system should send a warning. 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
```

Refer to Table 15-1 on page 512 for more information on these parameters.

Monitoring the MTA Queues and Logging Space

You will need to monitor MTA queue disk and logging space disk usage.

Monitoring CPU Usage

High CPU usage is either a sign that there is not enough CPU capacity for the level of usage or some process is using up more CPU cycles than is appropriate.

Symptoms of CPU Usage Problems

Poor system response time. Slow logging in of users. Slow rate of delivery.

To Monitor CPU Usage

Monitoring CPU usage is a platform specific task. Refer to the relevant platform documentation.

Monitoring the MTA

This section consists of the following subsections:

Monitoring the Size of the Message Queues

Excessive message queue growth may indicate that messages are not being delivered, are being delayed in their delivery, or are coming in faster than the system can deliver them. This may be caused by a number of reasons such as a denial of service attack caused by huge numbers of messages flooding your system, or the Job Controller not running.

See “Channel Message Queues,” on page 103, “Messages are Not Dequeued,” on page 482, and “MTA Messages are Not Delivered,” on page 484 for more information on message queues.

Symptoms of Message Queue Problems

- Disk space usage grows.
- User not receiving messages in a reasonable time.
- Message queue sizes are abnormally high.

To Monitor the Size of the Message Queues

Probably the best way to monitor the message queues is to use `imsimta qm`. Refer to “imsimta qm counters,” on page 520.

You can also monitor the number of files in the queue directories (`/`*ServeRoot*`/msg-`*instance*`/imta/queue/`). The number of files will be site-specific, and you’ll need to build a baseline history to find out what is “too many.” This can be done by recording the size of the queue files over a two week period to get an approximate average.

Monitoring Rate of Delivery Failure

A delivery failure is a failed attempt to deliver a message to an external site. A large increase in rate of delivery failure can be a sign of a network problem such as a dead DNS server or a remote server timing out on responding to connections.

Symptoms of Rate of Delivery Failure

There are no outward symptoms. Lots of `Q` records will appear in to `mail.log_current`.

To Monitor the Rate of Delivery Failure

Delivery failures are recorded in the MTA logs with the logging entry code `Q`. Look at the Q record in the file `msg-`*instance*`/log/imta/mail.log_current`

Monitoring Inbound SMTP Connections

An unusual increase in the number of inbound SMTP connections from a given IP address may indicate:

- An external user is trying to relay mail.
- An external user is trying to do a service denial attack.

Symptoms of Unauthorized SMTP Connections

- **External user relaying mail**: No outward symptoms.
- **Service denial attack**: External attempt to overload the SMTP servers with message requests.

To Monitor Inbound SMTP Connections

- **External user relaying mail**: Look in `msg-`*instance*`/log/imta/mail.log_current` for records with the logging entry code `J` (rejected relays). To turn on logging of remote IP addresses add the following line to the `option.dat` file:

  ```
  log_connection=1
  ```

  Note that there is a slight performance trade-off when this feature is enabled.

- **Service denial attack**: To find out who and how many users are connecting to the SMTP servers, you can run the command `netstat` and check for connections at the SMTP port (default: 25). Example:

  ```
  Local address        Remote
  address                      State

  192.18.79.44.25    192.18.78.44.56035    32768   0 32768     0
  CLOSE_WAIT
  192.18.79.44.25    192.18.136.54.57390    8760   0 24820     0
  ESTABLISHED
  192.18.79.44.25    192.18.26.165.48508   33580   0 24820     0
  TIME_WAIT
  ```

Note that you will first need to determine the appropriate number of SMTP connections and their states (`ESTABLISHED`, `CLOSE_WAIT`, etc.) for your system to determine if a particular reading is out of the ordinary.

If you find many connections staying in the `SYN_RECEIVED` state this might be caused by a broken network or a denial of service attack. In addition, the lifetime of an SMTP server process is limited. This is controlled by the MTA configuration variable `MAX_LIFE_TIME` in the `dispatcher.cnf` file. The default is 86,400 seconds (one day). Similarly, `MAX_LIFE_CONNS` specifies the maximum number of connections a server process can handle in its lifetime. If you find a particular SMTP server that has around for a long time you may wish to investigate.

Monitoring the Dispatcher and Job Controller Processes

The Dispatcher and Job Controller Processes must be operating for MTA to work. You should have one process of each kind.

Symptoms of Dispatcher and Job Controller Processes Down

If the Dispatcher is down or does not have enough resources, SMTP connections are refused.

If the Job Controller is down, queue size will grow.

To Monitor Dispatcher and Job Controller Processes

Check to see that the processes called `dispatcher` and `job_controller` exist. See “Check That the Job Controller and Dispatcher are Running,” on page 468.

Monitoring Message Access

This section consists of the following subsections:

Monitoring imapd, popd and httpd

These processes provide access to IMAP, POP and Webmail services. If any of these is not running or not responding, the service will not function appropriately. If the service is running, but is over loaded, monitoring will allow you to detect this and configure it more appropriately.

Symptoms of imapd, popd and httpd Problems

Connections are refused or system is too slow to connect. For example, if IMAP is not running and you try to connect to IMAP directly you will see something like this:

```
telnet 0 143
Trying 0.0.0.0...
telnet: Unable to connect to remote host: Connection refused
```

If you try to connect with a client, you will get a message such as:

```
netscape is unable to connect to the server at the location you have
specified. The server may be down or busy.
```

To Monitor imapd, popd and httpd

- Can be monitored with SNMP.

  If you have the SNMP set up, this is a very good way to monitor these processes. See Appendix A, “SNMP Support.” The server information is in the Network Services Monitoring MIB.

- Check log files.

  Look in the directory *msg-instance* / log / *service* where *service* can be http or IMAP or POP. In that directory you will find a number of log files. One filename is the name of the *service* (imap, pop, http) and the others are the name of the service plus a sequence number and a date concatenated to the service name. For example:

  ```
  imap  imap.29.1010221593 imap.31.1010394412  imap.33.1010567224
  ```

  The file with just the service name is the latest log. The other ones are ordered by the sequence number (here 29, 31, 33) and the one with the highest sequence number is the next newest one. (See Chapter 13, “Logging and Log Analysis.”)

If a server was shut down you might see something like this:

```
[05/Jan/2002:08:36:38 -0800] gotmail-a imapd[10275]: General
Warning: iPlanet Messaging Server IMAP4 5.2 (built Dec  9 2001)
shutting down
```

- Can be checked with `counterutil`. See "counterutil," on page 513 and the *iPlanet Messaging Server Reference Manual*.
- Run the platform-specific command to verify that the imapd, popd and httpd processes are running. For example, in Solaris you can use the `ps` command and look for `imapd`, `popd` and `mshttpd`. In Windows NT, you can either use the Task Manager window or the command line.
- You can set alarms for specified server performance thresholds by setting the server response configuration parameters described in "Recommended stored Parameters," on page 512.

Monitoring `stored`

`stored` performs a variety of important tasks such as deadlock and transaction operations of the message database, enforcing aging policies, and expunging and erasing messages stored on disk. If `stored` stops running, the messaging server will eventually run into problems. If `stored` doesn't start when `start-msg` is run, no other processes will start. For more information about `stored` see the *iPlanet Messaging Server Reference Manual*.

Symptoms of `stored` Problems

There are no outward symptoms.

To Monitor stored

- Check that the `stored` process is running. `stored` creates and updates a `pid` file in `msg-`*instance*`/config` called `pidfile.store`. The `pid` file shows an `init` state when recovering and a `ready` state when ready. For example:

  ```
  231: cat pidfile.store
  28250
  ready
  ```

  The number on the first line is the process ID of `stored`.

```
232: ps -eaf | grep stored
mailsrv 28250     1  0   Jan 05 ?        8:44
/usr/iplanet/server5/bin/msg/admin/bin/stored -d
```

- Check for log file build up in `msg-`*instance*`/store/mailboxlist`. Note that not every log file build up is caused by direct `stored` problems. Log files may also build up if `imapd` dies or there is a database problem.
- Check the timestamp on the following files in `msg-`*instance*`/config`:

  `stored.ckp` - Touched when attempt at checkpointing is made. Should get time stamped every 1 minute
  `stored.lcu` - Touched at every db log cleanup. Should get time stamped every 5 minutes
  `stored.per` - Touched at every spawn of peruser db writeout. Should get time stamped every 60 minutes
- Check for `stored` messages in the default log file `msg-`*instance*`/log/default/default`

Monitoring LDAP Directory Server

This section consists of the following subsection:

- "Monitoring slapd," on page 509

Monitoring `slapd`

The LDAP directory server (`slapd`) provides directory information for the messaging system. If `slapd` is down, the system will not work properly. If `slapd` response time is too slow, this will affect login speed and any other transaction that requires LDAP lookups.

Symptoms of `slapd` Problems

- Client POP, IMAP, or Webmail Authentication fails or slower than expected.
- MTA not working properly

To Monitor slapd

- Check that `ns-slapd` process is running.
- Check `slapd` log files `access` and `errors` in `slapd-`*instance*`/logs/`

- Check the `ns-slapd` response time while searching for a user.
- View the Admin Console to monitor slapd.

Monitoring the Message Store

Messages are stored in a database. The distribution of users on disks, the size of their mailbox, and disk requirements affect the store performance. This section consists of the following subsections:

- "Monitoring the State of Message Store Database Locks," on page 510
- "Monitoring the Number of Database Log Files in the mboxutil Directory," on page 510

Monitoring the State of Message Store Database Locks

The state of DB-locks is held by different server processes. These database locks can affect the performance of the message store. In case of deadlocks, messages will not be getting inserted into the store at reasonable speeds and the `ims-ms` channel queue will grow larger as a result. There are legitimate reasons for a queue to back up, so it is useful to have a history of the queue length in order to diagnose problems.

Symptoms of Message Store Database Lock Problems

Number of transactions are accumulating and not resolving.

To Monitor Message Store Database Locks

Use the command `counterutil -o db_lock`

Monitoring the Number of Database Log Files in the mboxutil Directory

Database log files refer to sleepycat transaction checkpointing log files (`msg-`*instance*`/store/mboxlist`). Log file build up is a symptom of database checkpointing not happening. Log file build up can also be due to `stored` problems.

Symptoms of Database Log File Problems

There should be 2 or 3 log files. If there are more, it is a sign of a potentially serious problem. The message store uses a few databases for messages and quotas, and a problem with those can lead to problems for all of the mail server.

To Monitor Database Log Files

Look in the `msg-`*instance*`/store/mboxlist` directory and make sure there are only 2 or 3 files.

Utilities and Tools for Monitoring

The following tools are available in for monitoring:

`stored`

The `stored` utility performs maintenance tasks on the server, but it also can do monitoring. It can periodically check the server state, disk space, service response times and, if specified, it can issue alarms in the form of email messages to the postmaster (see page 508).

An alarm comes in the form of an email message from `stored` to the postmaster warning of a specified condition. A sample email alarm sent by `stored` when a certain threshold is exceeded is shown below:

Subject: ALARM: server response time in seconds of "ldap_siroe.com_389" is 10
Date: Tue, 17 Jul 2001 16:37:08 -0700 (PDT)
From: postmaster@siroe.com
To: postmaster@siroe.com

Server instance: /usr/iplanet/server5/msg-europa
Alarmid: serverresponse
Instance: ldap_siroe_europa.com_389
Description: server response time in seconds
Current measured value (17/Jul/2001:16:37:08 -0700): 10
Lowest recorded value: 0
Highest recorded value: 10
Monitoring interval: 600 seconds
Alarm condition is when over threshold of 10
Number of times over threshold: 1

You can specify how often `stored` monitors disk and server performance, and under what circumstances it sends alarms. This is done by using the `configutil` command to set the alarm parameters. Table 15-1 shows useful `stored` parameters along with their default setting.

Table 15-1 Recommended `stored` Parameters

Parameter	Description (Default)
alarm.msgalarmnoticehost	(localhost) Machine to which you send warning messages.
alarm.msgalarmnoticeport	(25) The SMTP port to which to connect when sending alarm message.
alarm.msgalarmnoticercpt	(Postmaster@localhost) Whom to send alarm notice.
alarm.msgalarmnoticesender	(Postmaster@localhost) Address of sender the alarm.
alarm.diskavail.msgalarmdescription	Description for disk availability alarm.
alarm.diskavail.msgalarmstatinterval	(3600) Interval in seconds between disk availability checks. Set to 0 to disable checking of disk usage.
alarm.diskavail.msgalarmthreshold	(10) Percentage of disk space availability below which an alarm is sent.
alarm.diskavail.msgalarmthresholddirection	(-1) Specifies whether the alarm is issued when disk space availability goes below threshold (-1) or above it (1).
alarm.diskavail.msgalarmwarninginterval	(24). Interval in hours between subsequent repetition of disk availability alarms.
alarm.serverresponse.msgalarmdescription	Description for servers response alarm.
alarm.serverresponse.msgalarmstatinterval	(600) Interval in seconds between server response checks. Set to 0 to disable checking of server response.

Table 15-1 Recommended `stored` Parameters *(Continued)*

Parameter	Description (Default)
alarm.serverresponse.msgalarmthreshold	(10) If server response time in seconds exceeds this value, alarm issued.
alarm.serverresponse.msgalarmthresholddirection	(1) Specifies whether alarm is issued when server response time is greater that (1) or less than (-1) the threshold.
alarm.serverresponse.msgalarmwarninginterval	(24) Interval in hours between subsequent repetition of server response alarm.

counterutil

This utility provides statistics acquired from different system counters. Here is a current list of available counter objects:

```
counterutil -l
entry = alarm
entry = diskusage
entry = serverresponse
entry = db_lock
entry = db_log
entry = db_mpool
entry = db_txn
entry = popstat
entry = imapstat
entry = httpstat
entry = cgimsg
```

Each entry represents a counter object and supplies a variety of useful counts for this object. In this section we will only be discussing the `alarm`, `diskusage`, `serverresponse`, `db_lock`, `popstat`, `imapstat`, and `httpstat` counter objects. For details on `counterutil` command usage, refer to the *iPlanet Messaging Server Reference Manual*.

`counterutil` Output

`counterutil` has a variety of flags. A command format for this utility may be as follows:

```
counterutil -o CounterObject -i 5 -n 10
```

where,

`-o` *CounterObject* represents the counter object `alarm`, `diskusage`, `serverresponse`, `db_lock`, `popstat`, `imapstat`, and `httpstat`.

`-i 5` specifies a 5 second interval.

`-n 10` represents the number of iterations (default: infinity).

An example of `counterutil` usage is as follows:

```
counterutil -o imapstat -i 5 -n 10
Monitor counteroobject (imapstat)
registry /gotmail/iplanet/server5/msg-gotmail/counter/counter
opened
counterobject imapstat opened

count = 1 at 972082466 rh = 0xc0990 oh = 0xc0968

global.currentStartTime [4 bytes]: 17/Oct/2000:12:44:23 -0700
global.lastConnectionTime [4 bytes]: 20/Oct/2000:15:53:37 -0700
global.maxConnections [4 bytes]: 69
global.numConnections [4 bytes]: 12480
global.numCurrentConnections [4 bytes]: 48
global.numFailedConnections [4 bytes]: 0
global.numFailedLogins [4 bytes]: 15
global.numGoodLogins [4 bytes]: 10446
  ...
```

Alarm Statistics Using `counterutil`

These alarm statistics refer to the alarms sent by `stored`.The alarm counter provides the following statistics:

Table 15-2 `counterutil alarm` Statistics

Suffix	Description
`alarm.countoverthreshold`	Number of times crossing threshold.
`alarm.countwarningsent`	Number of warnings sent.
`alarm.current`	Current monitored valued.
`alarm.high`	Highest ever recorded value.
`alarm.low`	Lowest ever recorded value.
`alarm.timelastset`	The last time current value was set.
`alarm.timelastwarning`	The last time warning was sent.
`alarm.timereset`	The last time reset was performed.

Table 15-2 `counterutil alarm` Statistics

Suffix	Description
`alarm.timestatechanged`	The last time alarm state changed.
`alarm.warningstate`	Warning state (yes(1) or no(0)).

IMAP, POP, and HTTP Connection Statistics Using `counterutil`

To get information on the number of current IMAP, POP, and HTTP connections, number of failed logins, total connections from the start time, and so forth, you can use the command `counterutil -o` *CounterObject* `-i 5 -n 10`.where *CounterObject* represents the counter object `popstat`, `imapstat`, or `httpstat`. The meaning of the `imapstat` suffixes is shown in Table 15-3. The `popstat` and `httpstat` objects provide the same information in the same format and structure.

Table 15-3 `counterutil imapstat` Statistics

Suffix	Description
`currentStartTime`	Start time of the current IMAP server process.
`lastConnectionTime`	The last time a new client was accepted.
`maxConnections`	Maximum number of concurrent connections handled by IMAP server.
`numConnections`	Total number of connections served by the current IMAP server.
`numCurrentConnections`	Current number of active connections.
`numFailedConnections`	Number of failed connections served by the current IMAP server.
`numFailedLogins`	Number of failed logins served by the current IMAP server.
`numGoodLogins`	Number of successful logins served by the current IMAP server.

Disk Usage Statistics Using `counterutil`

The command: `counterutil -o diskusage` generates following information:

Table 15-4 `counterutil diskstat` Statistics

Suffix	Description
`diskusage.availSpace`	Total space available in the disk partition.

Table 15-4 `counterutil diskstat` Statistics

Suffix	Description
`diskusage.lastStatTime`	The last time statistic was taken.
`diskusage.mailPartitionPath`	Mail partition path.
`diskusage.percentAvail`	Disk partition space available percentage.
`diskusage.totalSpace`	Total space in the disk partition.

Server Response Statistics

The command: `counterutil -o serverresponse` generates following information. This information is useful for checking if the servers are running, and how quickly they're responding.

Table 15-5 `counterutil serverresponse` Statistics

Suffix	Description
`http.laststattime`	Last time http server response was checked.
`http.responsetime`	Response time for the http.
`imap.laststattime`	Last time imap server response was checked.
`imap.responsetime`	Response time for the imap.
`pop.laststattime`	Last time pop server response was checked.
`pop.responsetime`	Response time for the pop.
`ldap_host1_389.laststattime`	Last time ldap_host1_389 server response was checked.
`ldap_host1_389.responsetime`	Response time for the ldap_host1_389.
`ugldap_host2_389.laststattime`	Last time ugldap_host2_389 server response was checked.
`ugldap_host2_389.responsetime`	Response time for the ugldap_host2_389.

Log Files

Messaging server logs event records for SMTP, IMAP, POP, and HTTP. The policies for creating and managing the Messaging Server log files are customizable.

Since logging can affect the server performance, logging should be considered very carefully before the burden is put on the server. Refer to Chapter 13, "Logging and Log Analysis" for more information.

imsimta counters

The MTA accumulates message traffic counters based upon the Mail Monitoring MIB, RFC 1566 for each of its active channels. The channel counters are intended to help indicate the trend and health of your e-mail system. Channel counters are not designed to provide an accurate accounting of message traffic. For precise accounting, instead see MTA logging as discussed in Chapter 13, “Logging and Log Analysis.

The MTA channel counters are implemented using the lightest weight mechanisms available so that they cause as little impact as possible on actual operation. Channel counters do not try harder: if an attempt to map the section fails, no information is recorded; if one of the locks in the section cannot be obtained almost immediately, no information is recorded; when a system is shut down, the information contained in the in-memory section is lost forever.

The `imsimta counters -show` command provides MTA channel message statistics (see below). These counters need to be examined over time noting the minimum values seen. The minimums may actually be negative for some channels. A negative value means that there were messages queued for a channel at the time that its counters were zeroed (for example, the cluster-wide database of counters created). When those messages were dequeued, the associated counters for the channel were decremented and therefore leading to a negative minimum. For such a counter, the correct “absolute” value is the current value less the minimum value that counter has ever held since being initialized.

```
Channel          Messages    Recipients    Blocks
-------          --------    ----------    -------
tcp_local
   Received       29379       79714      982252                        (1)
   Stored            61         113       -2004                        (2)
   Delivered      29369       79723      983903 (29369 first time)     (3)
   Submitted      13698       13699       18261                        (4)
   Attempted          0           0           0                        (5)
   Rejected           1          10           0                        (6)
   Failed           104         104        4681                        (7)

   Queue time/count        16425/29440 = 0.56                          (8)
   Queue first time/count  16425/29440 = 0.56                          (9)

   Total In Assocs           297637
   Total Out Assocs           28306
```

1) `Received` is the number of messages enqueued to the channel named `tcp_local`. That is, the messages enqueued (E records in the `mail.log*` file) to the `tcp_local` channel by any other channel.

2) `Stored` is the number of messages stored in the channel queue to be delivered.

3) `Delivered` is the number of messages which have been processed (dequeued) by the channel `tcp_local`. (That is, D records in the `mail.log*` file.) A dequeue operation may either correspond to a successful delivery (that is, an enqueue to another channel), or to a dequeue due to the message being returned to the sender. This will generally correspond to the number `Received` minus the number `Stored`.

The MTA also keeps track of how many of the messages were dequeued upon first attempt; this number is shown in parentheses.

4) `Submitted` is the number of messages enqueued (E records in the `mail.log` file) by the channel `tcp_local` to any other channel.

5) `Attempted` is the number of messages which have experienced temporary problems in dequeuing, that is, Q or Z records in the `mail.log*` file.

6) `Rejected` is the number of attempted enqueues which have been rejected, that is, `J` records in the `mail.log*` file.

7) `Failed` is the number of attempted dequeues which have failed, that is, `R` records in the `mail.log*` file.

8) `Queue time/count` is the average time-spent-in-queue for the delivered messages. This includes both the messages delivered upon the first attempt, see (9), and the messages that required additional delivery attempts (hence typically spentnoticeable time waiting fallow in the queue).

9) `Queue first time/count` is the average time-spent-in-queue for the messages delivered upon the first attempt.

Note that the number of messages submitted can be greater than the number delivered. This is often the case, since each message the channel dequeues (delivers) will result in at least one new message enqueued (submitted) but possibly more than one. For example, if a message has two recipients reached via different channels, then two enqueues will be required. Or if a message bounces, a copy will go back to the sender and another copy may be sent to the postmaster. Usually that will be two submissions (unless both are reached through the same channel).

More generally, the connection between `Submitted` and `Delivered` varies according to type of channel. For example, in the conversion channel, a message would be enqueued by some other arbitrary channel, and then the conversion channel would process that message and enqueue it to a third channel and mark the message as dequeued from its own queue. Each individual message takes a path:

```
elsewhere -> conversion   E record   Received
conversion -> elsewhere   E record   Submitted
conversion                   D record   Delivered
```

However, for a channel such as `tcp_local` which is not a "pass through," but rather has two separate pieces (slave and master), there is no connection between `Submitted` and `Delivered`. The `Submitted` counter has to do with the SMTP server portion of the `tcp_local` channel, whereas the `Delivered` counter has to do with the SMTP client portion of the `tcp_local` channel. Those are two completely separate programs, and the messages travelling through them may be completely separate.

Messages submitted to the SMTP server:

```
tcp_local -> elsewhere  E record    Submitted
```

Messages sent out to other SMTP hosts via the SMTP client:

```
elsewhere -> tcp_local  E record    Received
tcp_local                 D record    Delivered
```

Channel dequeues (delivers) will result in at least one new message enqueued (submitted) but possibly more than one. For example, if a message has two recipients reached via different channels, then two enqueues will be required. Or if a message bounces, a copy will go back to the sender and another copy may be sent to the postmaster. Usually that will be reached through the same channel.

Implementation on UNIX and NT

For performance reasons, a node running the MTA keeps a cache of channel counters in memory using a shared memory section (UNIX) or shared file-mapping object (NT). As processes on the node enqueue and dequeue messages, they update the counters in this in-memory cache. If the in-memory section does not exist when a channel runs, the section will be created automatically. (The `imta start` command also creates the in-memory section, if it does not exist.)

The command `imta counters -clear` or the `imta qm` command `counters clear` may be used to reset the counters to zero.

imsimta qm counters

The `imsimta qm counters` utility displays MTA channel queue message counters. You must be root or `mailsrv` to run this utility. The output fields are the same as those described in “imsimta counters,” on page 517. See also *iPlanet Messaging Server Reference Manual* for usage details.

Example 1:

```
imsimta qm counters show
```

```
Channel                 Messages    Recipients     Blocks
----------------------  ----------  ----------    ----------
autoreply
   Received              13077        13859         264616
   Stored                   92           91           -362
   Delivered             12985        13768         264978
   Submitted              2594         2594           3641
...
```

Example 2:

```
imsimta qm counters today
```

```
4370 messages processed so far today
Your license permits an unlimited number of messages per day.
```

MTA Monitoring Using SNMP

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 this product), you can monitor certain parts of the iPlanet Messaging Server. Refer to Appendix A, “SNMP Support” for details.

mboxutil for Mailbox Quota Checking

You can monitor mailbox 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
------------------------------------------------------------------------
Domain red.siroe.com (diskquota = not set msgquota = not set) quota usage
------------------------------------------------------------------------
diskquota        size(K)    %use    msgquota      msgs    %use    user
# of domains = 1
# of users = 705

no quota         50418              no quota      4392            ajonkish
no quota         5                  no quota      2               andrewt
no quota         355518             no quota      2500            aniksri
...
```

The following example shows the quota usage for user sorook:

```
% mboxutil -u sorook
------------------------------------------------------------------------
quota usage for user sorook
------------------------------------------------------------------------
diskquota      size(K)    %use    msgquota      msgs     %use    user

no quota       1487                no quota      305              sorook
```

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. For more information on monitoring the iPlanet Messaging Server refer to Chapter 15, “Monitoring 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

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.

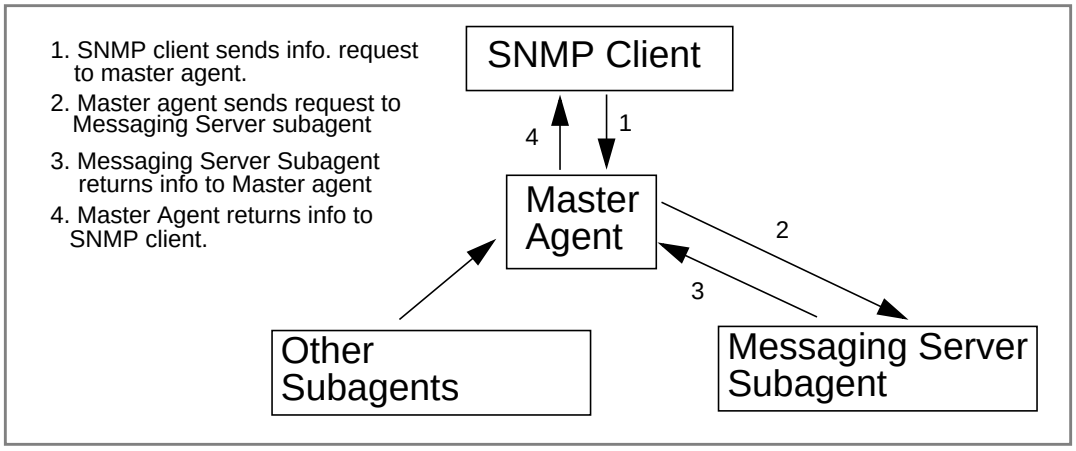
Limitations of the Messaging Server SNMP support are as follows:

- 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
```

Configuring SNMP Support for Windows Platforms

Although the overhead imposed by the SNMP monitoring is very small, Messaging Server nonetheless ships with SNMP support disabled. To enable the SNMP support, from a DOS prompt issue the following commands

```
X:\> server_root\msg-instance\configutil /o local.snmp.enable /v 1
X:\> %SYSTEMROOT%\SYSTEM32\regsvr32.exe server_root\bin\msg\imta\bin\madmand.dll
```

Then, restart the SNMP Service via the Windows Services utility. The Services Utility is sometimes referred to as the *Microsoft Management Console.*

Please note that the Windows SNMP Service must already be running in order for Messaging Server SNMP support to operate. By default, the Windows SNMP Service is not installed with Windows NT. You must manually install the Windows SNMP Service.

On Windows NT, install the SNMP Service as follows:

1. From the **Control Panel**, right click on the **Network** icon.
2. From the **Network** window, select the **Services** tab.
3. From the **Services** dialog, click on the **Add...** button.

4. From the popup **Select Network Service** window, select the **SNMP Service** in the **Network Service** list box. Then click the **OK** button.

5. Windows will now install the SNMP Service. You may need to supply your Windows NT CDROM to complete the installation.

For further information on installing the SNMP Service, consult the Windows documentation from Microsoft.

To disable Messaging Server SNMP support, issue the commands:

```
X:\> server_root\msg-instance\configutil /o local.snmp.enable /v 0
X:\> %SYSTEMROOT%\SYSTEM32\regsvr32.exe /u server_root\bin\msg\imta\bin\madmand.dll
```

Then restart the SNMP Service from the Windows Services utility.

On Windows platforms, the `start-msg snmp` and `stop-msg snmp` commands have no effect. Messaging Server SNMP support runs within the Windows SNMP Service and can only be started or stopped by starting or stopping the Windows SNMP Service.

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).

MTA Direct LDAP Operation

Prior to the iPlanet Messaging Server 5.2 release, directory information about users and groups used by the MTA was accessed through a number of files and databases. Data in these files and databases was updated by the `dirsync` process which monitored changes to the directory and updated the file and database data accordingly. This remains the default behavior in version 5.2, however there is a new option that allows the MTA to interact directly with the directory. This option is called the *direct LDAP mode*.

When the MTA is configured to operate in direct LDAP mode, the `dirsync` process and its databases are not used. Instead the MTA makes equivalent LDAP calls, first to determine whether a domain is hosted on the MTA and then to access the desired delivery information. There is almost no net effect on address translation when changing from the `dirsync` mode of operation to the direct LDAP mode, except that the direct LDAP mode is configurable and the mechanism more transparent. There is, however, a change in the way hosted domains work, and there is also an impact on the way the system performs. See “Implications of Changing to Direct LDAP Mode,” on page 561 for details.

This chapter consists of the following sections:

To Enable Direct LDAP Mode

To enable direct LDAP mode, make the following changes to the standard MTA configuration:

1. Add the line to the rewrite section in the file `.../imta/config/imta.cnf`

```
$*      $E$F$U%$H$V$H@localhost
```

where `localhost` is the primary host name of the MTA.

For instance if the MTA is called `island.siroe.com`, you would modify the start of the rules section of `.../imta/config/imta.cnf` to read something like

```
! Rules to select   local users
$*                  $E$F$U%$H$V$H@island.siroe.com
island.siroe.com    $U%$D@island.siroe.com
siroe.com           $U%$D@island.siroe.com
```

2. Change the definition of the `ims-ms` channel in `.../imta/config/imta.cnf` to remove the clause `filter ssrd:$A`.

   If your `ims-ms` channel definition reads

   ```
   ! ims-ms
   ims-ms defragment subdirs 20 notices 1 7 14 21 28  \
    backoff "pt5m" "pt10m" "pt30m" "pt1h" "pt2h" "pt4h" \
    maxjobs 1 pool IMS_POOL fileinto $U+$S@$D filter ssrd:$A
   ims-ms-daemon
   ```

   it should be changed to read

   ```
   ! ims-ms
   ims-ms defragment subdirs 20 notices 1 7 14 21 28 \
    backoff "pt5m" "pt10m" "pt30m" "pt1h" "pt2h" "pt4h" \
    maxjobs 1 pool IMS_POOL fileinto $U+$S@$D
   ims-ms-daemon
   ```

3. Add the following lines to the file `.../imta/config/option.dat`:

   ```
   ALIAS_MAGIC=8764
   ALIAS_URL0=ldap:///$V?*?sub?$R
   USE_REVERSE_DATABASE=4
   REVERSE_URL=ldap:///$V?mail?sub?$Q
   USE_DOMAIN_DATABASE=0
   ```

   If vanity domains are to be supported the following additional options must also be set:

   ```
   DOMAIN_MATCH_URL=ldap:///$B?msgVanityDomain?sub? \
   (msgVanityDomain=$D)
   ALIAS_URL1=ldap:///$B?*?sub?(&(msgVanityDomain=$D)$R)
   ALIAS_URL2=ldap:///$1V?*?sub?(mailAlternateAddress=@$D)
   ```

4. Remove the following lines from `.../imta/config/job_controller.cnf`.

```
[PERIODIC_JOB=dirsync_incr]
command=IMTA_TABLE:../../imsimta dirsync
time=/00:10
!
[PERIODIC_JOB=dirsync_full]
command=IMTA_TABLE:../../imsimta dirsync -F
time=02:00/24:00
!
```

5. Remove the following lines from the `SEND_ACCESS` mapping at the end of the file `.../imta/config/mappings`

```
*|*|inactive|*  $X4.2.1|$NMailbox$ temporarily$ disabled
*|*|deleted|*   $X5.1.6|$NRecipient$ no$ longer$ on$ server
```

6. Delete, or at least move these MTA databases:

```
.../imta/db/aliasesdb.db
.../imta/db/domaindb.db
.../imta/db/reversedb.db
```

7. Compiled the modified MTA configuration. This must happen before it comes into effect.

How Direct LDAP Mode Works

MTA processing of destination email addresses is fundamentally unchanged. Briefly, the process is as follows: First, the MTA uses the rewrite rules to 1) determine if a domain is recognized, 2) rewrite the address as appropriate, and 3) route the message to the appropriate channel. If the message is routed to the `l` channel, then the address is transformed using the alias lookup process (see “Direct LDAP Alias Resolution,” on page 544), and the resulting address or addresses are rewritten again using the rewrite rules to route these addresses to the channel associated with the aliases. Typically this is the `ims-ms` channel, `auto_reply` or one of the other standard MTA channels.

The direct LDAP mode of operation changes the rewrite rule phase and the alias phase of the address processing. These changes are described in the following subsections.

• “Reverse Address Translation,” on page 560

Resolving Addresses Using the Direct LDAP Rewrite Rule ($V)

The MTA resolves addresses by first checking the domain part of the address (the part to the right of the @) against the rewrite rules. Rewrite rules are found in the first half of the `.../imta/config/imta.cnf` file. If a match is found, the rules specify to which channel the mail is to be routed. For instance mail is routed to `tcp_local` for outbound internet traffic, or the local channel, `l`, for users provisioned in the directory.

When the MTA is configured in `dirsync` mode, the rule evaluation process uses information from the domain database, which is one of those databases maintained by the `dirsync` process. When the MTA is configured in direct LDAP mode, a special “try me first” rewrite rule is used. This rule looks like this:

```
$*      $E$F$U%$H$V$H@localhost
```

The left hand side of this rule, the `$*` pattern, means try this rule first, and try it on all addresses. The right hand side says:

- `$E` - use on envelope addresses only.
- `$F` - use on forward pointing (`To:`) addresses only.
- `$U%$H` - “Rewrite” an address to the form `user@host`. (Actually the rules specify that the unmodified original address be used.)
- `$V$H` - Only match this rule if the host part of the address, the part of the address to the right of the `@` sign, matches a domain defined in the directory.
- `@localhost` - route to the `l` channel.

How the LDAP Domain Lookup Works

The new part of the rewrite rule process is the `$V` matching parameter. `$V` is used to determine if an address is local, and if so, to find its location in the directory tree. `$V` takes a parameter, in this case `$H`, the host part of the address. The `$V` tag brings into play a number of LDAP lookups. The process involves looking up the domain part of the address in the DC tree to find the appropriate subtree of the user and group tree. For instance if the address under consideration is

```
robinson.crusoe@desert.island.siroe.com
```

the MTA first checks for the domain `desert.island.siroe.com`, then, if that fails, for `island.siroe.com`, `siroe.com`, and `com`. This LDAP lookup takes place in the DC tree in the directory (a detailed description of the iPlanet Messaging Server namespace and DIT structures is described in the *iPlanet Messaging Server Provisioning Guide*). This tree is rooted at the location specified by the `service.dcroot` configutil attribute, default value `o=internet`. The lookups will be for entries with distinguished names of

```
dc=desert,dc=island,dc=siroe,dc=com,o=internet
dc=island,dc=siroe,dc=com,o=internet
dc=siroe,dc=com,o=internet
dc=com,o=internet
```

A domain lookup is only considered a success if the entry found has either an object class of `inetDomain` and an attribute of `inetDomainBaseDn`, or an object class of `inetDomainAlias` and an attribute of `aliasedObjectName`.

If you want to prevent the checking of upper level domains, in the example this would be `island.siroe.com`, `siroe.com`, and `com`, you can do so by clearing the least significant bit of the option `DOMAIN_UPLEVEL`. `DOMAIN_UPLEVEL` is specified in `.../imta/config/option.dat`. Its default value is 1, so to prevent the up level checks add the line

```
DOMAIN_UPLEVEL=0
```

to `.../imta/config/option.dat`.

There is another new tag, `$Z`, which has exactly the opposite meaning to `$V`. `$V` makes a rule match if the host is found in the directory, `$Z` makes a rule match if the host is not found in the directory.

Vanity Domain Lookup

If you have any vanity domains (NOT hosted domains) defined for any users, you need to enable the LDAP check for these too. The check for vanity domains is disabled by default. To enable it, add the following line to `.../imta/config/option.dat`:

```
DOMAIN_MATCH_URL=ldap:///$B?msgVanityDomain?sub? \
  (msgVanityDomain=$D)
```

The check for vanity domains only takes place if the checks for hosted domains fail.

Domain Lookup Cache

Checking the directory for all domains is a potentially expensive operation, as it has to be performed for all domains, including any internet domains to which anyone sends mail. To reduce the cost, the results of the lookups are cached by the MTA. By default the results of up to 100000 lookups (successful or otherwise) are cached for up to 600 seconds. This caching can be controlled setting by the following options in `.../imta/config/option.dat`

```
DOMAIN_MATCH_CACHE_SIZE=100000
DOMAIN_MATCH_CACHE_TIMEOUT=600
```

Managing LDAP Errors During Address Rewrite

There are four possible outcomes to a domain lookup in the directory:

- The domain is found and is good
- The domain is found and is not good
- The domain is not found
- The lookup fails (an LDAP error)

The first case presents no problems. The second and third are treated as equivalent, and cause the `$V` rule to fail. The last case is more difficult. There are two reasonable courses of action the MTA could take in this case:

1. Reject the address with a *400 Temporary lookup failur*e SMTP response.
2. Redirect the mail to the `reprocess` channel for processing later.

The first action is the obvious and correct action if the mail is coming from some remote MTA. The second action is more appropriate if the mail is coming from a user agent submitting mail. The MTA needs to tell the difference between these two cases and act accordingly. The mechanism to enable this is the MTA option `DOMAIN_FAILURE`. `DOMAIN_FAILURE` specifies a string that is used to replace the unused part of a rewrite rule in the case of a domain lookup failure. Thus, if `DOMAIN_FAILURE` has its default value of

```
DOMAIN_FAILURE=reprocess-daemon$Mtcp_local$1M$1~-error$4000000?Temp
orary lookup failure
```

and the rewrite rule being processed is the standard

```
$*      $E$F$U%$H$V$H@localhost
```

and the domain lookup caused by the VH phrase fails, then processing continues as if the rewrite rule had been

```
$*      $E$F$U%$H$V$H@reprocess-daemon$Mtcp_local$1M$1~-error$4000000?Temporary lookup failure
```

The processing of this resulting rule is as follows:

- $E - use on envelope addresses only.
- $F - use on forward pointing (To:) addresses only.
- $U%$H - "Rewrite" an address to the form user@host. (Actually the rules specify that the unmodified original address be used.)
- VH - Only match this rule if the host part of the address—the part of the address to the right of the @ sign—matches a domain defined in the directory. This encounters an LDAP error, thus generating the modified rule.
- @reprocess-daemon - route to the reprocess channel.
- $Mtcp_local - "fail" if the source channel is not tcp_local. This failure is outcome of processing so far. Processing of the rule continues.
- ($1M) - "fail" if the channel is not an internal reprocessing channel such as reprocess or conversion.
- $~ - stop processing with a successful match if the rule is currently failing.
- -error - change the destination channel to the invalid channel reprocess-daemon-error
- $4000000?Temporary lookup failure - set the SMTP extended error code to 4.0.0 and the error text to "Temporary lookup failure."

Thus, if the source channel is tcp_local (in all likelihood a connection from some remote MTA) the rewrite rule succeeds, but no channel exists for reprocess-daemon-error, so the address is rejected, and rejected with the 400 error code specified in the rule.

If the source channel is tcp_intranet (probably a user agent), the rule succeeds routing the message to the reprocess channel.

The DOMAIN_FAILURE option and the effective rewrite rule constructed from it uses some new rewrite tags.

$1M is similar to the existing $Mchannel tag in that it causes a rule to fail if the source channel is a reprocessing channel. It is more or less equivalent to $Mreprocess$Mprocess$Mdefragment$conversion.

$~ causes the channel matching checks specified in any $M or $1M (or $M or $1N) tags to be performed, and if the result is a failure, to terminate processing immediately with a success.

$abbbccc?text specifies the error code and error text to be used in the event of a failure. The error code is actually three decimal numbers a, bbb, ccc, and generates an extended SMTP result code of a.bbb.ccc.

Direct LDAP Alias Resolution

The object of alias resolution is to take a message's incoming address (alias) and generate an email address used to deliver the message to a channel. This address is called the *delivery address* and typically has the form *uid@channel_name*.

Rewrite rules look only at the part of the address to the right of the @ sign. Alias resolution, however, potentially looks at the whole of the address. The mechanisms used for address resolution is controlled by the option ALIAS_MAGIC in .../imta/config/option.dat. The default behavior is to check for a match in the aliases file and then in the aliases database, which is maintained by the dirsync process. (See "Aliases," on page 141.)

To enable direct LDAP operation, add the following line to .../imta/config/option.dat.

```
ALIAS_MAGIC=8764
```

This causes alias resolution to be attempted using the aliases file (typically only used for the site postmaster), then through the LDAP directory. LDAP alias resolution goes through a number of steps before generating a delivery channel. These are as follows:

1. Find the address's user / group entry in the LDAP directory.
2. Determine the entry type (user or group).
3. Extract the entry status (example, active, inactive, deleted, hold).
4. Extract the uid attribute.
5. Find the location of the user.
6. Verify that the size of the message does not exceed specified limits.
7. Generate delivery address based on the mailDeliveryOption attribute (example, mailbox, autoreply, program, and forward).

The remainder of this section describes each of these steps in detail.

Finding the User/Group Entry in the LDAP Directory

The LDAP queries to find the user / group entry of the alias address are defined by the URLs specified by the following options in `.../imta/config/option.dat`:

```
ALIAS_URL0
ALIAS_URL1
ALIAS_URL2
```

Unless you support vanity domains, only `ALIAS_URL0` is used. The recommended setting for this option is

```
ALIAS_URL0=ldap:///$V?*?sub?$R
```

The processing for the `$V` tag is similar to the processing for the `$V` tag described “How the LDAP Domain Lookup Works,” on page 540. If the lookup of the domain part of the address succeeds, the `$V` in the URL is replaced by the DN pointed to by the `inetDomainBaseDn` or `aliasedObjectName` attribute in the entry found. If the lookup fails, the alias expansion fails. (There is a variant of the `$V` tag available, `$1V`, that, if the lookup fails, returns the DN of the top of the user and group tree—the value of `local.ugldapbasedn`.)

`$R` is replaced by a filter appropriate for the schema in use as defined by the `configutil` parameter `local.imta.schematag`. The possible schema values and the attributes to be searched for a matching email address are as follows:

```
ims50    mail,mailalternateAddress,mailEquivalentAddress
nms41    mail,mailalternateAddress
sims40   mail,rfc822mailalias
```

`local.imta.schematag` can specify more than one of these values, comma separated. If more than one schema is specified, the union of the attributes are searched for a match. If your directory schema does not exactly match any of these schemata, you can override the list of attributes to be searched by specifying the `configutil` parameter `local.imta.mailaliases`. For example:

```
local.imta.mailaliases=mail,mailAlternateAddresses,email
```

would cause a search for a match on the attributes `mail`, `mailAlternateAddresses`, and `email`.

By default, the filter generated by the `$R` tag only searches for the address given. However you might want implied aliases in higher level domains. Thus, although you have provisioned `robinson.crusoe` in the `desert.island.siroe.com` domain, you might want to match for his user name in all domains in the domain tree. Thus, if the domain that matched in the evaluation of the rewrite rules was `siroe.com`, then the addresses searched for in the directory would be

```
robinson.crusoe@desert.island.siroe.com
robinson.crusoe@island.siroe.com
robinson.crusoe@siroe.com
```

To get this behavior, you have to set the next to least significant bit of the option DOMAIN_UPLEVEL, for instance by adding the line to the file .../imta/config/option.dat:

```
DOMAIN_UPLEVEL=3
```

Domain Lookup in Nonstandard Directories

If you cannot use the standard iPlanet directory structure with its DC tree separate from the user and group tree, another mechanism is available for finding the base of the tree in which to search for aliases. Instead of using $V in the ALIAS_URL0 as described above, you can invoke a mapping. The syntax for achieving this is to place in the URL instead of $V the following:

$|/*mapping-name*/*mapping-argument*|

The | initiates and terminates the callout. The character immediately following the $| is the separator between the mapping name and argument; a character should be chosen that doesn't collide with the expected character values used in either the mapping name or argument. *mapping name* is the name of the *domain lookup* mapping table. *mapping-argument* is the name of a domain, for example, $D becomes the name of the domain.

Domain Lookup for Vanity Domain Aliases

To support vanity domain aliases, the following additional URLs must be defined in .../imta/config/option.dat

```
ALIAS_URL1=ldap:///$B?*?sub?(&(msgVanityDomain=$D)$R)
ALIAS_URL2=ldap:///$1V?*?sub?(mailAlternateAddress=@$D)
```

LDAP Failures During Alias Resolution

The result of an alias lookup in the directory can return zero, one, or several results. If more than one entry matches, the lookup is considered a failure, just as if no results had been returned, and the address will be rejected as invalid. If, for any reason, none of the configured directory servers can be reached, or if an LDAP query results in an error, the address will be rejected with a temporary failure indication (4xx error in SMTP). The sending MTA should retry the mail later, by which time the problem with the directory may have been resolved.

Determining Entry Type

Once an entry has been found in the directory, it can be processed and mail can be delivered to the appropriate channel. The first step in processing the entry is to determine whether it represents a user, group, or something unrecognizable. If we discover that the entry is a user or a group, processing continues as appropriate. If the entry is neither a user nor a group, then the entry, and therefore the address being processed, is quietly ignored.

Entry type is determined by looking at the object classes to which the entry belongs. The required object classes for users and groups is implied by the schema in use for the directory, as defined by the `local.imta.schematag` setting. The object classes that have to be present to define an entry as a user or group for the different schemata are:

```
ims50:       users:     inetLocalMailRecipient + inetmailuser
             groups:    inetLocalMailRecipient + inetmailgroup

nms41:       users:     mailRecipient + nsMessagingServerUser
             groups:    mailGroup

sims40:      users:     inetMailRouting + inetmailuser
             groups:    netMailRouting + inetmailgroup
```

If your directory schema does not exactly match any of these schemata, you can define you own determinant for discriminating between user and group directory entries. The MTA options `LDAP_USER_OBJECT_CLASSES` and `LDAP_GROUP_OBJECT_CLASSES` can be used to specify the object classes that must be present for an entry to be classified as a user or group respectively. For instance, adding the following lines to `.../imta/config/option.dat`

```
LDAP_USER_OBJECT_CLASSES=inetLocalMailRecipient+inetmailUser,mailRe
cipient+nsMessagingServerUser

LDAP_GROUP_OBJECT_CLASSES=inetLocalMailRecipient+inetmailgroup,mail
Group
```

is equivalent to setting `local.imta.schematag=ims50,nms41`, in that an entry will be determined to be a user if it has the object classes `inetLocalMailRecipient` and `inetmailUser`, or the object classes `mailRecipient` and `nsMessagingServerUser`.

Extracting Attributes Used to Build the Delivery Address

Once the address entry type is determined, the MTA needs to extract a set of attributes from the domain and user or group entries to build the delivery address and deliver the message. Some or all of the following attributes (see Table B-1, Table B-2, and Table B-3) from the domain and user or group entries are extracted.

The following tables give the required default attribute names used, and the MTA option that can be used to chose different attribute names. Typically these options will not be set as the default values correspond to the standard schema. However, should your directory use a different attribute name for one or more of these attributes, they can be changed by setting the appropriate option in `.../imta/config/option.dat`

Table B-1 Default Domain Attributes and Override Options

LDAP attribute name	MTA override option
domainUidSeparator	LDAP_DOMAIN_ATTR_UID_SEPARATOR
mailDomainCatchallAddress	LDAP_DOMAIN_ATTR_CATCHALL_ADDRESS
mailDomainConversionTag	LDAP_DOMAIN_ATTR_CONVERSION_TAG
mailDomainMsgMaxBlocks	LDAP_DOMAIN_ATTR_BLOCKLIMIT
mailDomainReportAddress	LDAP_DOMAIN_ATTR_REPORT_ADDRESS
mailDomainSieveRuleSource	LDAP_DOMAIN_ATTR_FILTER
mailDomainStatus	LDAP_DOMAIN_ATTR_STATUS
mailRoutingHosts	LDAP_DOMAIN_ATTR_ROUTING_HOSTS
mailRoutingSmarthost	LDAP_DOMAIN_ATTR_SMARTHOST

Table B-2 Default User Attributes and Override Options

LDAP attribute name	MTA override option
mailConversionTag	LDAP_CONVERSION_TAG
mailDeliveryFileURL	LDAP_PROGRAM_INFO
mailDeliveryOption	LDAP_DELIVERY_OPTION
mailhost	LDAP_MAILHOST
mailMsgMaxBlocks	LDAP_BLOCKLIMIT
mailMsgQuota	LDAP_MESSAGE_QUOTA
mailProgramDeliveryInfo	LDAP_PROGRAM_INFO
mailQuota	LDAP_DISK_QUOTA
mailRoutingAddress	LDAP_ROUTING_ADDRESS
mailSieveRuleSource	LDAP_FILTER

Table B-2 Default User Attributes and Override Options

LDAP attribute name	MTA override option
UID	LDAP_UID
	LDAP_SPARE_1*
	LDAP_SPARE_2*

* The two spare LDAP options may be important as they can be used to substitute into delivery options patterns. This is described later in Processing Delivery Options.

Table B-3 Default Group Attributes and Override Options

LDAP attribute name	MTA override option
mailRejectText	LDAP_REJECT_TEXT
memberURL	LDAP_GROUP_URL2
mgrpAddHeader	LDAP_ADD_HEADER
mgrpAllowedBroadcaster	LDAP_AUTH_URL
mgrpAllowedDomain	LDAP_AUTH_DOMAIN
mgrpAuthPassword	LDAP_AUTH_PASSWORD
mgrpBroadcasterPolicy	LDAP_AUTH_POLICY
mgrpDeliverTo	LDAP_GROUP_URL1
mgrpDisallowBroadcaster	LDAP_CANT_URL
mgrpDisallowDomain	LDAP_CANT_DOMAIN
mgrpErrorsTo	LDAP_ERRORS_TO
mgrpMsgMaxSize	LDAP_ATTR_MAXIMUM_MESSAGE_SIZE
mgrpMsgPrefixText	LDAP_PREFIX_TEXT
msgpMsgSuffixText	LDAP_SUFFIX_TEXT
mgrpModerator	LDAP_MODERATOR_URL
mgrpRemoveHeader	LDAP_REMOVE_HEADER
mgrpRFC822MailMember*	LDAP_GROUP_RFC822
rfc822MailMember*	LDAP_GROUP_RFC822
uniqueMember	LDAP_GROUP_DN

* Note that by default either mgrpRFC822MailMember and rfc822MailMember can be used, but not both.

Extracting the User/Group Status

One of the key attributes to control the generated delivery address is the user/group and domain status. If the status of the domain, as defined by `mailDomainStatus`, is `inactive` or `deleted`, then this used as the status of the user, and the user's status is not checked. If the status of the domain is `active`, then the status of the user or group entry is used. The attributes used to define the status of the entry depends upon the schema used. This is shown below:

```
ims50:       users:     inetuserstatus or mailuserstatus
             groups:    inetmailgroupstatus
nms41:       NO STATUS ATTRIBUTES
sims40:      users:     inetsubsscriberstatus
             groups:   inetmailgroupstatus
```

If necessary, the attribute names used to determine the status of users and groups can be overridden. The option `LDAP_USER_STATUS` can be used to specify the attribute to be used for user status, and the `LDAP_GROUP_STATUS` option can be used to specify the attribute to be used for group status. Once the user or group's status has been determined, it will be one of `active`, `inactive`, `deleted`, or `hold`.

`active` - If the user or group's status is found to be active, processing continues as described in "The Location of the User," on page 551.

`inactive` - If the user or group's status is found to be `inactive`, the address is immediately rejected with a temporary error status (4xx SMTP error code).

`deleted` - If the user or group's status is found to be `deleted`, the address is immediately rejected with a permanent error status (5xx SMTP error code).

`hold` - If the user or group's status is found to be `hold`, an alias is generated so as to cause the address to be rewritten to the hold channel. The alias generated is controlled by the pattern specified by the `HOLD_TEMPLATE` MTA option. The default value for the template is

```
$M?$2I@hold-daemon
```

The meaning of the tags in the pattern are described in "Generating Delivery Addresses Using DELIVERY_OPTIONS," on page 552. If the address given was

```
robinson.crusoe@desert.island.siroe.com
```

and the matching entry gave a UID of `rcrusoe` in a hosted domain of `island.siroe.com`, the alias generated would be

```
rcusoe?island.siroe.com@hold-daemon
```

This address then matches the rewrite rule in `.../imta/config/imta.cnf`

```
hold-daemon    $U%$H@hold-daemon
```

which matches but does not modify the address so that mail is delivered to the hold channel.

Extracting the UID

All valid user entries in the directory must have a `uid` attribute, and group entries *may* have a `uid`. The `uid` is used to generate the delivery address. If a user entry does not have a `uid` attribute, the entry is ignored. If a user entry has multiple `uid` attributes, only the first is used.

Sometimes, the `uid` attribute in the directory may contain more information than is needed. For example, entries in a hosted domain may have the form: real `uid`, a separator character (defined by the `domainUidSeparator` attribute), and then a domain (example: `uid=walter@siroe.com`). If the separator character is present in the `uid`, the `uid` used for the construction of aliases is only that part before the separator character.

If it is necessary to use an attribute other than `uid` as the `uid` for the delivery address, then the `LDAP_UID` option can be used to specify that other attribute name.

The Location of the User

Once a user or group has been identified as an active user, the MTA must check that the user is local to this MTA. To be deemed local, an entry must have a `mailhost` attribute that matches either the `local.hostname configutil` attribute, or one of the names specified by the `local.imta.hostnamealiases configutil` attributes. If the user is local, the MTA goes to the next step—making sure the message does not exceed the size limit.

If the mailhost cannot be matched against any of the names for this MTA, a new address of the form

```
@mailhost:user@domain
```

is generated. This is a source routed RFC822 address, and will be processed through the rewrite rules. For a source routed address, the rewrite rules look at the source route address rather the domain part.

If a user entry has no `mailhost` attribute, then the generated address will use the `mailRoutingSmarthost` associated with the domain:

```
@smarthost:user@domain
```

If a user entry has no `mailhost` attribute and the domain has no `mailRoutingSmartHost`, the address is discarded and a 5xx error reported.

If a group entry has no mailhost attribute, the group will be processed locally. This apparent inconsistency is important because sometimes it makes sense for groups to be expanded on any inbound relay MTA, rather than on a particular server.

Extracting the Size Limit

There is one final check that the MTA has to perform before the delivery addresses are constructed (for users), or the group is expanded. This last check ensures that the mail message does not, by itself, exceed the mailMsgMaxBlocks attribute for the user, or, if that attribute is not set, the mailDomainMsgMaxBlocks attribute for the domain. If the message is too big, the address is rejected with a 5xx size exceeded error.

Generating Delivery Addresses Using DELIVERY_OPTIONS

If the entry found is a user entry, it remains only to generate the delivery addresses for the user that will cause the mail to be routed back through the rewrite rules to the appropriate channels. The delivery address generating process also takes place for groups, but for groups there are some additional concerns that will be addressed in a later section.

Delivery addresses are generated through a set of patterns. The patterns used depend upon the values defined for the attribute mailDeliveryOption. A delivery address is generated for each valid mailDeliveryOption. The patterns are defined by the MTA option DELIVERY_OPTIONS, which can be defined in .../imta/config/option.dat. The default value for DELIVERY_OPTIONS is

```
DELIVERY_OPTIONS=*mailbox=$M%$2I+$2S@ims-ms-daemon,
       &members=*,
       *native=$M@native-daemon,
       *unix=$M@native-daemon,
       &file=+$F@native-daemon,
       hold=$M?$2I@hold-daemon,
       &$members_offline=*,
       program=$M%$P@pipe-daemon,
       forward=**,
       *autoreply=$M@autoreply-daemon
```

The value of DELIVERY_OPTIONS is a set of rules separated by commas. The left side of each rule is the name of a delivery method (example, mailbox, unix, forward) the right hand side is the pattern for construction the delivery address. Each rule may be preceded by one or two special flag characters that affect how and when the rule is applicable. The flag characters are

* this rule applies to users only

& this rule applies to groups only

$ this tag causes the message to be enqueued on the `reprocess` channel so that the expansion can take place offline

Thus, the delivery methods `mailbox`, `native`, `unix` and `autoreply` can only be used by users. The delivery methods `members` and `members_offline` can only be used by groups, and the delivery methods `program` and `forward` can be used by both users and groups.

The right hand side consists of simple substitution text and tags that insert the values of various LDAP attributes. See "Substitution Tags (Case-insensitive)," on page 337.

Generating Delivery Address—Example

Consider, as an example, a message sent to the address

`robinson.crusoe+goats@desert.island.siroe.com`

and suppose for the sake of the example that his directory entry contains the attributes

```
UID: rcrusoe@desert.island.siroe.com
mail: robinson.crusoe@desert.island.siroe.com
mailDeliveryOption: mailbox
mailDeliveryOption: native
mailDeliveryOption: program
mailDeliveryOption: forward
mailDeliveryOption: autoreply
mailProgramDeliveryInfo: capriform.msg
mailForwardingAddress: friday@desert.island.siroe.com
mailForwardingAddress: hulahula@londonbank.siroe.com
```

then the original address will generate six aliases, one for each delivery method, `mailbox`, `native`, `program` and `autoreply`, and two for delivery method `forward`.

The pattern for `mailbox`, `$M%$2I+$2S@ims-ms-daemon`, is one of the more complicated.

Table B-4 Pattern Expansion for Delivery Option `mailbox`

Pattern piece	Action	result
`$M`	generates `rcrusoe`	`rcrusoe`
`%`	generates `%`	`rcrusoe%`

Table B-4 Pattern Expansion for Delivery Option `mailbox`

Pattern piece	Action	result
`$2I`	generates `desert.island.siroe.com`	`rcrusoe%desert.island.siroe.com`
`+`	generates `+`	`rcrusoe%desert.island.siroe.com+`
`$2S`	generates `goats`	`rcrusoe%desert.island.siroe.com+goats`
`@ims-ms-daemon`	generates `@ims-ms-daemon`	`rcrusoe%desert.island.siroe.com+goats @ims-ms-daemon`

This resulting address has a domain part that exactly matches the channel tag for the `ims-ms` channel, so is routed to that channel without further rewriting.

The pattern for `native`, `$M@native-daemon`, is simpler.

Pattern expansion for delivery option native.

Table B-5 Pattern Expansion for Delivery Option `native`

Pattern piece	Action	Result
`$M`	generates `rcrusoe`	`rcrusoe`
`@native-daemon`	generates `@native-daemon`	`rcrusoe@native-daemon`

The resulting address has a domain part that exactly matches the channel tag for the native channel, so is routed to that channel without further rewriting.

The pattern for autoreply, `$M@autoreply-daemon`, is very similar.

Table B-6 Pattern Expansion for Delivery Option `autoreply`

Pattern piece	Action	Result
`$M`	generates `rcrusoe`	`rcrusoe`
`@autoreply-daemon`	generates `@autoreply-daemon`	`rcrusoe@autoreply-daemon`

This resulting address has a domain part that exactly matches the channel tag for the autoreply channel, so is routed to that channel without further rewriting.

The pattern for `program`, `$M%$P@pipe-daemon`, is almost the same:

Table B-7 Pattern Expansion for Delivery Option `program`

Pattern piece	Action	Result
`$M`	generates `rcrusoe`	`rcrusoe`
`%`	generates `%`	`rcrusoe%`
`$P`	generates `prog`	`rcrusoe%prog`
`@pipe-daemon`	generates `@pipe-daemon`	`rcrusoe%prog@pipe-daemon`

This resulting address has a domain part that exactly matches the channel tag for the pipe channel, so is routed to that channel without further rewriting.

The pattern for `forward`, `**`, simply results in the values of the attribute `mailForwardingAddress` being used. This results in the addresses

```
friday@desert.island.siroe.com
hulahula@londonbank.siroe.com
```

being generated. Thus, the message sent to `robinson.crusoe` generates the following delivery addresses and is delivered to the following channels:

```
rcrusoe%desert.island.siroe.com+goats@ims-ms-daemon      ims-ms
rcrusoe@native-daemon                                    native
rcrusoe@autoreply-daemon                                 autoreply
rcrusoe%prog@pipe-daemon                                 pipe
friday@desert.island.siroe.com
hulahula@londonbank.siroe.com
```

SIEVE Rules

The final LDAP attribute obtained from the user's entry is `mailSieveRuleSource`. This contains the SIEVE filter rules for the user. These rules are not applied until the message is on the point of being enqueued to the delivery channel. Although the SIEVE filter is obtained while the MTA is expanding aliases, the SIEVE rules are not used until after the resulting delivery addresses have been expanded and are being sent to the `ims-ms`, `native`, `autoreply` or `pipe` channels. Note that this is a change in behavior from the non-dirsync mode of operation where only mail delivered to the `ims-ms` channel is processed through the SIEVE rules.

Processing Group Entries

There are four program delivery options available for groups. These are `program`, `forward`, `members`, and `members_offline`.

`Program` and `forward` are handled exactly as for users.

The pattern for both `members` and `members_offline` is `*`, which invokes the full weight of the group expansion processing that will be described in the next section.

The rule for `members_offline` is preceded by a `$`, which means that the group expansion takes place on the `reprocess` channel. If the enqueuing channel is other than the `reprocess` channel—and initially the enqueuing channel is almost certainly one of the `tcp_` channels—then the processing for the address stops and the original address is accepted and the message enqueued for the `reprocess` channel. When the `reprocess` channel runs, the same logic for processing the address is engaged, except that the enqueuing channel is the `reprocess` channel, and so `members_offline`, with its `$`, is processed exactly the same as `members`.

In principle, processing groups is straight forward: there are a couple of attributes that list the members of the group, either as email addresses or as distinguished names. In either case, those addresses are used as part of the result for the expansion of the group.

In practice, there is a lot more to group processing than that, and there are over a dozen other attributes that can affect the processing of a group entry.

Details on Processing Group Entries

The MTA processes a group entry by considering each of the various group processing options in turn. The order in which the options is processed is important. The group attributes can be divided loosely into three types:

- Attributes that provide parameters to the processing such as `mailRejectText`. These don't affect what can or can't be done, but provide some input to the process.
- Attributes that control under what circumstances the mail can be sent to the list. This include such attributes as `mgrpAllowDomain`, which specifies which domains are allowed to submit messages to the group. These attributes are processed in the order in which they are listed in the table below.
- Attributes that give the actual membership of the list.

The tables below list group processing attributes.

Table B-8 Attributes Providing Parameters for Group Processing

Attribute	Description
mailRejectText	Provides the text to be returned as an SMTP response if any of the authentication mechanisms associated with the group cause the message to be rejected. This attribute ought to be a single valued attribute in US-ASCII only to comply with the protocol rules of SMTP. If the attribute is multivalued, only the first attribute is used. If the value is more than one line, only the first line is used.
mgrpMsgMaxSize	Obsolete attribute. You should use mailmsgMaxBlocks instead as it is checked earlier in the processing of the entry. If the message exceeds this size (specified in bytes), the message will be rejected with a *message too large* error.
mgrpAuthPassword	Specifies the password for the group and is used if the mgrpBroadcasterPolicy specified requires a password.
mgrpErrorsTo	The envelope originator (MAIL FROM) address is set to the value of this attribute if it is specified. If it is not specified the envelope originator of the message is left unchanged.
mgrpAddHeader	(unsupported as yet)
mgrpRemoveHeader	(unsupported as yet)
mgrpMsgPrefixText	(unsupported as yet)
mgrpMsgSuffixText	(unsupported as yet)

Table B-9 Mail Group Access Control Attributes

Attribute	Description
mgrpBroadcasterPolicy	Specifies what level of authentication is required to send a message to the group. Possible values are: SMTP_AUTH_REQUIRED or AUTH_REQ either of which mean that the SMTP AUTH command must have been used to identify the sender before they can post to the group. PASSWORD_REQUIRED, PASSWD_REQUIRED, or PASSWD_REQ, any of which means that the password to the list specified by the mgrpAuthPassword attribute must appear in an Approved: header field in the message. NO_REQUIREMENTS, which is the same as the attribute not being present and means that there are no special requirements.

Table B-9 Mail Group Access Control Attributes

Attribute	Description
mgrpAllowedDomain	Multivalued. Lists the domains from which users can submit messages to the group. If absent, users from any domain can post to the group.
mgrpDisallowedDomain	Multivalued. Lists the domains from which users cannot post to the group.
mgrpAllowedBroadcaster	URLs which, when expanded, generate a list of addresses permitted to send to the list. If an address resulting from the expansion of a URL is a group, the group is expanded to generate a further list, but that is the limit to which the MTA expands the URL. The envelope from address for the message is checked against each address resulting from this expansion, and only if it matches one is the message allowed.
mgrpDisallowedBroadcaster	URLs which, when expanded, generate a list of addresses not permitted to send to the list. If an address resulting from the expansion of a URL is a group, the group is expanded to generate a further list, but that is the limit to which the MTA expands the URL. The envelope from address for the message is checked against each address resulting from this expansion and only if it does not match any is the message allowed.
mgrpModerator	URLs which, when expanded generate a list of addresses permitted to send to the list. If an address resulting from the expansion of a URL is a group, the group is expanded to generate a further list, but that is the limit to which the MTA expands the URL. The envelope from address for the message is checked against each address resulting from this expansion. If the envelope from address matches one of these addresses, the message is from a moderator and is allowed to post to the group. If the envelope from address does not match any of these addresses, then the message will be sent to the list of moderator addresses just expanded and will not be delivered to the group in any other way. That is, all the attributes in the group member table below are ignored, and any `program` or `forward` delivery options are ignored.

Table B-10 Mail Group Expansion Attributes

Attribute	Description
`mgrpDeliverTo`	URLs which, when expanded, generate a list of addresses. If an address resulting from the expansion of a URL is a group, the group is expanded to generate a further list, and so on. Duplicate addresses are eliminated, but it is possible to construct groups that refer to each other so that infinite recursion occurs. The MTA resolves this be allowing nested expansion of lists to 10 levels.
`memberURL`	Another list of URLs which are expanded in the same way as `mgrpDeliverTo`.
`uniqueMember`	Distinguished names of group members. Each DN can refer either to a user entry, a group entry, or to a subtree in the directory, in which case all entries in that tree are expanded.
`mgrpRFC822MailMember`, `rfc822MailMember`	The values of these attributes are the mail addresses of members of the group. In any given entry only one of these attributes is permitted. `rfc822MailMember` is supported only to provide backwards compatibility with the Netscape Messaging Server.

Alias caching

All this LDAP activity could seriously impact the performance of the MTA. To mitigate this, the MTA process caches the results of the LDAP lookups. This caching is controlled by the following options, shown with their default values:

```
ALIAS_ENTRY_CACHE_SIZE=1000
ALIAS_ENTRY_CACHE_TIMEOUT=600
ALIAS_ENTRY_CACHE_NAGATIVE=0
```

This means that the maximum number of cache entries held is 1000, and the maximum length of time an entry is held is ten minutes (600 seconds). The cache entries are larger than the domain cache entries, but if you have sufficient memory on your system, it might well be worth increasing the cache size. `ALIAS_ENTRY_CACHE_NEGATIVE` controls whether or not alias match failures are cached. By default they are not. Not caching failures speeds up the activation of new users, and it is unlikely that repeated failed attempts are going to be made to deliver the same user at a frequency that will impact the performance of the system.

Reverse Address Translation

Reverse addresses, such as `From:` headers, generally are normalized as they flow through the MTA. (Normalized means that in the header address, the personal name is moved to the front, and comments get moved to the back. Furthermore, for `From:` addresses, the address is looked up, and if it is found as a `mailalternateaddress`, that mail address is used instead.) The general principle applied is that the first mail address listed in a user's directory entry is the address that should be used. The process involves looking up the domain part of the address in the DC tree to find the subtree of the user and group tree to search for the address to look up, and then looking for an entry that contains any email address that matches the one we were given, and returning the first mail address in that entry. This is a very similar process to alias processing.

Direct LDAP address translation relies on two options being set in `.../imta/config/option.dat`

```
USE_REVERSE_DATABASE=4
REVERSE_URL=ldap:///$V?mail?sub?$Q
```

`USE_REVERSE_DATABASE=4` tells the MTA not to use the old reverse database, but instead to use the direct LDAP mechanisms. The `REVERSE_URL` is very similar to the `ALIAS_URL0` URL discussed in "Finding the User/Group Entry in the LDAP Directory," on page 545. The `$V` tag is expanded in the way described in that section. The `$Q` tag is similar to the `$R` tag used in the standard `ALIAS_URL0`, but produces a filter that searches attributes containing addresses that may match the reverse address the MTA is trying to match. The filter generated by `$R` depends on the setting of the `local.imta.schematag` configutil option:

```
ims50         mail,mailalternateAddress
nms41         mail,mailalternateAddress
sims40        mail,rfc822mailalias
```

You can override the attributes to be used for the search by specifying them in the MTA option `LDAP_MAIL_REVERSES`.

The actual search generated is very similar to the search generated by the `$R` tag used for alias lookup: it searches not only for the address originally given, but also for the address with the domain actually found in the DC tree substituted. This is described in "Finding the User/Group Entry in the LDAP Directory," on page 545.

If the reverse address lookup fails, the reverse address is left unchanged.

As with the other LDAP lookups, the results of reverse address lookups are cached. The size and time-out for this cache are controlled by the options, shown below with their default values:

```
REVERSE_ADDRESS_CACHE_SIZE=10000
REVERSE_ADDRESS_CACHE_TIMEOUT=600
```

Implications of Changing to Direct LDAP Mode

For MTA address translation, there is almost no net effect of changing from the `dirsync` mode of operation to the direct LDAP mode, except that the process is configurable and the mechanism more transparent. There is, however, a change in the way hosted domains work. In the `dirsync` mode, all subdomains of hosted domains are implicitly owned, rather like setting `DOMAIN_UPLEVEL=3` in the direct LDAP mode. However, only the actual domain configured is owned for the principle domain, rather like setting `DOMAIN_UPLEVEL=0` in the direct LDAP mode. This dichotomous mode of operation is not available in the direct LDAP mode: you have to make up your mind which way you want domain ownership to work. This difference probably makes no difference, but be aware.

Clearly there is an impact on the overall way the system performs. The areas affected are:

- A changed LDAP load.
- Reduced dependency on databases.
- Changed overall mail throughput.

Changed LDAP Load

The `dirsync` process made few but potentially massive queries against the LDAP directory. In the direct LDAP mode, the MTA makes many small queries on the directory. The net effect of this would be a substantial reduction in throughput were it not for the caching that is used. However the load imposed on the directory becomes much more conventional, and therefore the system is more scalable. Whereas with `dirsync` it was difficult to scale an MTA much above about six million users, in direct LDAP mode it should be possible to provision for tens of millions of users.

Reduced Dependency on Databases

In the dirsync mode, the MTA depended on a number of databases—in particular the alias and domain databases—for its operation. These databases are complicated disk structures and can become corrupt if a system fails suddenly. This has proved to be a problem area for high availability systems. In direct LDAP mode, the MTA has almost no dependency on databases.

Changed Overall Mail Throughput

The increased use of the directory and decreased use of databases has an impact on throughput. Extracting information from the directory and processing it into the form required is more expensive than simply looking up the result in a database. However, if the entry is found in a cache, the overall cost is less than looking it up in a database. This means that if most mail is processed to a few users whose entries have been cached, throughput goes up. If mail is scattered throughout a large user community, throughput will go down.

Performance Tuning Mail Throughput for Direct LDAP Mode

The performance of the system is sensitive to the size of the alias cache, set by ALIAS_ENTRY_CACHE_SIZE. The default value for this, 1000, is probably too small for any significant system: these cache entries can be large, about 2K bytes, and the default value was chosen to avoid overloading a small evaluation system. It is probably reasonable to take this value up to 10,000, or even 50,000 on a large system. For this change to be useful, it is also important to increase the value of MAX_LIFE_CONNS in dispatcher.cnf. MAX_LIFE_CONNS should be at least twice, and probably four times ALIAS_ENTRY_CACHE_SIZE, to take advantage of the cache.The net effect of changing from the dirsync mode of operation to the direct LDAP mode is that the address translation achieved is almost completely unchanged, but is configurable and the mechanism more transparent.

Appendix C

Administering Event Notification Service in iPlanet Messaging Server

This appendix describes what you need to do to enable the iPlanet Event Notification Service Publisher (ENS Publisher) and administer iPlanet Event Notification Service (ENS) in iPlanet Messaging Server.

This chapter / appendix contains these sections:

- Loading the ENS Publisher in iPlanet Messaging Server
- Running Sample Event Notification Service Programs
- Administering Event Notification Service

For more information on ENS and ENS APIs, see the *iPlanet Messaging and Collaboration Event Service Notification Manual* at the iPlanet Calendar Server and Messaging Server Documentation web page.

Loading the ENS Publisher in iPlanet Messaging Server

The Event Notification Service (ENS) is iPlanet's underlying publish-and-subscribe service. ENS acts as a dispatcher used by iPlanet applications as a central point of collection for certain types of *events* that are of interest to them. Events are changes to the value of one or more properties of a resource. Any application that wants to know when these types of events occur registers with ENS, which identifies events in order and matches notifications with subscriptions.

ENS and iBiff (the ENS publisher for iPlanet Messaging Server) are bundled starting with iPlanet Messaging Server. By default ENS is enabled, however, iBIFF is not loaded. (See "To Load the ENS Publisher on iPlanet Messaging Server."

In order to subscribe to notifications in iPlanet Messaging Server, you need to load the `libibiff` file on the iPlanet Messaging Server host then stop and restart the messaging server.

To Load the ENS Publisher on iPlanet Messaging Server

Perform the following steps from the command line. In these steps, the location of the iPlanet Messaging Server installation directory is *server_root*, and the iPlanet Messaging Server user is *mailsrv*. Typical values for these variables are `/usr/iplanet/server5`, and `mailsrv`, respectively.

1. As `mailsrv`, run the `configutil` utility to load the `libibiff` file.

   `cd` *server_root*`/msg-`*instance*

```
./configutil -o "local.store.notifyplugin" -v "server_root/bin/msg/lib/libibiff"
```

2. As `root`, stop then restart the messaging server.

   `cd` *server_root*`/msg-`*instance*

   `./stop-msg`

   `./start-msg`

3. You are now ready to receive notifications through ENS. See “Running Sample Event Notification Service Programs” for more information.

Running Sample Event Notification Service Programs

iPlanet Messaging Server contains sample programs to help you learn how to receive notifications. These sample programs are located in the *server_root*`/bin/msg/enssdk/examples` directory.

To Run the Sample ENS Programs

1. Change to the *server_root*`/bin/msg/enssdk/examples` directory.

2. Using a C compiler, compile the `apub` and `asub` examples using the `Makefile.sample` file. Set your library search path to include the *server_root*`/bin/msg/lib` directory.
3. Once the programs have been compiled, you can run them as follows in separate windows:

   `apub localhost 7997`

   `asub localhost 7997`

   Whatever is typed in the `apub` window should appear on the `asub` window. Also, if you use the default settings, all iBiff notifications should appear in the `asub` window.
4. To receive notifications published by iBiff, write a program similar to `asub.c`

   For more information on the sample programs, and writing your own programs for ENS, see the *iPlanet Event Notification Service for Messaging and Collaboration Manual.*

NOTE Once you set your library search path to include the *server_root*`/bin/msg/lib` directory, you can no longer stop and start the directory server. The workaround is to remove the entry from the library search path.

Administering Event Notification Service

Administering ENS consists of starting and stopping the service, and changing the configuration parameters to control the behavior of the iBiff publisher for ENS.

Starting and Stopping ENS

You use the `start-msg ens` and `stop-message ens` commands to start and stop the ENS server. You must be `root` to run these commands.

To Start and Stop ENS

- To start ENS:

  server_root`/msg-`*instance*`/start-msg ens`

- To stop ENS:

  server_root`/msg-`*instance*`/stop-msg ens`

iPlanet Event Notification Service Configuration Parameters

Several configuration parameters control the behavior of iBiff. Use the `configutil` utility program to set these parameters.

Table C-1 iBiff Configuration Parameters

Parameter	Description
`local.store.notifyplugin.maxHeaderSize`	Specifies the maximum size (in bytes) of the header that will be transmitted with the notification. The default is 8192 bytes.
`local.store.notifyplugin.maxBodySize`	Specifies the maximum size (in bytes) of the body that will be transmitted with the notification. The default is 100 bytes.
`local.store.notifyplugin.`*eventType*`.enable`	Specifies if the given event type will generate a notification. See the *iPlanet Messaging Server for Messaging and Collaboration Manual* for the various *eventTypes* such as `ReadMsg`, `NewMsg`, and so on. The legal values are 1 (to enable) and 0 (to disable). The default value is 1; that is, setting `local.store.notifyplugin.ReadMsg.enable` to 0 will disable `ReadMsg` notifications.
`local.store.notifyplugin.ensHost`	Specifies the hostname of the ENS server. The default is `127.0.0.1`.
`local.store.notifyplugin.ensPort`	Specifies the TCP port of the ENS server. The default is `7997`.
`local.store.notifyplugin.ensEventKey`	Specifies the event key to use for ENS notifications. The default is `enp://127.0.0.1/store`. The hostname portion of the event key is not used to determine the ENS host. It is simply a unique identifier used by ENS. This key is what the subscriber should subscribe to in order to be notified of events matching this key.

Appendix D

Managing Mail Users and Mailing Lists

This appendix describes how to use the Console interface to create and manage your users' mail accounts and mailing lists. It is recommended that you **NOT** use the Console interface as described here for creating and managing user and mailing lists. Instead it is recommended that the iPlanet Delegated Administrator for Messaging for the Delegated Administrator command line utilities be used to create and modigy users and mailing lists. Refer to the *iPlanet Messaging Server Reference Manual* for user / group command-line utilities.

CAUTION Creating users and groups with the Console interface as described in this appendix will prevent you from viewing and modifying them with the Delegated Administrator. iPlanet recommends that you use the iPlanet Delegated Administrator for Messaging, Delegated Administrator command line utilities, or the instructions in the iPlanet Messaging Server Provisioning Guide to create / modify users and groups.

This appendix has the following sections:

- Managing Mail Users
- Managing Mailing Lists
-

NOTE If you install iPlanet Directory Server 5.1, you must manage it through iPlanet Console 5.0 (installed with Directory Server 5.1). iPlanet Messaging Server 5.2 must be managed through Netscape Console 4.2 (installed with Messaging Server 5.2).

Managing Mail Users

To Access 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.

To Create 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.

To Access 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.

To Specify 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 "To Access Mail Users" on page 568.
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.

To Configure 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 "To Access Mail Users" on page 568.
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 571.
   - To specify program delivery, follow the instructions in "Specifying Program Delivery" on page 572.
   - To specify UNIX delivery, follow the instructions in "To Specify UNIX Delivery" on page 573.
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 341.
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 351), 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 351), 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 "To Deliver Messages to Programs Using the Pipe Channel" on page 275".

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.

To Specify 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

To Specify 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 "To Access Mail Users" on page 568.
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.

To Configure 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 "To Access Mail Users" on page 568.
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.

To Configure 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 "To Access Mail Users" on page 568.
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

To Access 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.

To Create 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.

To Access 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.

To Specify 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 "To Access Mailing Lists" on page 576.
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 delivering messages posted 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.

To Specify 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*.

To Define 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 D-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.

To Add 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 "To Access Mailing Lists" on page 576.
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 "To Define Dynamic Membership Criteria" on page 580.
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.

To Define 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 “To Access Mailing Lists” on page 576.
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.

To Define 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 “To Access Mailing Lists” on page 576.
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.

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)**, **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.

administration console See **Console**.

administration domain A region of administraive control. See also **domain**.

administration privileges A 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**, **mail 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. See also **Deny filter**.

allowed attributes The attributes that optionally can be present in entries using a particular object class, but are not required to be present. See also **attributes**, **required attributes**.

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.

attributes LDAP data is represented as attribute-value pairs. Any specific piece of information is associated with a descriptive attribute. See also **allowed attributes**, **required attributes**.

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.

class path A path to directories and `.jar` files needed to run the servlet engine and servlet templates.

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. See also **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**.

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.

CLI See **command line interface**.

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.

command line interface Command that can be executed from the command-line. Also called utility.

comment character A character that, when placed at the beginning of a line, turns the line into a nonexecutable comment.

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.

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.

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 also **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 mail list subscriptions.

Delegated Administrator for Messaging and Collaboration. A set of interfaces (GUI and utilities) 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**.

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. See also **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 to 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 See **mail list**.

distribution list owner See **mail list owner**.

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 domain 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`. See also **domain**.

DNS spoofing A form of network attack in which a DNS server has been subverted to provide false information.

document root A directory on the server machine that contains files, images, and data that will be displayed to users accessing iPlanet Web Server.

domain Resources under control of a single computer system. See also **administration domain**, **DNS domain**, **hosted domain**, **virtual domain**.

domain administrator User who has administrative privileges to create, modify, and delete mail users, mail lists, and family accounts in a hosted domain by using the Delegated Administrator for Messaging and Collaboration 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**, **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 mail lists. Users send messages to a single address (for example, `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 mail lists. The act of converting a message addressed to a mail list into enough copies for each mail list member.

EXPN An SMTP command for expanding a mail 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 mail 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 also **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. See also **domain**, **virtual domain**.

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**, **IP address**.

host name hiding The practice of having domain-based email addresses that do not 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 also **Messenger Express**.

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 commands A set of command line utilities for managing domain administrators, users, and groups.

imsimta commands A set of command line utilities for performing various maintenance, testing, and management tasks for the Message Transfer Agent (MTA).

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: *server-root*`/bin/msg/`. Se also **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_root See **instance directory.**

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: *server_root*/msg-*instance*/, where *instance* is the name of the server as specified at installation. See also **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**, **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 method 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, mail 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**.

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 also **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**.

mail list A list of email addresses to which a message can be sent by way of a mail list address. Sometimes called a group.

mail list owner A user who has administrative privileges to add members to and delete members from the mail 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**.

mailing list See **mail list**.

mailing list owner See **mail list owner.**

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.

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 mail list. See also mail list, expansion, moderator, and 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**.

Messenger Express Multiplexor A proxy messaging server that acts as a Multiplexor; it allows you to connect to the HTTP service of iPlanet Messaging Server (Messenger Express). The Messenger Express Multiplexor facilitates distributing mail users across multiple server machines.

MHS See **Message Handling System**.

MIME See **Multipurpose Internet Mail Extension**.

MMP See **Messaging Multiplexor**.

moderator A person who first receives all email addressed to a mail list before (A) forwarding the message to the mail list, (B) editing the message and then forwarding it to the mail list, or (C) not forwarding the message to the mail list. See also **mail list**, **expansion**, **member**.

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**, **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 also **directory information tree**.

naming attribute The final attribute in a directory information tree distinguished name. See also **relative distinguished name**.

naming context A specific suffix 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.

node An 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 information al purposes and requires no action from the postmaster. See also **delivery status notifications**.

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 mail lists in an organization or suborganization by using the Delegated Administrator for Messaging and Collaboration 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. See also **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. See also **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.

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 See **RDN**.

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.

required attributes Attributes that must be present in entries using a particular object class. See also **allowed attributes**, **attributes**.

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 top-level 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 software installation.

server_root The directory into which all iPlanet servers associated with a given Administration Server on a given host are installed. Typically designated *server-root*. See also **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** on UNIX platforms.

Service Control Manager Windows NT administrative program for managing services.

servlet server-side Java programs that Web servers run to generate content in response to a client request. Servlets are similar to applets in that they run on the server-side but do not use a user interface.

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. See also **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**.

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**, **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, mail lists, family accounts, and domains in an entire Messaging Server namespace by using the Delegated Administrator for Messaging and Collaboration 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. (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. See also **domain**, **hosted domain**.

VRFY An SMTP command for verifying a user name. Defined in RFC 821.

Web server A software program or server computer equipped to offer World Wide Web access. A Web server accommodates requests from users, retrieves requested files or applications, and issues error messages.

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

NUMERICS

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X