Sun Java™ System

Messaging Server 6 Administration Guide

2004Q2

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817-6266-10

Contents

List of Tables

List of Figures

About This Guide

This guide explains how to administer Sun Java™ System Messaging Server 6 and its accompanying software components. Messaging Server provides a powerful and flexible cross-platform solution to meet 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

You should read this book if you are responsible for administering and deploying Messaging Server at your site. You should also have read the *Sun Java Messaging Server Deployment Planning Guide* (`http:/docs.sun.com/doc/817-6440/`).

What You Need to Know

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

- The Internet and the World Wide Web

- Messaging Server protocols
- Sun Java System Administration Server
- Sun Java System Directory Server and LDAP
- Sun ONE Console
- System Administration and Networking on the following platforms:
- General Deployment Architectures

How This Book is Organized

This book contains the following chapters and appendixes:

- About This Guide (this chapter)
- Chapter 1, “Post-install Tasks and Layout”
- Chapter 2, “Upgrading to Sun Java Systems Messaging Server”
- Chapter 3, “Configuring High Availability”
- Chapter 4, “Configuring General Messaging Capabilities”
- Chapter 5, “Configuring POP, IMAP, and HTTP Services”
- Chapter 6, “Enabling Single Sign-On (SSO)”
- Chapter 7, “Configuring and Administering Multiplexor Services”
- Chapter 8, “MTA Concepts”
- Chapter 9, “MTA Address Translation and Routing”
- Chapter 10, “About MTA Services and Configuration”
- Chapter 11, “Configuring Rewrite Rules”
- Chapter 12, “Configuring Channel Definitions”
- Chapter 13, “Using Pre-defined Channels”
- Chapter 14, “Integrating Anti-spam and Anti-virus Programs”
- Chapter 15, “LMTP Delivery”
- Chapter 16, “Automatic Message Reply”
- Chapter 17, “Mail Filtering and Access Control”

- Chapter 18, “Managing the Message Store”
- Chapter 19, “Configuring Security and Access Control”
- Chapter 20, “Logging and Log Analysis”
- Chapter 21, “Troubleshooting the MTA”
- Chapter 22, “Monitoring the Messaging Server”
- Appendix A, “SNMP Support”
- Appendix B, “Administering Event Notification Service in Messaging Server”
- Appendix C, “Managing Mail Users and Mailing Lists with the Console Interface (NOT RECOMMENDED)”
- Appendix D, “Short Message Service (SMS)”
- Appendix E, “Installation Worksheets”
- Glossary

Document Conventions

Monospaced Font

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

Bold Monospaced Font

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

```
./installer
```

In this example, **`./installer`** is what you would type at the command line.

Italicized Font

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

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

msg_svr_base/. . .

The Messaging Server Base (*msg_svr_base*) represents the directory path in which you install the server. The default value of the *msg_svr_base* is `/opt/SUNWmsgsr`.

Italicized font is also used for variables within the synopsis of a command line utility. For example, the synopsis for the `commadmin admin remove` command is:

```
commadmin admin remove -D login -l userid -n domain -w password [-d domain]
      [-h] [-i inputfile] [-p port] [-X host] [-s] [-v]
```

In the above example, the italicized words are arguments for their associated option. For example, in the `-w` *password* option, you would substitute the Administrator's password for *password* when you enter the `commadmin admin remove` command.

Square or Straight Brackets

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

`./configutil` *[options] [arguments]*

It is possible to run the `configutil` command by itself or to list some or all `configutil` parameters and values:

```
./configutil
```

However, the presence of *[options]* and *[arguments]* indicate that there are additional optional parameters that may be added to the `configutil` command with the -p option to list all parameters with the prefix service.imap:

```
./configutil -p service.imap
```

Command Line Prompts

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

Platform-specific Syntax

Note that the examples in this book use the UNIX C shell. If necessary, make appropriate adjustments to your preferred shell.

Where to Find Related Information

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

`http://docs.sun.com`

Listed below are some of the documents that are available:

- *Sun Java System Messaging Server Release Notes*
- *Sun Java System Messaging Server Administration Guide*
- *Sun Java System Messaging Server Administration Reference*
- *Sun Java System Communications Services Schema Reference Manual*
- *Sun Java System Communications Services Event Notification Service Manual*
- *Sun Java System Communications Express Administration Guide*
- *Sun Java System Messaging Server Developer's Reference*

The Sun Java System Messaging Server product suite contains other products such as Sun ONE Console, Directory Server, and Administration Server. Documentation for these and other products can be found at the following URL:

`http://docs.sun.com/`

In addition to the software documentation, see the Sun Java System Messaging Server Software Forum for technical help on specific Messaging Server product questions. The forum can be found at the following URL:

http://swforum.sun.com/jive/forum.jsp?forum=15

NOTE Sun is not responsible for the availability of third-party Web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Where to Find This Book Online

You can find the *Sun Java System Messaging Server Administration Guide* online in PDF and HTML formats. This book can be found at the following URL:

http://docs.sun.com/doc/817-6266

Chapter 1

Post-install Tasks and Layout

This chapter assumes that you have read the *Sun Java Messaging Server Deployment Planning Guide* (http:/docs.sun.com/doc/817-6440/) and installed Messaging Server with the Sun Java™ Enterprise System installer (see the *Java Enterprise System Installation Guide* at http://docs.sun.com/doc/817-5760). Performing the following tasks should get you to a point where you have a functioning Messaging Server. You will still want to customize your deployment as well as provision and/or migrate users and groups. Customizing is described in the later chapters of this guide. Provisioning is described in the *User Management Utility Administration Guide* (http://docs.sun.com/doc/817-5703).

This chapter consists of the following sections:

To Create UNIX System Users and Groups

System users run specific server processes, and privileges need to be given to these users so that they have appropriate permissions for the processes they are running.

Set up a system user account and group for all Sun Java System servers, and set permissions for the directories and files owned by that user. To do so, follow the steps below.

NOTE For security reasons, in some deployments it may be desirable to have different system administrators for different servers. This is done by creating different system users and groups per server. For example, the system user for Messaging Server would be different from the system user for Web Server, and system administrators Messaging Server would not be able to administer the Web Server.

1. Log in as superuser.
2. Create a group to which your system users will belong. In the following example, the `mailsrv` group is created:

```
# groupadd mailsrv
```

3. Create the system user and associate it with the group you just created. In addition, set the password for that user. In the following example, the user `mail` is created and associated with the `mailsrv` group:

```
# useradd -g mailsrv mail
```

   `useradd` and `usermod` commands are in `/usr/sbin`. See UNIX man pages for more information.

4. You might also need to check the `/etc/groups` file to be sure that the user has been added to the system group that you created.

NOTE Should you decide not to set up UNIX system users and groups prior to installing Messaging Server, you will be able to specify them when you run the "To Create the Initial Messaging Server Runtime Configuration" on page 43.

To Prepare Directory Server for Messaging Server Configuration

This section provides instructions on how to run the Directory Server Setup script (`comm_dssetup.pl`) that configures your LDAP Directory Server to work with your Messaging Server, Calendar Server, or User Management Utility configurations. The `comm_dssetup.pl` script prepares the Directory Server by setting up new schema, index, and configuration data in your Directory Server. It must be run for new installations of Messaging Server.

The following topics are explained:

Location of comm_dssetup.pl

After installing the Messaging Server component through the Java Enterprise Server installer, you can access the `comm_dssetup.pl` script from the *msg_svr_base*`/lib` directory.

comm_dssetup.pl Requirements

Before you run the `comm_dssetup.pl` script, be sure to read the following requirements:

- Prior to running the `comm_dssetup.pl` script, your directory server must be installed and configured.
- Run the `comm_dssetup.pl` script as superuser.
- Run `comm_dssetup.pl` prior to running the Messaging Server, Calendar Server, or User Management Utility Initial Runtime Configuration programs.
    - Generally, if you run `comm_dssetup.pl` on a directory server for one product (for example, Calendar Server), you don't need to run it again for another product (like Messaging Server), as long as both products are using the same directory server. However, you will need to run

comm_dssetup again if you change some of the answers you gave when running comm_dssetup. For example, if you want to use a different user / group suffix for the *next* configuration of Messaging Server, say because you ran `commdirmig`.

- The `comm_dssetup.pl` script must be run on your directory server machine.
- Be sure that your directory server is running prior to running `comm_dssetup.pl`.
- Whenever you install a new version of Messaging Server, you will need to run the new version of `comm_dssetup.pl` on your Directory Server machine. New schema and new indexes may be added to each Messaging Server distribution.
- If the configuration data and user and group data are split into separate directory instances, you will need to run the `comm_dssetup` script on both instances.
- Use the version of Perl that ships with Directory Server to avoid versioning problems: *dir_server_root*`/bin/slapd/admin/bin/perl`.
- If you are running `comm_dssetup.pl` on a remote directory server:
  - Copy the `dssetup.zip` file from the *msg_svr_base*`/lib` directory to the remote directory server. You may want to copy it to a directory like `/tmp` or `/var/tmp`.
  - Unzip the `dssetup.zip` file (which contains the `comm_dssetup.pl` and required schema).
  - Run the `comm_dssetup.pl` script on the remote directory server.
- If you are running a replicated directory server, you need to make sure you run the `comm_dssetup.pl` script against the master and replica directories.
- When you run the Directory Server Setup script (`comm_dssetup.pl`) to prepare Directory Server for Messaging Server configuration, record your installation parameters in Table E-3 on page 844. You will need some of these parameters for the Messaging Server initial runtime configuration.

To Run the comm_dssetup.pl Script

The `comm_dssetup.pl` script is located in your *msg_svr_base*`/lib` directory.

You can either run `comm_dssetup.pl` in:

- “Interactive Mode” on page 35
- “Silent Mode” on page 41

Use the Installation Worksheets in the *Sun Java Messaging Server Deployment Planning Guide* (http://docs.sun.com/doc/817-6440) to record your answers.

Interactive Mode

The following questions will be asked if you specify `comm_dssetup.pl` without any arguments:

1. Introduction

```
# perl comm_dssetup.pl

Welcome to the Directory Server preparation tool for Java Enterprise
Communications Server.
(Version X.X Revision X.X)

This tool prepares your directory server for Sun Java System Messaging
Server install.

The logfile is /var/tmp/dssetup_YYYYMMDDHHSS

Do you want to continue [y]:
```

Press enter to continue. Enter `No` to exit.

2. Installation Root of Directory Server

```
Please enter the full path to the directory where the Java
Enterprise Directory Server was installed.

Directory server root [/var/opt/mps/serverroot]
```

Indicate the location of the installation root of the Directory Server on the Directory Server machine.

3. Directory Server Instance

```
Please select a directory server instance from the following
list:

[1]   slapd-varrius

Which instance do you want [1]:
```

If multiple instances of Directory Server reside on the machine, choose the one that will be configured with Messaging Server.

4. Directory Manager Distinguished Name (DN)

```
Please enter the directory manager DN [cn=Directory Manager]:
Password:
```

The Directory Manager DN (`cn=Directory Manager`) is the administrator who is responsible for the user and group data in the Organization Tree. Be sure that the Directory Manager DN you specify in this script is the same DN that you set up in your Directory Server installation as well as your Messaging Server installation.

5. User and Group Directory Server

```
Will this directory server be used for users/groups [Yes]:
```

If you enter `Yes`, you will answer questions on selecting a DC Tree base suffix (Sun LDAP Schema 1) and a User and Group base suffix for your Organization Tree.

If you enter `No`, it is assumed that this directory instance is only used to store configuration data; you will skip to the question about updating schema files. After you finish running this script against the configuration directory instance, you need to run this script against the directory instance that stores user and group data before moving on in the installation process.

6. User and Group Base Suffix

```
Please enter the Users/Groups base suffix [o=usergroup]:
```

The User and Group base suffix is the top entry in the Organization Tree which holds the namespace for user and group entries. Be sure that the User and Group base suffix you select is the same as what you specified during your Directory Server installation and in your Messaging Server installation.

NOTE If you installed Identity Server, be sure the suffix specified in Identity Server installation is the same as what you specify for this question. If you do not use the same suffix, Messaging Server will not recognize your Identity Server installation.

For more information on the Organization Tree, see Chapter 12, Provisioning and Schema Concepts for Messaging Server 6.0 in the *Sun Java Enterprise System 2003Q4 Installation Guide* at http://docs.sun.com/source/816-6874/provisioning-concepts.html.

7. Schema Type

```
There are 3 possible schema types:
  1  - schema 1 for systems with iMS 5.x data
  1.5 - schema 2 compatibility for systems with iMS 5.x data
        that has been converted with commdirmig
  2  - schema 2 native for systems using Identity Server

Please enter the Schema Type (1, 1.5, 2) [1]:
```

Choose Option `1` if you are planning to use Sun LDAP Schema 1.

Choose Option `1.5` if you plan to use Sun LDAP Schema 2, Compatibility Mode. For more information, see the *Sun Java System Communications Services Schema Migration Guide*.

Choose Option `2` if you plan to use Sun LDAP Schema 2, Native Mode.

If Identity Server is not installed and configured prior to choosing a Sun LDAP Schema 2 option, `comm_dssetup.pl` will terminate. You will be asked to rerun the program once Identity Server is installed.

For more information on your schema options, see *Sun Java Messaging Server Deployment Planning Guide* (`http:/docs.sun.com/doc/817-6440/`).

8. Domain Component (DC) Tree Base Suffix

```
Please enter the DC Tree base suffix [o=internet]:
```

NOTE In Step 7, if you choose Option 1 or 1.5, you will be asked to provide your DC Tree Base Suffix. If you choose Option 2 - Sun LDAP Schema 2 - Native Mode, you will not be asked this question.

The DC Tree mirrors the local DNS structure and is used by the system as an index to the Organization Tree that contain the user and group data entries. The DC Tree base suffix is the name of the top entry on the DC tree. You can either choose the default `o=internet` or another name.

For more information on the DC Tree or the Organization Tree, see Chapter 12, Provisioning and Schema Concepts for Messaging Server 6.0 in the *Sun Java Enterprise System 2003Q4 Installation Guide* at http://docs.sun.com/source/816-6874/provisioning-concepts.html.

9. Updating Schema Files

```
Do you want to update the schema files [yes]:
```

If you answer Yes, new elements will be added to your schema. It is recommended that you update the Directory with the new schema files each time you install newer versions of Messaging Server.

10. Configuring New Indexes

```
Do you want to configure new indexes [yes]:
```

If you answer Yes to Step 5 (User and Group Directory Server), you will be asked if you want to configure new indexes, which are used to create caches to improve efficiency of directory searches. It is recommended that you answer Yes to this question. However, there are several conditions under which you wouldn't want to create the indexes:

- If this is for a master user / group Directory Server that is only used to serve replicas, that is, there are no direct queries done against the user / group Directory Server.
- If you have a production user / group Directory Server with lots of entries in which you don't want a lot of downtime while the indexes are created.

11. Summary of Settings

```
Here is a summary of the settings that you chose:
  Server Root                       : /var/opt/mps/serverroot/
  Server Instance                   : slapd-varrius
  Users/Groups Directory            : Yes
  Update Schema                     : yes
  Schema Type                       : 1
  DC Root                           : o=internet
  User/Group Root                   : o=usergroup
  Add New Indexes                   : yes
  Directory Manager DN              : cn=Directory Manager

Now ready to generate a shell script and ldif file to modify the
Directory.
No changes to the Directory Server will be made this time.

Do you want to continue [y]:
```

A summary of your settings will be displayed before your directory configuration is updated. No changes will be made at this time.

NOTE In Step 7, if you choose Option 2: Sun LDAP Schema 2 - Native Mode, the DC Root in the Summary of Settings will be the same value that you entered for the User/Group Root.

If you want to change any of your settings, enter `No` and re-run the script.

If you enter `Yes` to continue, the `comm_dssetup.pl` script will create an LDIF file and a shell script that will be used to update the indexes and schema in your directory server:

`/var/tmp/dssetup_`*YYYYMMDDHHMMSS*`.sh`
`/var/tmp/dssetup_`*YYYYMMDDHHMMSS*`.ldif`

where *YYYYMMDDHHMMSS* indicates the time and date stamps when the files were created.

NOTE You can either choose to run the script now or later. If you choose to run the script now, enter `Yes` when asked if you want to continue. If you want to run the script later, you can invoke the script by using `/var/tmp/dssetup_`*YYYYMMDDHHMMSS*`.sh`.

Silent Mode

To enable the silent mode, specify all the arguments at one time:

Syntax

```
# perl comm_dssetup.pl -i yes|no -c Directory_Server_Root -d
Directory_instance -r DC_tree -u User_Group_suffix -s yes|no -D
"DirectoryManagerDN" -w password -b yes|no -t 1|1.5|2 -m yes|no [-S
path-to-schema-files]
```

Options

The options for this command are:

Option	Description
`-i yes|no`	Answers the following question: “Do you want to configure new indexes?” Specify `yes` to configure new indexes. Specify `no` if you don’t want to configure new indexes.
`-c` *Directory_Server_Root*	Directory Server Root path name.For example: `/var/opt/mps/serverroot`
`-d` *Directory_instance*	Directory Server instance subdirectory. For example: `slapd-budgie`
`-r` *DC_tree*	DC tree suffix. For example: `o=internet`
`-u` *User_Group_suffix*	User / Group suffix e.g. `o=usergroup`
-s yes | no	Answers the following question: “Do you want to update the schema?” Specify `yes` to update the schema files. Specify `no` if you don’t want to update the schema files.
`-D` *DirectoryManagerDN*	Directory Manager DN. For example, `"cn=Directory Manager"`
`-w` *password*	Directory Manager password
-b yes | no	Answers the following question: “Will this directory server be used for users and groups?” Specify `yes` if the directory server will be used for configuration and user / groups. Specify `no` if this directory will be only used for configuration data.

Option	Description
`-t 1|1.5|2`	Determines the schema version that you want to use for your Messaging Server: • Choose `1` for Sun LDAP Schema 1. • Choose `1.5` for Sun LDAP Schema 2 (Compatibility Mode). See the *Sun Java System Communications Services Schema Migration Guide* for more information. • Choose `2` for Sun LDAP Schema 2 (Native Mode).
`-m yes|no`	Answers the following question: "Do you want to modify the directory server?" Specify `yes` to modify the directory. Specify `no` if you don't want to modify the directory.
`-S` *path-to-schema-files*	Specifies the directory path to schema files. For example: `./schema`.

Example

```
# perl comm_dssetup.pl -i yes -c /var/opt/mps/serverroot -d slapd-budgie
-r o=internet -u o=usergroup -s yes -D "cn=Directory Manager" -w password -b
yes -t 1 -m yes
```

Once you set all the options for the `comm_dssetup.pl` script, you will see the following summary screen before the script is actually run:

```
Here is a summary of the settings that you chose:
  Server Root                       : /var/opt/mps/serverroot/
  Server Instance                   : slapd-budgie
  Users/Groups Directory            : Yes
  Update Schema                     : yes
  Schema Type                       : 1
  DC Root                           : o=internet
  User/Group Root                   : o=usergroup
  Add New Indexes                   : yes
  Schema Directory                  : ./schema
  Directory Manager DN             : "cn=Directory Manager"
```

Each option is further described in the "Interactive Mode" on page 35 section.

To Create the Initial Messaging Server Runtime Configuration

The initial runtime configuration program provides a configuration to get your Messaging Server up and running. It is meant to create an *initial runtime configuration* to setup a generic functional messaging server configuration. Thus it gives you a base working configuration from which you can make your specific customization. The program is only meant to be run once. Subsequent running of this program may result in your configuration being overwritten. To modify your initial runtime configuration, use the configuration utilities described in both the *Sun Java System Messaging Server Administration Guide* and the *Sun Java System Messaging Server Administration Reference*.

Messaging Server Pre-requisites

Before running the initial runtime configuration program, you must:

- Install and configure the Administration Server. (See the *Java Enterprise System Installation Guide* at http://docs.sun.com/doc/817-5760.)
- Install and configure the Directory Server. (See the *Java Enterprise System Installation Guide*.)
- Run the `comm_dssetup.pl` program. (See "To Prepare Directory Server for Messaging Server Configuration" on page 33).
- Record your Administration and Directory installation and configuration parameters in the checklists supplied in Appendix E, "Installation Worksheets".

Messaging Server Configuration Checklist

When you run the Messaging Server initial runtime configuration program, record your parameters in Table E-4 on page 846. To answer certain questions, refer to your Directory and Administration Server installation checklists in the *Messaging Server Deployment Planning Guide*.

Running the configure Program

The following steps walk you through configuring the Messaging Server initial runtime configuration:

1. Invoke the Messaging Server initial runtime configuration with the following command:

```
/msg_svr_base/sbin/configure [flag]
```

   You might need to use the `xhost`(1) command if you are configuring Messaging Server on a remote system.

   Table 1-1 describes optional flags you can set with the `configure` program:

Table 1-1 Optional Flags for the Messaging Server `configure` Program

Flag	Description
`-nodisplay`	Invokes a command-line configuration program.
`-noconsole`	Invokes a GUI user interface program.
`-state` *[statefile]*	Uses a silent installation file. Must be used with `-nodisplay` and `-noconsole` flags. See To Perform a Silent Installation.

   Once you run the `configure` command, the configuration program will start:

2. Welcome

   The first panel in the configure program is a copyright page. Select Next to continue or Cancel to exit.

3. Select directory to store configuration and data files

   Select the directory where you want to store the Messaging Server configuration and data files. For example, `/var/opt/SUNWmsgsr`. Symbolic links will be created under *msg_svr_base* to the configuration and data directory. For more information on these symbolic links, see"Post-Installation Directory Layout" on page 53.

   Make sure you have large enough disk space set aside for these files.

4. You will see a small window indicating that components are being loaded.

   This may take a few minutes.

5. Select Components to Configure

   Select the Messaging components that you want to configure.

   - Message Transfer Agent: Handles routing, delivering user mail, and handling SMTP authentication. The MTA provides support for hosted domains, domain aliases, and server-side filters.
   - Message Store: Provides the foundation for unified messaging services through its universal Message Store. Access to the message store is available through multiple protocols (HTTP, POP, IMAP). If you are only configuring a Message Store, you must also select the MTA.
   - Messenger Express: Handles the HTTP protocol retrieval of messages from the Message Store. If you are only configuring Messenger Express, you must also select the Message Store and the MTA.
   - Messaging Multiplexor: Acts as a proxy to multiple messaging server machines within an organization. Users connect to the Multiplexor server, which redirects each connection to the appropriate mail server. This component is not enabled by default. If you do check the MMP as well as the Message Store, they will be enabled on the same system; a warning message will appear for you to change your change port numbers after configuration (For instructions on doing so, see "Post-Installation Port Numbers" on page 55.)

     To configure the MMP, see the *Chapter 7, "Configuring and Administering Multiplexor Services,"* and the *Sun Java System Messaging Server Administration Reference* `http://docs.sun.com/doc/817-6267`.

   Check any components you want to configure, and uncheck those components you do not wish to configure.

6. Enter the system user name and the group that will own the configured files.

   For information on setting up system users and groups, see "To Create UNIX System Users and Groups" on page 32.

7. Configuration Directory Server Panel

   Enter your Configuration Directory LDAP URL, Administrator and Password.This is taken from the Administration Server configuration.

   Gather the Configuration Server LDAP URL from your Directory Server installation. See the Directory Server Installation worksheet from *Sun Java Messaging Server Deployment Planning Guide* (http:/docs.sun.com/doc/817-6440/).

   The Directory Manager has overall administrator privileges on the Directory Server and all Sun Java System servers that make use of the Directory Server (for example, the Messaging Server) and has full administration access to all entries in the Directory Server. The default and recommended Distinguished Name (DN) is `cn=Directory Manager` and is set during Directory Server configuration.

8. User / Group Directory Server Panel

   Enter your Users and Groups Directory LDAP URL, Administrator and Password.

   Gather the User / Group Server LDAP URL information from the host and post number information from your Directory Server installation. See the Directory Server Installation worksheet from *Sun Java Messaging Server Deployment Planning Guide* (http:/docs.sun.com/doc/817-6440/).

   The Directory Manager has overall administrator privileges on the Directory Server and all Sun Java System servers that make use of the Directory Server (for example, the Messaging Server) and has full administration access to all entries in the Directory Server. The default and recommended Distinguished Name (DN) is `cn=Directory Manager` and is set during Directory Server configuration.

   If you are installing against a replicated Directory Server instance, you must specify the credentials of the replica, not the master directory.

9. Postmaster Email Address

   Enter a Postmaster Email Address.

   Select an address that your Administrator will actively monitor. For example, `pma@siroe.com` for a postmaster on the siroe domain. Note that this address cannot begin with “Postmaster.”

   Note that the user of the email address is not automatically created. Therefore, you will need create it by using a provisioning tool.

10. Password for administrator accounts

    Enter a universal password that will be used for service administrator, server, user / group administrator, end user administrator privileges as well as PAB administrator and SSL passwords.

    After the initial runtime configuration, you might change this password for individual administrator accounts. For more information, see "To Modify Your Passwords" on page 94.

11. Default Email Domain

    Enter a Default Email Domain.

    This email domain is the default that is used if no other domain is specified. For example, if `siroe.com` is the default email domain, then the is the domain to which messages addressed to user IDs without a domain will be sent.

    If you are using the User Management Utility, the command-line interface for provisioning users and groups with Sun LDAP Schema 2, you will want to specify the same default domain during its configuration. For more information, see the *User Management Utility Administration Guide* (`http://docs.sun.com/doc/817-5703`).

12. Organization DN

    Enter an Organization DN under which users and groups will be created. The default is the email domain prepended to the user / group suffix

    For example, if your user / group suffix is `o=usergroup`, and your email domain is siroe.com, then the default is `o=siroe.com, o=usegroup` (where `o=usergroup` is your user / group Directory suffix which was specified in "To Prepare Directory Server for Messaging Server Configuration" on page 33).

    If you choose the same your user / group Directory suffix as your Organization DN, you may have migration problems if you decide to create a hosted domain. If you want to set up a hosted domain during initial runtime configuration, then specify a DN one level below the User / Group suffix.

13. Ready to Configure

    The configuration program will check for enough disk space on your machine and then outline the components it is ready to configure.

    To configure the Messaging components, select Configure Now. To change any of your configuration variables, select Back. Or to exit from the configuration program, select Cancel.

14. Starting Task Sequence, Sequence Completed, and Installation Summary Panels

    You can read the installation status by selecting Details on the final Installation Summary page. To exit the program, select Close.

    A log file is created in */msg_svr_base*/install/configure_*YYYYMMDDHHMMSS*.log, where *YYYYMMDDHHMMSS* identifies the 4-digit year, month, date, hour, minute, and second of the configuration.

    An initial runtime configuration is now set up for your Messaging Server. To change any configuration parameter, refer to other parts of this document for instructions on doing so.

To start Messaging Server, use the following command:

```
/opt/SUNWmsgsr/sbin/start-msg
```

To Perform a Silent Installation

The Messaging Server initial runtime configuration program automatically creates a silent installation *state* file (called saveState) that can be used to quickly configure additional Messaging Server instances in your deployment where the Messaging Server Solaris packages have been installed. All of your responses to the configuration prompts are recorded in that file.

By running the silent installation, you instruct the configure program to read the silent installation *state* file. The configure program uses the responses in this file rather than ask the same installation questions again for subsequent initial runtime configurations of Messaging Server. When you use the *state* file in a new installation, you are not asked any questions. Instead, all of the *state* file responses are automatically applied as the new installation parameters.

The silent installation saveState *state* file is stored in the */msg_svr_base*/install/configure_*YYYYMMDDHHMMSS* directory, where *YYYYMMDDHHMMSS* identifies the 4-digit year, month, date, hour, minute, and second of the saveState file.

To use the silent installation *state* file to configure another Messaging Server instance on another machine in the deployment, follow these steps:

1. Copy the silent installation *state* file to the installation directory on the machine where you are performing the new installation.

2. Review and edit the silent installation *state* file as necessary.

   You will probably want to change some of the parameters and specifications in the *state* file. For example, the default email domain for the new installation may be different than the default email domain recorded in the *state* file. Remember that the parameters listed in the *state* file will be automatically applied to this installation.

3. Run the following command to configure other machines with the silent installation file:

   msg_svr_base`/sbin/configure -nodisplay -noconsole -state \`
   fullpath`/saveState`

   where *fullpath* is the full directory path of where the `saveState` file is located. (See Step 1 of this section).

NOTE After running the silent installation program, a new *state* file is created from the silent installation in directory location: *msg_svr_base*`/install/configure_`*YYYYMMDDHHMMSS*`/saveState`, where *YYYYMMDDHHMMSS* identifies the 4-digit year, month, date, hour, minute, and second of the `saveState` file.

To Install Messaging Server against a Directory Server Replica

There might be limitations that prevent you from installing Messaging Server against a Directory Server replica:

- You do not have Directory Server master credentials.
- Messaging Server cannot communicate directly with the Directory Server master.

To install Messaging Server against a Directory Server replica, follow these steps:

1. Run the `comm_dssetup.pl` program against all Directory Servers including the Directory Server replicas as noted in “comm_dssetup.pl Requirements” on page 33.

2. Run the Messaging `configure` program (located in *msg_svr_base*`/sbin/configure`) using the replicated Directory Server credentials as described in Step 7 and Step 8 in To Create the Initial Messaging Server Runtime Configuration.

   Because of invalid privileges, the `configure` program will fail in trying to configure the Directory Server Administrators. It will, however, produce the *msg_svr_base*`/configure/*.ldif` files that are needed to allow proper privileges to the Directory Server replicas.

3. Move the `*.ldif` files to the Directory Server master.

4. Run the `ldapmodify` command on the `*.ldif` files.

   See the Sun Java System Directory Server documentation for more information on `ldapmodify` or in the *msg_svr_base*`/install/configure_`*YYYYMMDDHHMMSS*`.log`.

5. Re-run the `configure` program.

   Your Directory Server replica (and master) are now configured to work with your Messaging Server.

To Install Messaging Server Provisioning Tools

The following sections provide a summary of install information about the supported provisioning tools:

- “Sun Java System Delegated Administrator for Messaging” on page 51
- “LDAP Provisioning Tools” on page 52
- “User Management Utility” on page 53
- “To Create UNIX System Users and Groups” on page 32

Sun Java System Delegated Administrator for Messaging

To install Delegated Administrator, you need to download it from the Sun Software page. Contact your Sun Java System representative for information on the download location information.

NOTE Delegated Administrator can only be installed after Messaging Server and Web Server are installed and configured. For more information on installing Delegated Administrator, see the Sun Java System Delegated Administrator documentation.

Delegated Administrator is only available for those customers with existing Messaging Server 5.x installations and who are currently installing Messaging Server 6. It is not available to those customers new to the Messaging Server product.

Delegated Administrator must be used with Sun Java System Web Server 6.0 (which is only bundled with the previous Messaging Server 5.2 product). You cannot use Web Server 6.1 (bundled with the Java Enterprise System installer) with Delegated Administrator.

Be sure to read the *Sun Java System Messaging Server Release Notes* (http://docs.sun.com/db/doc/817-6363).

Summary of Installation Steps: To install and configure Delegated Administrator for Messaging with Messaging Server:

NOTE When you install the following products, use the Java Enterprise System installer. Note that some of these products have their own configuration whereas other product configurators are embedded in the Java Enterprise System installer/configurator. For more information, refer to specific product documentation.

1. Be sure that either Sun Java System Directory Server 5.1 or 5.2 is installed and configured.

   For more information, read the appropriate *Sun Java System Directory Server Installation Guide.*

2. Install and configure Messaging Server 6.

   Messaging Server will detect that you are using Sun LDAP Schema 1 since Sun Java System Identity Server will not be installed.

3. Install Sun Java System Web Server 6.0 from your previous Messaging Server 5.2 bundle.

   Review the Sun Java System Web Server documentation and the Sun Java System Delegated Administrator documentation.

4. Install Sun Java System Delegated Administrator for Messaging 1.2 Patch 2. Contact your Sun support representative to obtain the latest version.

   Refer to the Sun Java System Delegated Administrator documentation.

LDAP Provisioning Tools

Sun LDAP Schema 1 users and groups can be provisioned using the LDAP Directory tools (Schema 2 is not supported).

Summary of Installation Steps:

1. If Directory Server is not already installed, be sure to install and configure it.

   For more information, refer to the *Sun Java System Directory Server Installation Guide (*http://docs.sun.com/doc/817-5760).

2. Configure Identity Server to recognize data in your Directory Server.

   Before Identity Server can recognize the data in your LDAP directory, you must add special object classes to entries for all organizations, groups and users that will be managed by Identity Server. If you have not done this already, do it before you start provisioning new accounts. Sample scripts are bundled in the Identity Server product to help you automatically add these object classes to your directory. For more information on these post-installation steps, see the *Sun Java System Identity Server 6.1 Installation and Migration Guide*.

3. Install and configure Messaging Server with help from this guide.

   Messaging Server will detect which Sun Java System LDAP Schema you are using, depending on whether or not Identity Server is installed.

4. Install and configure Sun Java System Web Server 6.1 to enable mail filtering in Messenger Express. For more information on enabling mail filtering, see "Configuring Messenger Express Mail Filters" on page 62. To install Web Server, refer to the *Sun Java Enterprise Installation Guide*.

   Though mail filtering is not a provisioning tool, its functionality existed in the previous GUI version of Delegated Administrator for Messaging.

5. Refer to the Sun Java System Messaging Server documentation to perform LDAP provisioning.

   For Sun LDAP Schema 1 LDAP provisioning, use the *Messaging Server 5.2 Provisioning Guide* and *Sun Java System Communications Services Schema Reference Manual* (The *Sun Java System Schema Reference Manual* contains object classes and attributes for both Sun LDAP Schema 1 and v.2).

User Management Utility

Refer to the *User Management Utility Administration Guide* for installation details.

Post-Installation Directory Layout

After installing the Sun Java System Messaging Server, its directories and files are arranged in the organization described in Table 1-2. The table is not exhaustive; it shows only those directories and files of most interest for typical server administration tasks.

Table 1-2 Post-Installation Directories and Files

Directory	Default Location and Description
Messaging Server Base (*msg_svr_base*)	`/opt/SUNWmsgsr/` (default location) The directory on the Messaging Server machine dedicated to holding the server program, configuration, maintenance, and information files. Note that only one Messaging Server Base directory per machine is permitted.

Table 1-2 Post-Installation Directories and Files *(Continued)*

Directory	Default Location and Description
Configuration `config`	*msg_svr_base*`/config/` (required location)
	Contains all of the Messaging Server configuration files such as the `imta.cnf` and the `msg.conf` files.
	On Solaris platforms only: This directory is symbolically linked to the `config` sub-directory of the data and configuration directory (default: `/var/opt/SUNWmsgsr/`) that you specified in the initial runtime configuration.
Log `log`	*msg_svr_base*`/log/`
	Contains the Messaging Server log files like the `mail.log_current` file.
	On Solaris platforms only: This directory is symbolically linked to the `log` sub-directory of the data and configuration directory (default: `/var/opt/SUNWmsgsr/`) that you specified in the initial runtime configuration.
Data `data`	*msg_svr_base*`/data/` (required location)
	Contains databases, configuration, log files, site-programs, queues, store and message files.
	The `data` directory includes the `config` and `log` directories.
	On Solaris platforms only: This directory is symbolically linked to the data and configuration directory (default: `/var/opt/SUNWmsgsr/`) that you specified in the initial runtime configuration.
System Administrator Programs `sbin`	*msg_svr_base*`/sbin/` (required location)
	Contains the Messaging Server system administrator executable programs and scripts such as `imsimta`, `configutil`, `stop-msg`, `start-msg`, and `uninstaller`.
Library `lib`	*msg_svr_base*`/lib/` (required location)
	Contains shared libraries, private executable programs and scripts, daemons, and non-customizable content data files. For example: `imapd`, `NscpMsg.sh`, and `qm_maint.hlp`.
SDK Include Files `include`	*msg_svr_base*`/include/` (required location)
	Contains Messaging header files for Software Development Kits (SDK).
Examples `examples`	*msg_svr_base*`/examples/` (required location)
	Contains the examples for various SDKs, such as Messenger Express AUTH SDK.

Table 1-2 Post-Installation Directories and Files *(Continued)*

Directory	Default Location and Description
Installation Data `install`	*msg_svr_base*`/install/` (required location) Contains installation-related data files such as installation log files, silent installation files, factory default configuration files, and the initial runtime configuration log files.

Post-Installation Port Numbers

In the installation and initial runtime configuration programs, port numbers will be chosen for various services. These port numbers can be any number from 1 to 65535.

Table 1-3 lists the port numbers that are designated after installation:

Table 1-3 Port Numbers Designated During Installation

Port Number	Service (configutil parameter)
389	Standard Directory Server LDAP Port on the machine where you install Directory Server. This port is specified in the Directory Server installation program. (`local.ugldapport`)
110	Standard POP3 Port. This port may conflict with the MMP port if installed on the same machine. (`service.pop.port`)
143	Standard IMAP4 Port. This port may conflict with the MMP port if installed on the same machine. (`service.imap.port`)
25	Standard SMTP Port. (`service.http.smtpport`)
80	Messenger Express HTTP Port. This port may conflict with the Web Server port if installed on the same machine. (`service.http.port`)
992	POP3 over SSL port. For encrypted communications. (`service.pop.sslport`)
993	IMAP over SSL Port. For encrypted communications. This port may conflict with the MMP port if installed on the same machine. (`service.imap.sslport`)
443	HTTP over SSL Port. For encrypted communications. (`service.http.sslport`)
7997	Messaging and Collaboration ENS (Event Notification Service) Port.
27442	Port that is used Job Controller for internal product communication.

Table 1-3 Port Numbers Designated During Installation *(Continued)*

Port Number	Service (configutil parameter)
49994	Port that is used by the Watcher for internal product communication. See the *Sun Java System Messaging Server Administration Guide* for more information on the Watcher. (`local.watcher.port`)
user-specified	Administration Server HTTP Port. (For listening to Sun ONE Console requests). (`service.http.port`)

If certain products are installed on the same machine, you will encounter port number conflicts. Table 1-4 shows potential port number conflicts:

Table 1-4 Potential Port Number Conflicts

Conflicting Port Number	Port	Port
143	IMAP Server	MMP IMAP Proxy
110	POP3 Server	MMP POP3 Proxy
993	IMAP over SSL	MMP IMAP Proxy with SSL
80	Identity Server (Web Server port)	Messenger Express

If possible, it is recommended that you install products with conflicting port numbers on separate machines. If you are unable to do so, then you will need to change the port number of one of conflicting products.

To change port numbers, use the `configutil` utility. See the *Sun Java System Messaging Server Administration Reference* for complete syntax and usage.

The following example uses the `service.http.port configutil` parameter to change the Messenger Express HTTP port number to `8080`.

```
configutil -o service.http.port -v 8080
```

SMTP Blocking

By default, Messaging Server is 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.

After installation, it is important to manually modify your configuration to match the needs of your site. Specifically, your messaging server should recognize its own internal systems and subnets from which SMTP relaying should always be accepted. If you do not update this configuration, you might encounter problems when testing your MTA configuration.

IMAP and POP clients that attempt to submit messages via the 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. Which systems and subnets are recognized as internal is typically controlled by the `INTERNAL_IP` mapping table, which may be found in the file *msg_svr_base*`/config/mappings`.

For instance, on a Messaging Server system whose IP address is `192.45.67.89`, the default INTERNAL_IP mapping table would appear as follows:

```
INTERNAL_IP

  $(192.45.67.89/24)  $Y
  127.0.0.1  $Y
  *  $N
```

The initial entry, using the `$(IP-pattern/significant-prefix-bits)` syntax, is specifying that any IP address that matches the first 24 bits of `192.45.67.89` 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.

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 192.45.67.0 subnet, then the site would want to modify the initial entry so that the mapping table appears as follows:

```
INTERNAL_IP

  $(192.45.67.89/24)  $Y
  127.0.0.1  $Y
  *  $N
```

Or if the site owns only those IP addresses in the range 192.45.67.80-192.45.67.99, then the site would want to use:

```
INTERNAL_IP

  ! Match IP addresses in the range 192.45.67.80-192.45.67.95
  $(192.45.67.80/28)  $Y
  ! Match IP addresses in the range 192.45.67.96-192.45.67.99
  $(192.45.67.96/30)  $Y
  127.0.0.1  $Y
  *  $N
```

Note that the *msg_svr_base*/sbin/imsimta test -match utility can be useful for checking whether an IP address matches a particular $(... / ...) test condition. The 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 *msg_svr_base*/sbin/imsimta cnbuild and the *msg_svr_base*/sbin/imsimta restart utilities 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 *Sun Java System Messaging Server Administration Reference* (http://docs.sun.com/doc/817-6267). In addition, information on the INTERNAL_IP mapping table can be found in "To Add SMTP Relaying" on page 493."

Enabling Start-up After a Reboot

You can enable Messaging Server start-up after system reboots by using the bootup script: *msg_svr_base*/lib/Sun_MsgSvr. That is, by default, Messaging Server will not restart after a system reboot unless you run this script. In addition, this script can also start up your MMP, if enabled.

To enable Sun_MsgSvr:

1. Copy the Sun_MsgSvr script into the /etc/init.d directory.
2. Change the following ownerships and access modes of the Sun_MsgSvr script:

Table 1-5 Ownership and Access Mode Changes to Sun_MsgSvr

Ownership (chown(1M))	Group Ownership (chgrp(1M))	Access Mode (chmod(1M))
root (superuser)	sys	0744

3. Go to the /etc/rc2.d directory and create the following link:

```
ln /etc/init.d/Sun_MsgSvr S92Sun_MsgSvr
```

4. Go to the /etc/rc0.d directory and create the following link:

```
ln /etc/init.d/Sun_MsgSvr K08Sun_MsgSvr
```

Handling sendmail Clients

If end users send messages through sendmail clients, you can configure Messaging Server to work with those clients over protocol. Users can continue to use the UNIX sendmail client.

To create compatibility between sendmail clients and Messaging Server, you can create and modify a sendmail configuration file.

NOTE Each time a new sendmail patch is applied to your system, you will need to modify the submit.cf file as described in the following instructions for Solaris 8 and Solaris 9.

When you upgraded previous versions of Messaging Server, the /usr/lib/sendmail binary was replaced with a component of the sendmail product. In Messaging Server 6 2004Q2, this replacement during upgrade no longer occurs. Therefore, you need to obtain the proper version of the /usr/lib/sendmail binary from the most current sendmail patch.

Solaris 8

On Solaris 8 operating systems, follow these steps:

1. Find the file main-v7sun.mc file in directory /usr/lib/mail/cf and create a copy of this file.

   In the example in this section, a copy called sunone-msg.mc is created.

2. In the sunone-msg.mc file, add the following lines before the MAILER macros:

```
FEATURE('nullclient', 'smtp:rhino.west.sesta.com')dnl
MASQUERADE_AS('west.sesta.com')dnl
define('confDOMAIN_NAME', 'west.sesta.com')dnl
```

   Note that rhino.west.sesta.com is the localhost name and west.sesta.com is the default email domain as described in "Default Email Domain" on page 47 in To Create the Initial Messaging Server Runtime Configuration. In an HA environment, use the logical host name. See Chapter 3, "Configuring High Availability," for more information about logical host names for High Availability.

3. Compile the sunone-msg.mc file:

```
/usr/ccs/bin/make sunone-msg.cf
```

   The sunone-msg.mc will output sunone-msg.cf.

4. Make a backup copy of the existing sendmail.cf file located in the /etc/mail directory.

   a. Copy and rename /usr/lib/mail/cf/sunone-msg.cf to sendmail.cf file.

   b. Move the new sendmail.cf file to /etc/mail directory.

Solaris 9

On Solaris 9 platforms, sendmail is no longer a setuid program. Instead, it is a a setgid program.

To create the sendmail configuration file on Solaris 9 platforms:

1. Find the file submit.mc file in directory /usr/lib/mail/cf and create a copy of this file.

   In the example in this section, a copy called sunone-submit.mc is created.

2. Change the following line in the file sunone-submit.mc:

```
FEATURE('msp')dn
```

   to

```
FEATURE('msp', 'rhino.west.sesta.com')dnl
```

   where rhino.west.sesta.com is the localhost name.

   Note that rhino.west.sesta.com is the localhost name and west.sesta.com is the default email domain as described in "Default Email Domain" on page 47 in To Create the Initial Messaging Server Runtime Configuration. In an HA environment, use the logical host name. See Chapter 3, "Configuring High Availability," for more information about logical host names for High Availability.

3. Compile the sunone-submit.mc file:

```
/usr/ccs/bin/make sunone-submit.cf
```

   The sunone-submit.mc will output sunone-submit.cf.

4. Make a backup copy of the existing submit.cf file in /etc/mail directory.

   a. Copy and rename /usr/lib/mail/cf/sunone-submit.cf file to submit.cf file.

b. Move the new `submit.cf` file to the `/etc/mail` directory.

Configuring Messenger Express Mail Filters

To install the mail filters, follow these steps:

1. When you installed Messaging Server, the mail filter package (`SUNWmsgmf`) was one of many Messaging Server packages that you installed.

   Verify that the `MailFilter.war` file, which implements management of sieve filters, is in the *msg_svr_base*`/SUNWmsgmf` directory.

2. Be sure that Sun Java System Web Server 6.1 is already installed and configured through the Java Enterprise System installer.

NOTE Web Server needs to be installed on the same system where Messenger Express is configured.

3. Set the environment variable `IWS_SERVER_HOME` to the Web Server installation root directory. For example:

```
setenv IWS_SERVER_HOME webserver_install_root
```

4. Issue the following Web Server command:

```
web_svr_base/bin/https/httpadmin/bin/wdeploy deploy -u /MailFilter -i \
     https-vs_id -v https-vs_id msg_svr_base/SUNWmsgmf/MailFilter.war
```

where *web_svr_base* is the web server root directory, *vs_id* is the virtual server ID of the web server, and *msg_svr_base* is the messaging root directory.

Refer to the Web Server documentation for detailed information on the `wdeploy` command.

When you have completed installing the mail filters, a `MailFilter` directory will be placed in the Web Server's `webapps/https-vs_id/` directory.

5. Use the `configutil` utility to set the following option:

   `local.webmail.sieve.port` = *port*

   where *port* is the Web Server port number.

6. Stop and restart the HTTP daemon:

```
# msg_svr_base/sbin/stop-msg http
# msg_svr_base/sbin/start-msg http
```

   Refer to the *Sun Java System Communications Express Customization Guide* for mail filter usage information.

7. If you want to delete the `MailFilter.war` file in order to install a new version of it, use the following command:

```
web_svr_base/bin/https/httpadmin/bin/wdeploy delete -u /MailFilter
-i https-vs_id -v https-vs_id -n hard
```

   where *web_svr_base* is the web server root directory and *vs_id* is the virtual server ID of the web server.

NOTE With the `-n` option, you have the choice of specifying a `hard` or `soft` value. If you use the `hard` value, it denotes a hard delete and the mail filter is physically removed. The `hard` value should only be used when the new `MailFilter.war` file is available.

Refer to the Web Server documentation for detailed information on the `wdeploy` command.

Performance and Tuning

Please refer to the *Sun Java Messaging Server Deployment Planning Guide* at http://docs.sun.com/doc/817-6440, particularly the section concerning Performance Considerations for a Messaging Server Architecture .

Chapter 2

Upgrading to Sun Java Systems Messaging Server

This chapter describes how to upgrade from Messaging Server 5.2 to Messaging Server 6 2004Q2.

Before You Begin

Prior to performing the upgrade, ensure the following:

- Messaging Server 6 2004Q2 is installed and configured on either the same or a different system from the Messaging Server 5.2 system.

NOTE Unlike previous versions of Messaging Server, you cannot upgrade your existing Messaging Server without first installing and configuring Messaging Server 6 2004Q2.

Also note that you cannot use this upgrade program with Messaging Server versions older than version 5.2. Therefore, you must first migrate or upgrade to Messaging Server 5.2, install Messaging Server 6 2004Q2 and then run this upgrade program. See the iPlanet Messaging Server 5.2 Migration Guide `http://docs.sun.com/source/816-6017-10/index.html` for more information on migrating to Messaging Server 5.2.

- Existing Messaging Server 5.2 installations are configured with MTA Direct LDAP Lookup not with `imsimta dirsync`.

- In addition, Messaging Server 6 2004Q2 does not support multiple instances. If you have multiple instances of Messaging Server version 5.2, you may only choose one instance to upgrade to Messaging Server 6 2004Q2. Furthermore, running the upgrade utility more than once in an attempt to migrate multiple instances will overwrite your configuration.

Overview of the Upgrade Process

There are three steps to upgrading from Messaging Server 5.2 to Messaging Server 6 2004Q2. The following topics outline the process:

1. "Creating Upgrade Files to Update your Configuration" on page 66 (`UpgradeMsg5toMsg6.pl`)
2. "Running the Upgrade Utility" on page 70 (`do_the_upgrade.sh`)
    - MTA Configuration (`make_mta_config_changes.sh`)
    - configutil Parameters (`make_configutil_changes.sh`)
    - Backup Configuration (`make_backup_config_changes.sh`)
    - mboxlist Database (`make_mboxlistdb_changes.sh`)
3. "Migrating User Mailboxes" on page 72 (optional)

Creating Upgrade Files to Update your Configuration

This section describes how special upgrade files are created in order to update your configuration on your Messaging Server 6 2004Q2 system:

- "About Upgrade Files" on page 66
- "Running the UpgradeMsg5toMsg6.pl Perl Script" on page 68

About Upgrade Files

Prior to running an upgrade utility to move from Messaging Server 5.2 to 6, you first need to run the `UpgradeMsg5toMsg6.pl` Perl script (located in *msg_svr_base*`/sbin`).

`UpgradeMsg5toMsg6.pl` compares your 5.2 configuration files with your Messaging Server 6 configuration files and creates two sets of files for each configuration file: `*.CHANGES` files and `*.MERGED` files.

The `*.CHANGES` files and `*.MERGED` files are generated in the work space directory, `/var/tmp/UpgradeMsg5toMsg6.ScratchDir`.

The `*.CHANGES` files show critical configuration file differences between Messaging Server 5.2 and Messaging Server 6 2004Q2. These files highlight the configuration entities that are only found in Messaging Server 6 2004Q2, the configuration entities from Messaging Server 5.2 that are obsolete in Messaging Server 6 2004Q2, and the configuration entities that are only found in the Messaging Server 5.2. Note that not all `*.CHANGES` files will show differences between the versions of configuration files, and not all configuration files will generate `*.CHANGES` files.

The `*.MERGED` files are a consolidation of Messaging Server 5.2 and Messaging Server 6 configuration values and settings. In general, configuration parameter values from Messaging Server 5.2 are retained over the Messaging Server 6 2004Q2 version if:

- There is no default value in the Messaging Server 6 2004Q2 version, or
- The value specified in your 5.2 configuration is not a default setting.

Table 2-1 lists the configuration files that generate `*.MERGED` or `*.CHANGES` files:

Table 2-1 Messaging Server Configuration Files that Generate `*.MERGED` or `*.CHANGES` files

Configuration Information	Description	Generates *.MERGED File	Generates *.CHANGES File
`job_controller.cnf`	Job Controller File	**X**	**X**
conversions	Conversions File	**X**	
channel`_option`, where *channel* is an SMTP channel	SMTP channel option files	**X**	
`native_option`	Native channel option file (exception to *channel*`_option`)	**X**	**X**
channel`_headers.opt`, where *channel* is an SMTP channel	Header option files	**X**	
`dispatcher.cnf`	Dispatcher File	**X**	**X**
`imta_tailor`	Tailor File	**X**	**X**

Table 2-1 Messaging Server Configuration Files that Generate *.MERGED or *.CHANGES files *(Continued)*

Configuration Information	Description	Generates *.MERGED File	Generates *.CHANGES File
option.dat	Global MTA Option File	**X**	**X**
aliases	Aliases File	**X**	
imta.cnf	MTA Configuration File; only the include references (like file directory locations) are changed. Rewrite rule and channel settings are retained from your 5.2 configuration. To include LMTP in your imta.cnf, copy the LMTP information from your Messaging Server 6 imta.cnf file.	**X**	In some instances, a *.CHANGES file may be generated.
mappings	Mappings File	**X**	
mappings.locale	Localized Mappings File	**X**	
internet.rules	Internet Rule Configuration File	**X**	
backup-groups.conf	Backup Group Definitions	**X**	**X**
configutil	Changes of configuration parameters in local.conf and msg.conf configuration files.		**X**

Running the UpgradeMsg5toMsg6.pl Perl Script

To run the UpgradeMsg5toMsg6.pl to create sets of files by which you'll be able to update your configuration, follow these steps:

1. If your Messaging Server 5.2 and 6 versions are not on the same machine, transfer, extract and copy the Messaging Server 5.2 *server-root* directory to the Messaging Server 6 2004Q2 system. If your server versions are installed on the same machine, you can skip this step.

   a. If your Message Store is too large to transfer from one system to another, you can unmount the disks from the 5.2 system and mount them onto the Messaging Server 6 system by using the `umount` (1M) and `mount` (1M) commands.

      You don't have to copy the Messaging Server 5.2 store data to the Messaging Server 6 2004Q2 system, however, you must ensure that the Messaging Server 5.2 store data is accessible during the upgrade process.

   b. Both your 5.2 and Messaging Server 6 2004Q2 systems can be running at this point.

2. Run the `UpgradeMsg5toMsg6.pl` upgrade script (located in *msg_svr_base*`/sbin`) against the `msg-`*instance* of 5.2 version and the *msg_svr_base* of the Messaging Server 6 2004Q2 version. For example:

```
perl UpgradeMsg5toMsg6.pl /usr/sunone/server5/msg-budgie \
  /opt/SUNWmsgsr
```

   where `/usr/sunone/server5/msg-budgie` is the `msg-`*instance* of the 5.2 Messaging Server and `/opt/SUNWmsgsr` is the *msg_svr_base* of the Messaging Server 6 2004Q2.

   `*.MERGED` and `*.CHANGES` files (as described in Table 2-1) will be created.

3. Carefully review the `*.MERGED` files; if you don't want to use the suggested recommendations, you must manually adjust the settings.

This utility does not update the Messenger Express customization files. Therefore, you need to manually change these files in order to keep the relevant information from Messaging Server 5.2 and add any new information from the Messaging Server 6 2004Q2 installation.

Running the Upgrade Utility

This section describes the `do_the_upgrade.sh` utility (located in `/var/tmp/UpgradeMsg5toMsg6.ScratchDir`), a shell script that is made up of four sub-scripts. The following topics are covered in this section:

Overview of the Upgrade Utility

The `do_the_upgrade.sh` utility is made up of four shell scripts that, with your `*.MERGED` files, update the configuration and file directory locations of your MTA configuration, your configutil parameters, backup parameters, and your `mboxlist` database in your Messaging Server 6 2004Q2 system.

You can either run the `do_the_upgrade.sh` utility, or you can individually run one or more of the scripts that make up the `do_the_upgrade.sh` utility (`make_mta_config_changes.sh`, `make_configutil_changes.sh`, `make_backup_config_changes.sh`, and `make_mboxlistdb_changes.sh`).

If you want to upgrade an MTA relay machine from Messaging Server 5.2 to Messaging Server 6 2004Q2, you only need to run the `make_mta_config_changes.sh` and the `make_backup_config_changes.sh` (described in "Backup Configuration" on page 72).

When executing either the `do_the_upgrade.sh` utility or any of the sub-scripts, be sure that neither Messaging Server 5.2 nor 6 2004Q2 is up and running.

Running the do_the_upgrade.sh Utility

To run the `do_the_upgrade.sh` utility:

1. Shut down both the 5.2 and 6 Messaging Servers.
2. Run the utility:

```
# sh /var/tmp/UpgradeMsg5toMsg6.ScratchDir/do_the_upgrade.sh
```

After running the `do_the_upgrade.sh` script, you can either continue to reference your 5.2 partition paths (though you will not be able to remove your Messaging Server 5.2 *server-root* directory) or you can manually move the 5.2 store partitions to the appropriate Messaging Server 6 2004Q2 directory location. You should perform this step prior to restarting Messaging Server.

MTA Configuration

The MTA upgrade configuration sub-script that makes up of part of the `do_the_upgrade.sh` utility is called `make_mta_config_changes.sh` (located in `/var/tmp/UpgradeMsg5toMsg6.ScratchDir`).

The `make_mta_config_changes.sh` script backs up, renames, and moves the `*.MERGED` server configuration files to their original names and locations within the Messaging Server 6 2004Q2 file directory structure.

Once the script has finished renaming and moving the files, it automatically runs the `imsimta cnbuild` command to recompile the MTA configuration.

NOTE If you want to upgrade an MTA relay machine from Messaging Server 5.2 to Messaging Server 6 2004Q2, you only need to run the `make_mta_config_changes.sh` and the `make_backup_config_changes.sh` (described in "Backup Configuration" on page 72).

---

configutil Parameters

The `configutil` upgrade configuration sub-script that makes up part of the `do_the_upgrade.sh` utility is called `make_configutil_changes.sh` script (located in `/var/tmp/UpgradeMsg5toMsg6.ScratchDir`).

The `make_configutil_changes.sh` script incorporates new or updated parameters in the `msg.conf` and `local.conf` files. If default values are not specified in `configutil` parameters in Messaging Server 6 2004Q2, any Messaging Server 5.2 values are carried forward to the Messaging Server 6 2004Q2 version.

Backup Configuration

The backup upgrade configuration sub-script that makes up part of the `do_the_upgrade.sh` utility is called `make_backup_config_changes.sh` script (located in `/var/tmp/UpgradeMsg5toMsg6.ScratchDir`).

The `make_backup_config_changes.sh` script upgrades the configuration of your backup services such as those in your `backup-groups.conf` file.

mboxlist Database

The `mboxlist` database upgrade configuration sub-script that makes up part of the `do_the_upgrade.sh` utility is called `make_mboxlistdb_changes.sh` script (located in `/var/tmp/UpgradeMsg5toMsg6.ScratchDir`).

The `make_mboxlistdb_changes.sh` script transfers and upgrades your 5.2 `mboxlist` database and upgrades it to the Messaging Server 6 2004Q2 directory structure. The script copies the four `*.db` files (`folder.db`, `quota.db`, `peruser.db`, and `subscr.db`) from *server-root*`/msg-`*instance*`/store/mboxlist` on your Messaging Server 5.2 system to *msg_svr_base*`/data/store/mboxlist` on your Messaging Server 6 2004Q2 system.

Migrating User Mailboxes

This section describes how to migrate user mailboxes from Messaging Server 5.2 to your Messaging Server 6 2004Q2 system. If you are upgrading Messaging Server 5.2 to Messaging Server 6 and upgrading the entire message store database, you do not need to follow this procedure. The `make_mboxlistdb_changes.sh` script described in the previous section can be used to upgrade your database more efficiently.

You only need to perform this procedure if:

- You are migrating from Windows to UNIX or from UNIX to Windows.
- You do not want to migrate the entire message store all at once.
- You need to rename your users, including `UIDs`, domain names, and default domain changes.

If you choose to migrate mailboxes using this procedure, do not map the partition paths to the Messaging Server 5.2 partitions and also do not run the `make_mboxlist_changes.sh` script.

The make_configutil_changes.sh script generated by the upgrade script automatically sets the partition path to map to the Messaging Server 5.2 partitions. You need to alter this manually. In addition, you should remove the invocation of the make_mboxlistdb_changes.sh script from your do_the_upgrade.sh script.

To move user mailbox data from Messaging Server 5.2 to Messaging Server 6 2004Q2 in an online method, follow the steps described in this section. You should not need to bring the Messaging Server down while you move data.

The following topics are outlined:

Requirements

The only requirement for migration is that stored should be running on both the old and new messaging servers.

Migration Instructions

To migrate your user mailboxes from your 5.2 system to your Messaging Server 6 2004Q2 system:

1. Notify users in advance that until data move process is completed, they do not have access to their mailboxes. Be sure that users log out of their mail systems before the data move takes place.
2. Change the mailUserStatus user LDAP attribute on all user entries on the 5.2 message store from active to hold in order to hold incoming users' messages in the hold queue and to prevent access to the mailboxes over IMAP, POP, and HTTP.

   For more information on mailUserStatus, see the *Sun Java System Communcations Services Schema Reference Manual* (http://docs.sun.com/doc/817-5702).
3. Ensure that both your 5.2 and 6 2004Q2 Messaging Servers are up and running during this process.

4. Change the `mailHost` attribute in all user entries from the old mail server to the new mail server.

   To do so, use the following `ldapsearch` command to find the user entries where the `mailHost` attribute needs to be modified:

```
ldapsearch -h ldap.siroe.com -b "o=internet" \
   "(&(objectclass=maildomain)(mailHost=oldmail.siroe.com))"
```

   Then, use the `ldapmodify` command to appropriately change the entries to the new mail server. (Use the `ldapmodify` that comes with Messaging and / or Directory Server. Do NOT use the Solaris `ldapmodify`.

   For more information on `mailhost`, see the *Sun Java System Communcations Services Schema Reference Manual*.

5. On the old system, split your user entries into equal groups using the `backup-groups.conf` file. (You can also place the user names into files and use the `-u` option in step 6)

6. Move the user data from the Messaging Server 5.2 message store to the Messaging Server 6 2004Q2 message store.

   This is done by backing up the Messaging Server 5.2 message store with the `imsbackup` utility and restoring the message store to Messaging Server 6 2004Q2 with `imsrestore` utility. For example, to migrate mailboxes from `oldmail.siroe.com` to `newmail.siroe.com`, run the following command on `oldmail.siroe.com`:

```
/<server-root>/bin/msg/store/bin/imsbackup -f-
/<instance>/<group> | rsh newmail.siroe.com
/opt/SUNWmsgsr/lib/msg/imsrestore.sh -f- -cy -v1
```

   You can run multiple concurrent backup and restore sessions (one per group) to maximize the transfer rate into the new message store. See the *Messaging Server Reference Manual* (`http://docs.sun.com/doc/817-6267`)for more information about the `imsbackup` and `imsrestore` utilities.

7. Set Messaging Server 6 2004Q2 to be the new default messaging server for the system.

   Change the `A` record of `oldmail.siroe.com` to point to `newmail.siroe.com` (the server responsible for domain(s) previously hosted on `oldmail.siroe.com`).

8. Release the messages in the hold queue on your Messaging Server 5.2 system by issuing the following command:

```
imsimta process_held -uid=user -domain=domain
```

   where *user* is the user ID and *domain* is the domain where the user resides.

9. Ensure that the user clients are pointing to the new mail server.

   After the upgrade finishes, have the users point to the new server through their mail client program (in this example, have it point to `newmail.siroe.com` from `oldmail.siroe.com`).

   An alternative is to use an MMP which obviates the need to have users point their clients directly at the new mail server; the MMP will get that information from the `mailHost` attribute which is stored in the LDAP user entries and will automatically re-direct them to the new server.

Chapter 3

Configuring High Availability

This section provides the information you need to configure the Veritas Cluster Server or Sun Cluster high availability clustering software and prepare it for use with the Messaging Server. It is assumed you have read the chapter on High Availability in the *Messaging Server Deployment Planning Guide* (http://docs.sun.com/doc/817-6440) as well as the appropriate Veritas or Sun Cluster Server documentation for detailed planning, installation instructions, required patches, and other information as needed.

Table 3-1 lists the versions of Sun Cluster Server and Veritas Cluster Server that are currently supported with Messaging Server:

Table 3-1 Supported Versions of Sun Cluster Server and Veritas Cluster Server

Cluster	Supported Versions
Sun Cluster Server	Sun Cluster 3.1
Veritas Cluster Server	Veritas Cluster Server 1.3, Veritas Cluster Server 2.0, and Veritas Cluster Server 3.5

This chapter consists of the following sections:

Cluster Agent Installation

A cluster agent is a Messaging Server program that runs under the cluster framework.

The Sun Cluster Messaging Server agent (SUNWscims) is installed when you select Sun Cluster 3.1 through the Java Enterprise System installer. The Veritas Cluster Messaging Server agent (SUNWmsgvc) can be found in the Messaging Server Product subdirectory on the Java Enterprise System CD, Solaris_sparc/Product/messaging_svr/Packages/SUNWmsgvc. (Note that you must use the pkgadd(1M) command to install the VCS cluster agent.)

Some items of note regarding the Messaging Server and high availability (applies to both Veritas Cluster, and Sun Cluster) installation:

- Clustering software needs to be installed before installing and configuring Messaging Server. Run the installation on the cluster node currently pointed at by the HA logical host name for Messaging Server.
- After running the Messaging Server Initial Runtime Configuration (see "To Create the Initial Messaging Server Runtime Configuration" on page 43), be sure to manually configure the fully-qualified HA logical host name of the cluster of Messaging Server.

Using the useconfig utility

The useconfig utility allows you to share a single configuration between multiple nodes in an HA environment. This utility is not meant to upgrade or update an existing configuration.

For example, if you are upgrading your first node, you will install through the Java Enterprise System installer and then configure Messaging Server. You will then failover to the second node where you will install the Messaging Server package through the Java Enterprise System installer, but you will not have to run the Initial Runtime Configuration Program (configure) again. Instead, you can use the useconfig utility.

To enable the utility, run useconfig utility to point to your previous Messaging Server configuration.

```
msg_svr_base/sbin/useconfig install/configure_YYYYMMDDHHMMSS
```

where configure_*YYYYMMDDHHMMSS* is your previous configuration settings file.

On a brand new node, you can find the configure_*YYYYMMDDHHMMSS* in the *msg_svr_base*/data/setup directory on the shared disk.

The following sections on "Veritas Cluster Server Agent Installation" on page 79 and "Sun Cluster Agent Installation" on page 83 describe when you can use the `useconfig` utility.

Veritas Cluster Server Agent Installation

Messaging Server can be configured with Veritas Cluster Server 1.3, 2.0, and 3.5. The instructions in this section only cover Veritas Cluster 3.5; for Veritas 1.3 and 2.0, review the *Messaging Server 5.2 Installation Guide*.

Be sure to review the Veritas Cluster Server documentation prior to following these procedures.

NOTE

- Veritas Volume Manager (VxVM) has a cluster feature that requires a separate license. This feature provides a global view of the file systems on shared storage, similar to the Sun Cluster 3.0 global file system. See the Veritas Cluster Server documentation for more information.
- `FsckOpt` was optional in pre-3.5 Veritas releases. However, it is required for configuring the `Mount` resource. `FsckOpt` must include a `-y` or `-n`, otherwise the resource will not come online.
- Veritas Cluster Server 2.0 Explorer cannot be used to manage Veritas Cluster Server 3.5.

After installing Messaging Server through the Java Enterprise System installer and configuring HA, be sure to review "Binding IP Addresses on a Server" on page 88 for additional steps associated with configuring HA support.

Veritas Cluster Server Requirements

- Veritas Cluster Software is already installed and configured.
- As described in the following instructions (in "VCS 3.5 Installation and Configuration Notes" on page 79), you will install the Veritas Cluster Agent package for Messaging Server along with the Messaging Server software on both nodes.

VCS 3.5 Installation and Configuration Notes

The following instructions describe how to configure Messaging Server as an HA service, by using Veritas Cluster Server 3.5.

The default main.cf configuration file sets up a resource group called ClusterService that launches the VCSweb application. This group includes network logical host IP resources like csgnic and webip. In addition, the ntfr resource is created for event notification.

1. Launch Cluster Explorer from one of the nodes.

   Note that these Veritas Cluster Server instructions assume you are using the graphical user interface to configure Messaging Server as an HA service.

   To launch Cluster Explorer, run the following command:

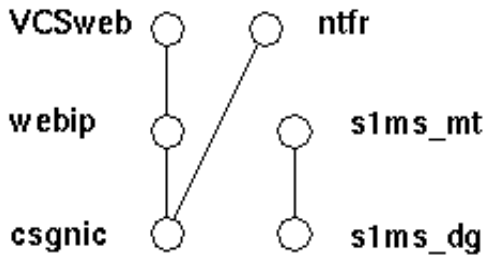
```
# /opt/VRTSvcs/bin/hagui
```

   The VRTScscm package must be installed in order to use the GUI.

2. Add s1ms_dg disk group resource of type DiskGroup and enable it.
3. Add s1ms_mt mount resource of type Mount.
   a. Unlike in Veritas Cluster Server 2.0, you must add -y (or -n) to FsckOpt. Null options cause Mount to hang. See the Solaris Man page for more information on fsck_vxfs.
   b. Be sure to click the Link button to enable linking resources, if they are not already enabled.
4. Create a link between s1ms_mt and s1ms_dg. Enable the resource s1ms_mt.

   Figure depicts the dependency tree:

Figure 3-1 Veritas Cluster Server Dependency Tree



5. Run the Java Enterprise System installer, selecting Administration Server and Messaging Server.

   a. During Administration Server configuration, be sure to specify the logical host name when asked to provide a hostname.

   b. Run the Messaging Server Initial Runtime Configuration from the primary node (for example, Node_A) to install Messaging Server.

   c. Install the Veritas Cluster Server agent package, `SUNWmsgvc`, (located in the Messaging Server `Product` subdirectory on the Java Enterprise System CD) by using the `pkgadd`(1M) command.

      Messaging Server and the Veritas agent are now installed on Node_A.

6. Switch to the backup node (for example, Node_B).

7. Run the Java Enterprise System installer to install Messaging Server on the backup node (Node_B).

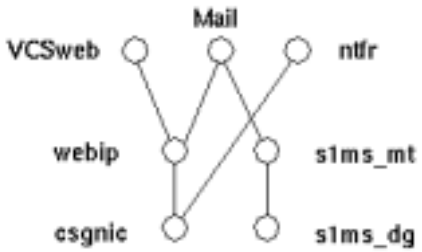
8. After installing Messaging Server, you can use the `useconfig` utility to obviate the need for creating an additional initial runtime configuration on the backup node (Node_B). The `useconfig` utility allows you to share a single configuration between multiple nodes in an HA environment. This utility is not meant to upgrade or update an existing configuration. See "Using the useconfig utility" on page 78.

   The Veritas agent is now installed on Node_B.

9. From the Cluster Explorer, Select Import Types... from the File menu which will display a file selection box.

10. Import the `MsgSrvTypes.cf` type from the `/etc/VRTSvcs/conf/config` directory. Import this type file. Note that you need to be on a cluster node to find this file.
11. Now create a resource of type `MsgSrv` (for example, `Mail`). This resource requires the logical host name property to be set.
12. The `Mail` resource depends on `s1ms_mt` and `webip`. Create links between the resources as shown in the following dependency tree:

Figure 3-2 Veritas Cluster Dependency Tree



   a. Enable all resources and bring `Mail` online.
   b. All servers should be started.
13. Switch over to Node_A and check if the High Availability configuration is working.
14. Change the group attribute `OnlineRetryLimit` from `3` to `0`, otherwise the failed-over service might restart on the same node.

MsgSrv Attributes

This section describes `MsgSrv` additional attributes that govern the behavior of the `mail` resource. To configure Messaging Server with Veritas Cluster Server, see Table 3-2.

Table 3-2 Veritas Cluster Server Attributes

Attribute	Description
`FaultOnMonitorTimeouts`	If unset (=0), monitor (probe) time outs are not treated as resource fault. Recommend setting this to 2. If the monitor times out twice, the resource will be restarted or failed over.

Table 3-2 Veritas Cluster Server Attributes *(Continued)*

Attribute	Description
`ConfInterval`	Time interval over which faults / restarts are counted. Previous history is erased if the service remains online for this duration. Suggest 600 seconds.
`ToleranceLimit`	Number of times the monitor should return OFFLINE for declaring the resource FAULTED. Recommend leaving this value at '0' (default).

Sun Cluster Agent Installation

This section describes how to install and configure the Messaging Server as a Sun Cluster Highly Available (HA) Data Service. These installation instructions apply to Sun Cluster 3.1. The following topics are covered in this section:

- "Sun Cluster Requirements" on page 83
- "About HAStoragePlus" on page 84
- "Configuring Messaging Server with Sun Cluster and HA StoragePlus" on page 84
- "Binding IP Addresses on a Server" on page 88

Documentation for Sun Cluster 3.1 can be found at:

`http://docs.sun.com/db/prod/cluster#hic`

Note that Veritas File System (VxFS) is supported with Sun Cluster 3.1.

Sun Cluster Requirements

This section presumes the following:

- Sun Cluster 3.1 is installed and configured on a Solaris 8 or 9 operating system with required patches.
- The Sun Cluster agent `SUNWscims` is installed on your systems.
- If logical volumes are being created, either Solstice DiskSuite or Veritas Volume Manager is used.

About HAStoragePlus

It is highly recommended that you use the HAStoragePlus resource type to make locally mounted file systems highly available within a Sun Cluster environment. Any file system resident on a Sun Cluster global device group can be used with HAStoragePlus. Unlike a globally mounted file system like HAStorage, HAStoragePlus is available only on one cluster node at any given point of time. These locally mounted file systems can only be used in failover mode and in failover resource groups. HAStoragePlus offers FFS (failover file system), in contrast to HAStorage's GFS (global file system).

HAStoragePlus has a number of benefits:

- HAStoragePlus bypasses the global file service layer completely. For disk-IO intensive data services, this leads to a significant performance increase.
- HAStoragePlus can work with any file system (like UFS, VxFS, and so forth), even those that might not work with the global file service layer. If a file system is supported by the Solaris operating system, it will work with HAStoragePlus.

For more information on HAStoragePlus, read the *Sun Cluster 3.1 Data Service Planning and Administration Guide*.

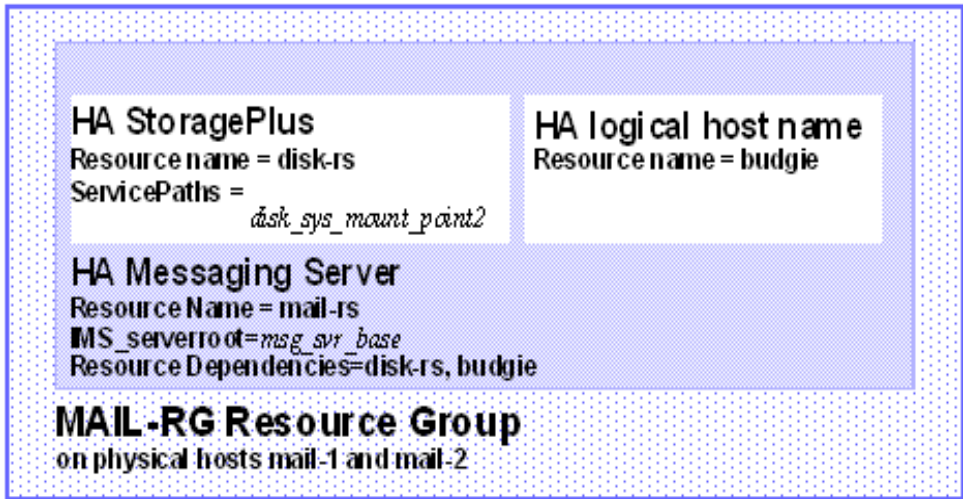
Configuring Messaging Server with Sun Cluster and HA StoragePlus

This section describes how to configure HA support and HA StoragePlus for Sun ONE Messaging Server for Sun Cluster 3.1 through a simple example.

After configuring HA, be sure to review "Binding IP Addresses on a Server" on page 88 for additional steps associated with HA support.

The following example assumes that the messaging server has been configured with a HA logical host name and IP address. The physical host names is assumed to be `mail-1` and `mail-2`, with an HA logical host name of `budgie`. Figure 3-3 depicts the nested dependencies of the different HA resources you will create in configuring Messaging Server HA support.

Figure 3-3 A Simple Sun ONE Messaging Server HA configuration



1. Become the superuser and open a console.

   All of the following Sun Cluster commands require that you have logged in as superuser. You will also want to have a console or window for viewing messages output to `/dev/console`.

2. Add required resource types.

   Configure Sun Cluster to know about the resources types we will be using. This is done with the `scrgadm -a -t` command:

   ```
   # scrgadm -a -t SUNW.HAStoragePlus
   # scrgadm -a -t SUNW.ims
   ```

3. Create a resource group for the Messaging Server.

   If you have not done so already, create a resource group and make it visible on the cluster nodes which will run the Messaging Server. The following command creates a resource group named `MAIL-RG`, making it visible on the cluster nodes `mail-1` and `mail-2`:

   ```
   # scrgadm -a -g MAIL-RG -h mail-1,mail-2
   ```

   You may, of course, use whatever name you wish for the resource group.

4. Create an HA logical host name resource and start resource group.

   If you have not done so already, create and enable a resource for the HA logical host name, placing it in the resource group. The following command does so using the logical host name `budgie`. Since the `-j` switch is omitted, the name of the resource created will also be `budgie`.

   ```
   # scrgadm -a -L -g MAIL-RG -l budgie
   # scswitch -Z -g MAIL-RG
   ```

5. Create an HAStoragePlus resource.

   Next, you need to create an HAStoragePlus resource type for the file systems on which Messaging Server is dependent. The following command creates an HAStoragePlus resource named `disk-rs` and the file system *disk_sys_mount_point* is placed under its control:

   ```
   # scrgadm -a -j disk-rs -g MAIL-RG \
   -t SUNW.HAStoragePlus \
   -x ServicePaths=disk_sys_mount_point-1, disk_sys_mount_point-2
   ```

   The comma-separated list of `ServicePaths` are the mount points of the cluster file systems on which Messaging Server is dependent. In the above example, only two mount points, *disk_sys_mount_point-1* and *disk_sys_mount_point-2*, are specified. If one of the servers has additional file systems on which it is dependent, then you can create an additional HA storage resource and in Step 9 to indicate that additional dependency.

6. Install and configure the Administration Server (See the Sun Java Enterprise System 2004Q2 Installation Guide).

   When specifying the fully qualified domain name, use the HA logical host name created in Step 4.

7. Install and configure the Messaging Server. See “To Create the Initial Messaging Server Runtime Configuration” on page 43.

   a. In the Initial Runtime Configuration, you are asked to specify a configuration directory in “To Create the Initial Messaging Server Runtime Configuration” on page 43. Be sure to use the shared disk directory path of your HAStoragePlus resource.

b. Run the following command to enable the watcher process under Sun Cluster:

```
configutil -o local.autorestart -v 1
```

For more information on the watcher process, refer to the *Sun Java System Messaging Server Administration Guide*.

8. Run the `ha_ip_config` script to set `service.listenaddr` and `service.http.smtphost` and to configure the `dispatcher.cnf` and `job_controller.cnf` files for high availability. The script will ensure that the logical IP address is set for these parameters and files, rather than the physical IP address.

For instructions on running the script, see "Binding IP Addresses on a Server" on page 88.

The `ha_ip_config` script should only be run once on the machine with the shared disk (for configuration and data).

9. Create an HA Messaging Server resource.

It's now time to create the HA Messaging Server resource and add it to the resource group. This resource is dependent upon the HA logical host name and HA disk resource.

In creating the HA Messaging Server resource, we need to indicate the path to the Messaging Server top-level directory—the *msg_svr_base* path. These are done with the `IMS_serverroot` extension properties as shown in the following command.

```
# scrgadm -a -j mail-rs -t SUNW.ims -g MAIL-RG \
          -x IMS_serverroot=msg_svr_base \
          -y Resource_dependencies=disk-rs,budgie
```

The above command, creates an HA Messaging Server resource named `mail-rs` for the Messaging Server which is installed on `IMS_serverroot` in the *msg_svr_base* directory. The HA Messaging Server resource is dependent upon the HA disk resource `disk-rs` as well as the HA logical host name `budgie`.

If the Messaging Server has additional file system dependencies, then you can create an additional HA storage resource for those file systems. Be sure to include that additional HA storage resource name in the `Resource_dependencies` option of the above command.

10. Remove the term `global` from the `/etc/vfstab` file. At bootup, `/etc/vbstab` must be set to 'no..' For more information, refer to your Sun Cluster 3.1 documentation.

    Before the `vfstab` file is enabled with HAStoragePlus, you might first `umount` the file systems that are currently global file systems. You can then enable the `vfstab` file with HAStoragePlus and remount the file systems.

11. Enable the Messaging Server resource.

    It's now time to activate the HA Messaging Server resource, thereby bringing the messaging server online. To do this, use the command

    ```
    # scswitch -e -j mail-rs
    ```

    The above command enables the `mail-rs` resource of the MAIL-RG resource group. Since the MAIL-RG resource was previously brought online, the above command also brings `mail-rs` online.

12. Verify that things are working.

    Use the `scstat` command to see if the MAIL-RG resource group is online. You may want to look at the output directed to the console device for any diagnostic information. Also look in the `syslog` file, `/var/adm/messages`.

13. Fail the resource group over to another cluster node in order to make sure failover properly works.

    Manually fail the resource group over to another cluster node. (Be sure you have superuser privileges on the node to which you failover.)

    Use the `scstat` command to see what node the resource group is currently running on ("online" on). For instance, if it is online on mail-1, then fail it over to mail-2 with the command:

    ```
    # scswitch -z -g MAIL-RG -h mail-2
    ```

Binding IP Addresses on a Server

If you are using the Symmetric or N + 1 high availability models, there are some additional things you should be aware of during configuration in order to prepare the Sun Cluster Server for Messaging Server.

Messaging Server running on a server requires that the correct IP address binds it. This is required for proper configuration of Messaging in an HA environment.

Part of configuring Messaging Server for HA involves configuring the interface address on which the Messaging Servers bind and listen for connections. By default, the servers bind to all available interface addresses. However, in an HA environment, you want the servers to bind specifically to the interface address associated with an HA logical host name.

A script is therefore provided to configure the interface address used by the servers belonging to a given Messaging Server instance. Note that the script identifies the interface address by means of the IP address which you have or will be associating with the HA logical host name used by the servers.

The script effects the configuration changes by modifying or creating the following configuration files. For the file

msg_svr_base/config/dispatcher.cnf

it adds or changes INTERFACE_ADDRESS option for the SMTP and SMTP Submit servers. For the file

msg_svr_base/config/job_controller.cnf

it adds or changes the INTERFACE_ADDRESS option for the Job Controller.

Finally it sets the configutil service.listenaddr and service.http.smtphost parameters used by the POP, IMAP, and Messenger Express HTTP servers.

Note that the original configuration files, if any, are renamed to *.pre-ha.

Run the script as follows:

1. Become superuser.
2. Execute *msg_svr_base*/sbin/ha_ip_config
3. The script presents the questions described below. The script may be aborted by typing control-d in response to any of the questions. Default answers to the questions will appear within square brackets, [ ]. To accept the default answer, simply press the RETURN key.

    a. Logical IP address: Specify the IP address assigned to the logical host name which the Messaging Server will be using. The IP address must be specified in dotted decimal form, for example, 123.456.78.90.

    The logical IP address is automatically set in the configutil parameter service.http.smtphost which allows you to see which machine is running your messaging system in a cluster. For example, if you are using Messenger Express, your server will be able to determine from which mail host to send outgoing mail.

b. Messaging Server Base (*msg_svr_base*): Specify the absolute path to the top-level directory in which Messaging Server is installed.

c. Do you wish to change any of the above choices: answer “no” to accept your answers and effect the configuration change. Answer “yes” if you wish to alter your answers.

NOTE In addition, the `ha_ip_config` script automatically enables two new processes `watcher` and `msprobe` with the following parameters: `local.autorestart` and `local.watcher.enable`. These new parameters help to monitor the health of the messaging server. Process failures and unresponsive services result in log messages indicating specific failures. The cluster agents now monitor the `watcher` process and failover whenever it exits. Note that the parameters must be enabled in order for Sun Cluster to function properly.

For more information on the `watcher` and `msprobe` processes, see “Automatic Restart of Failed or Unresponsive Services” on page 100.

Unconfiguring High Availability

This section describes how to unconfigure high availability. To uninstall high availability, follow the instructions in your Veritas or Sun Cluster documentation.

The High Availability unconfiguration instructions differ depending on whether you are removing Veritas Cluster Server or Sun Cluster.

The following topics are covered in this section:

Unconfiguring Veritas Cluster Server

To unconfigure the high availability components for Veritas Cluster Server:

1. Bring the `iMS5` service group offline and disable its resources.
2. Remove the dependencies between the `mail` resource, the `logical_IP` resource, and the `mountshared` resource.

3. Bring the iMS5 service group back online so the sharedg resource is available.
4. Delete all of the Veritas Cluster Server resources created during installation.
5. Stop the Veritas Cluster Server and remove following files on both nodes:

   ```
   /etc/VRTSvcs/conf/config/MsgSrvTypes.cf
   /opt/VRTSvcs/bin/MsgSrv/online
   /opt/VRTSvcs/bin/MsgSrv/offline
   /opt/VRTSvcs/bin/MsgSrv/clean
   /opt/VRTSvcs/bin/MsgSrv/monitor
   /opt/VRTSvcs/bin/MsgSrv/sub.pl
   ```

6. Remove the Messaging Server entries from the /etc/VRTSvcs/conf/config/main.cf file on both nodes.
7. Remove the /opt/VRTSvcs/bin/MsgSrv/ directory from both nodes.

Unconfiguring Messaging Server HA Support for Sun Cluster 3.x

This section describes how to undo the HA configuration for Sun Cluster. This section assumes the simple example configuration (described in the "Sun Cluster Agent Installation" on page 83). For other configurations, the specific commands (for example, Step 3) may be different but will otherwise follow the same logical order.

1. Become the superuser.

   All of the following Sun Cluster commands require that you be running as user superuser.

2. Bring the resource group offline.

   To shut down all of the resources in the resource group, issue the command

   ```
   # scswitch -F -g MAIL-RG
   ```

   This shuts down all resources within the resource group (for example, the Messaging Server and the HA logical host name).

3. Disable the individual resources.

   Next, remove the resources one-by-one from the resource group with the commands:

   ```
   # scswitch -n -j mail-rs
   # scswitch -n -j disk-rs
   # scswitch -n -j budgie
   ```

4. Remove the individual resources from the resource group.

   Once the resources have been disabled, you may remove them one-by-one from the resource group with the commands:

   ```
   # scrgadm -r -j mail-rs
   # scrgadm -r -j disk-rs
   # scrgadm -r -j budgie
   ```

5. Remove the resource group.

   Once the all the resources have been removed from the resource group, the resource group itself may be removed with the command:

   ```
   # scrgadm -r -g MAIL-RG
   ```

6. Remove the resource types (optional).

   Should you need to remove the resource types from the cluster, issue the commands:

   ```
   # scrgadm -r -t SUNW.ims
   # scrgadm -r -t SUNW.HAStoragePlus
   ```

Chapter 4

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 Sun ONE Server 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:

To Modify Your Passwords

Because you set up a number of administrators with the same password in during initial configuration (see "To Create the Initial Messaging Server Runtime Configuration" on page 43), you might want to change the passwords of those administrators.

Refer to Table 4-1, which shows the parameters where default passwords are set up during initial runtime configuration and the utilities you can use to change them. For those parameters that use the `configutil` utility to change for complete syntax and usage.

Table 4-1 Passwords Set in Messaging Server Initial Runtime Configuration

Parameter	Description
`local.ugldapbindcred`	Password for the user/group administrator set through the `configutil` utility.
`local.service.pab.ldappasswd`	Password for user specified by Bind DN for PAB searches set through the `configutil` utility.
SSL passwords for key files	Passwords that are directly set in the `sslpassword.conf` file.
Service Administrator Credentials	These are credentials that are directly set in your LDAP Directory (with the `ldapmodify` command).
Service Administrator for Delegated Administrator	You will only need to change the password of this administrator if you have enabled Sun LDAP Schema 1 and you are using the Delegated Administrator utility. To change the password of the Delegated Administrator Service Administrator, you can do so in the Sun ONE Console, your LDAP Directory (with the `ldapmodify` command), or the Delegated Administrator UI.
Store Administrator	To change the password of the Store Administrator, you can do so in either the Sun ONE Console or in your LDAP Directory (with the `ldapmodify` command).

The following example uses the `local.enduseradmincred configutil` parameter to change the password of the end user administrator.

```
configutil -o local.enduseradmincred -v newpassword
```

Managing Mail Users, Mailing Lists and Domains

All user, mailing list and domain 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 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 User Management Utility for Sun LDAP Schema 2 and the iPlanet Delegated Administrator for Messaging, Delegated Administrator command line utilities, or by directly modifying the LDAP directory for Sun LDAP Schema 1.

To Remove a User from Messaging Server

1. Mark the user as deleted by running the `commadmin user delete` command. (See *Sun Java System Communications Services 6 2004Q2User Management Utility Admin Guide* at `http://docs.sun.com/doc/817-5703`.

2. Remove resources from the user.

   A resource can be a mailbox or a calendar. For Messaging Server, the program is called `msuserpurge`. (See *Sun Java System Messaging Server Administration Reference* at `http://docs.sun.com/doc/817-6267`.) For calendar services, the program is `csclean`. (See S*un Java System Calendar Server Administration Guide* `http://docs.sun.com/doc/817-5697`.)

3. Permanently remove the user, by invoking the `commadmin domain purge` command.

To Remove a Domain from Messaging Server

1. Mark the domain as deleted by running the `commadmin domain delete` command. (See *Sun Java System Communications Services 6 2004Q2User Management Utility Admin Guide* at `http://docs.sun.com/doc/817-5703`.)
2. Remove resources from the users of that domain.

   A resource can be a mailbox or a calendar. For Messaging Server, the program is called `msuserpurge`. (See *Sun Java System Messaging Server Administration Reference* at `http://docs.sun.com/doc/817-6267`.) For calendar services, the program is `csclean`. (See S*un Java System Calendar Server Administration Guide* `http://docs.sun.com/doc/817-5697`.)
3. Permanently remove the domain, by invoking the `commadmin domain purge` command.

Managing Messaging Server with Sun ONE Console

When the messaging server installation process and initial runtime configuration program completes, you can start your Messaging Server through the Admin Console. If your directory and messaging server reside on a single machine, you can use the Console interface to manage both servers.

To invoke the console, run the `/usr/sbin/mpsconsole` command.

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

To display the Information form:

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

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

Starting and Stopping Services

Services are started and stopped differently depending on whether they are installed in an HA environment or 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. If you attempt a `stop-msg` in an HA deployment, the system warns that it has detected an HA setup and will tell you how to properly stop the system.

The appropriate start, stop and restart commands are shown in the tables below. Note that there are no specific HA commands to individually start, restart, or stop other Messaging Server services (for example, SMTP). However, you can run a `stop-msg` *service* command to stop / restart individual servers such as `imap`, `pop` or `sched`.

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 4-2 Start, Stop, Restart in a Sun Cluster 3.0 / 3.1 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 4-3 Start, Stop, Restart in Veritas 1.3, 2.0, 2.1, and 3.5 Environments

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. In addition, you only need to run the services that your server actually uses. For example, if you are using 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 services such as POP, IMAP, and HTTP, before starting or stopping them. For more information, see “Enabling and Disabling Services” on page 114.

Important: If a server process crashes, other processes may hang as they wait for locks held by the server process that crashed. If you are not using automatic restart (see “Automatic Restart of Failed or Unresponsive Services” on page 100), and 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. 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 (`smtp`, `imap`, `pop`, `store`, `http`, `ens`, `sched`), as shown in the following example:

msg_svr_base`/sbin/start-msg imap`
msg_svr_base`/sbin/stop-msg pop`
msg_svr_base`/sbin/stop-msg sched`
msg_svr_base`/sbin/stop-msg smtp`

Note that the services must be enabled in order to stop or start them. See "To Specify What Services are Started" on page 100.

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, you can use the start/stop msg command for the dispatcher and the job controller. For more information, see the *Messaging Server Reference Manual*.

To Specify What Services are Started

By default the following services are started with `start-msg`:

```
# ./start-msg
Connecting to watcher ...
Launching watcher ...
Starting ens server .... 21132
Starting store server .... 21133
checking store server status ... ready
Starting imap server .... 21135
Starting pop server .... 21138
Starting http server .... 21141
Starting sched server .... 21143
Starting dispatcher server .... 21144
Starting job_controller server .... 21146
```

These can be controlled by enabling or disabling the `configutil` parameters: `service.imap.enable`, `service.pop.enable`, `service.http.enable`, `local.smsgateway.enable`, `local.snmp.enable`, `local.imta.enable`, `local.mmp.enable`, `local.ens.enable`, and `local.sched.enable`. Note that you need to set both `service.imap.enable` and `service.imap.enablesslport` to `0` in order to disable IMAP. The same goes for POP and HTTP. See the *Sun Java System Messaging Server Administration Reference* for details on how these work.

Automatic Restart of Failed or Unresponsive Services

Messaging Server provides two processes called `watcher` and `msprobe` that transparently monitor services and automatically restart them if they fail or become unresponsive (the services hang or freeze up). `watcher` monitors server failures. `msprobe` monitors server response time. When a server fails or stops responding to requests, it is automatically restarted. Table 4-4.

Table 4-4 Services Monitored by `watcher` and `msprobe`

watcher	msprobe
IMAP, POP, HTTP, job controller, dispatcher, message store (`stored`, utilities), `imsched`, MMP. (LMTP/SMTP servers are monitored by the dispatcher and LMTP/SMTP clients are monitored by the job_controller.)	IMAP, POP, HTTP, job controller, message store (`stored`, utilities), `imsched`, ENS, LMTP, SMTP

Setting `local.watcher.enable=on` (default) will monitor process failures and unresponsive services and will log error messages to the default log file indicating specific failures. To enable automatic server restart, set the `configutil` parameter `local.autorestart` to `yes`. By default, this parameter is set to `no`.

If any of the message store services fail or freeze, all message store services that were enabled at start-up are restarted.For example, if `imapd` fails, at the least, `stored` and `imapd` are restarted. If other message store services were running, such as the POP or HTTP servers, then those will be restarted as well, whether or not they failed.

Automatic restart also works if a message store utility fails or freezes. For example, if `mboxutil` fails or freezes, the system will automatically restart all the message store servers. Note, however, that it will not restart the utility. `msprobe` runs every 10 minutes. Service and process restarts will be performed up to two times within a 10 minute period (configurable using `local.autorestart.timeout`).

Whether or not `local.autorestart` is set to `yes`, the system still monitors the services and sends failure or non-response error messages to the console and *msg_svr_base*`/data/log`. `watcher` listens to port 49994 by default, but this is configurable with `local.watcher.port`.

NOTE A new watcher log file is generated in *msg_svr_base*`/data/log/watcher`. This log file is not managed by the logging system (no rollover or purging).

Automatic Restart in High Availability Deployments

Automatic restart in high availability deployments require the following `configutil` parameters to be set:

Table 4-5 HA Automatic Restart Parameters

Parameter	Description/HA Value
`local.watcher.enable`	Enable watcher. On (Default is On)
`local.autorestart`	Enable autorestart. On
`local.autorestart.timeout`	Failure retry time-out. If a server fails more than twice in this designated amount of time, then the system stops trying to restart the server. If this happens on an HA system, Messaging Server is shutdown and a failover to the other system occurs. The value (set in seconds) should be set to a period value longer than the `msprobe` interval (`local.schedule.msprobe`).
`local.schedule.msprobe`	`msprobe` run schedule. A crontab style schedule string (see Table 18-10 on page 552). Default is 600 seconds.

To Schedule Automatic Tasks

Messaging Server provides a general task scheduling mechanism using a process called `imsched`. It is enabled by setting the `local.schedule.`*taskname* `configutil` parameter.

This parameter requires a command and a schedule on which to execute the command. The format is as follows:

`configutil -o local.schedule.`*taskname* `-v "`*schedule*`"`

taskname is a unique name for this command / schedule combination.

schedule has the format:

minute hour day-of-month month-of-year day-of-week command args

command args can be any Messaging Server command and its arguments. A fully qualified command pathname is required.

minute hour day-of-month month-of-year day-of-week is the schedule for running the command. It follows the UNIX `crontab` format.

The values are separated by a space or tab and can be 0-59, 0-23, 1-31, 1-12 or 0-6 (with 0=Sunday) respectively. Each time field can be either an asterisk (meaning all legal values), a list of comma-separated values, or a range of two values separated by a hyphen. Note that days can be specified by both day of the month and day of the week and both will be required if specified. For example, setting the 17th day of the month and Tuesday will only run the command on the 17th day of a month when it is Tuesday. See Table 18-10 on page 552 for examples of how to set the schedule parameter.

Note that if you modify scheduler, you must either restart the scheduler with the command `stop-msg sched` and `start-msg sched`, or you can send SIGHUP to the scheduler process:

`kill -HUP` *scheduler_pid*

Scheduler Examples

Run `imexpire` in verbose mode at 12:30am, 8:30am, and 4:30pm:

```
configutil -o local.schedule.rm_messages -v 30 0,8,16 * * *
/opt/SUNWmsgsr/sbin/imexpire -v
```

Display MTA channel queue message counters every 20 minutes:

```
configutil -o local.schedule.counters -v 20,40,60 * * * *
/opt/SUNWmsgsr/sbin/imsimta qm counters -show > temp.txt
```

Run `imsbackup` Monday through Friday at midnight (12AM):

```
configutil -o local.schedule.msbackup -v 0 0 * * 1-5
/opt/SUNWmsgsr/sbin/imsbackup -f backupfile /primary
```

To Configure a Greeting Message

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

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

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

4. Create a new-user greeting or make changes, as needed.

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

   When you create a message, specify its language with the drop-down list above the message field. You can create several messages in several languages, if desired.

5. Click Save.

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

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

Where *Message* must contain a header (with at least a subject line), followed by $$, then the message body. The message must contain a header (with at least a subject line), followed by $$, then the message body. The $ represents a new line.

For example, to enable this parameter, you can set the following configuration variables:

```
configutil -o gen.newuserforms -v 'Subject: Welcome!! $$ Sesta.com welcomes you
to the premier internet experience in Dafandzadgad!
```

Depending on the shell that you are using, it might be necessary to append a special character before $ to escape the special meaning of $. ($ is often the escape character for the shell.)

To Set a Per-Domain Greeting Message

Whenever you create a new hosted domain, it is a good idea to create per-domain greeting messages for your supported languages. If this is not done, the generic greeting message set by `gen.newuserform` will be sent.

You can set a greeting message for new users in each domain. The message can vary depending on the user's, the domain's, or the site's preferred language. This is done by setting the `mailDomainWelcomeMessage` attribute in the desired LDAP domain entry. The syntax is as follows:

`mailDomainWelcomeMessage;lang-`*user_prefLang*
`mailDomainWelcomeMessage;lang-`*domain_prefLang*
`mailDomainWelcomeMessage;lang-`*gen.sitelanguage*

The following example sets the domain welcome message for English:

```
mailDomainWelcomeMessage;lang-en: Subject: Welcome!! $$Welcome to the mail
system.
```

The following example sets the domain welcome message for French:

```
mailDomainWelcomeMessage;lang-fr: Subject: Bienvenue!! $$Bienvenue a siroe.com!
```

Using the above examples, we assume the following: 1) the domain is siroe.com, 2) a new user belongs to this domain, 3) the user's preferred language is French as specified by the LDAP attribute `preferredlanguage`, 4) siroe.com has the above English and French welcome messages available, 5) the site language is en as specified by `gen.sitelanguage`. See the *Directory Server Reference Manual* for a list of supported locales and their language value tag (http://docs.sun.com/source/816-6699-10/ax_inter.html#18744).

When the user logs in for the first time, they will receive the French greeting. If the French welcome message isn't available, they will get the English greeting.

Greeting Message Theory of Operations

Greeting messages can be set by both the LDAP attribute `mailDomainWelcomeMessage` and the `configutil` parameter `gen.newuserforms`. The order in which a message will get chosen, with the top one having the highest preference, is shown below:

`mailDomainWelcomeMessage;lang-`*user_prefLang*
`mailDomainWelcomeMessage;lang-`*domain_prefLang*
`mailDomainWelcomeMessage;lang-`*gen.sitelanguage*
`mailDomainWelcomeMessage`
`gen.newuserforms;lang-"$user-`*prefLang*`"`
`gen.newuserforms;lang-"$domain-`*prefLang*`"`
`gen.newuserforms;lang-"$gen.`*sitelanguage*`"`
`gen.newuserforms`

The algorithm works as follows: if there are no domains (or there are, but there is no per domain welcome message provisioned for them), a welcome message is configured with the `gen.newuserforms` parameter, if specified. If a user has a preferred language (set with the `preferredlanguage` LDAP attribute) and `gen.newuserforms;lang-`*user_prefLang* is set, the user will receive that welcome message at the time of their first log in to the server. If `gen.newuserforms;lang-`*gen.sitelanguage* is set, and `preferredlanguage` is not set, but the site language is set (using `gen.sitelanguage` parameter), user will receive that message. If no language tag parameter is set and a plain `gen.newuserforms` is set, then that message will be sent to the user. If none of the values are set, user will not receive any welcome message.

If the user is in a domain, then similar to the discussion above, the user might receive one of `mailDomainWelcomeMessage;lang-xx`, depending on which one is available in the list and in the order given.

Example: Domain is fantasia.com. The domain preferred language is German (de). But the new user in this domain has preferred language of Turkish (tr). Site language is English. The following values are available (`mailDomainWelcomeMessage` are attributes of the domain fantasia.com):

```
mailDomainWelcomeMessage;lang-fr
mailDomainWelcomeMessage;lang-ja
gen.newuserforms;lang-de
gen.newuserforms;lang-en
gen.newuserforms
```

According to the algorithm, the message sent to user will be `gen.newuserforms;lang-de`.

To Set a User-Preferred Language

Administrators can set a preferred language by setting the attribute `preferredLanguage` in the user's LDAP entry.

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 set a domain preferred language by setting the attribute `preferredLanguage` in 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. See the *Directory Server Reference Manual* for a list of supported locales and the language value tag (http://docs.sun.com/source/816-6699-10/ax_inter.html#18744).

To Customize Directory Lookups

Messaging Server cannot function without an LDAP-based directory system such as the Sun Java System 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 configuration directory that defines your *administrative topology*—the set of Sun Java System 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 *Sun ONE Server Console 5.2 Server Management Guide*.)
- 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 *Sun ONE Server Console 5.2 Server Management Guide*.)
- 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.

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 *Sun ONE Server Console 5.2 Server Management Guide* 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 Administration 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*`[:`*port_number*`]`

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

Setting a Failover LDAP Server

It is possible to specify more than one LDAP server for the user/group directory so that if one fails another takes over:

1. Set `local.ugldaphost` to the multiple LDAP machines. Example:

   ```
   configutil -o local.ugldaphost -v "server1 server2 ..."
   ```

2. Set `local.ugldapuselocal` to yes. This specifies that the user/group LDAP configuration data will be stored in the local configuration file. Otherwise, it is stored in LDAP. Example:

   ```
   configutil -o local.ugldapuselocal -v yes
   ```

If the first server on the list fails, the existing LDAP connections will get recognized as down and new connections will be made. When a new ldap connection is needed, the ldap library will try all the ldap servers in the order they're listed.

Just as there is failover for the user/group directory one can similarly set failover servers for configuration directory. The configuration attribute is `local.ldaphost`.

Chapter 5

Configuring POP, IMAP, and HTTP Services

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 Sun ONE Console or by using command-line utilities.

For information on configuring Simple Mail Transfer Protocol (SMTP) services, see Chapter 10, “About MTA Services and Configuration”.

This chapter contains the following sections:

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 121, “To Configure IMAP Services” on page 123, and “To Configure HTTP Services” on page 125.

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 97); 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. “To Specify What Services are Started” on page 100 also describes this process. You can also enable or disable services at the user level by setting specified LDAP attribute `mailAllowedServiceAccess`.

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 7, “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 599.

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 601), 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 Sun Java System 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 Sun ONE Console or the `configutil` utility (`service.imap.banner`, `service.pop.banner`) to set service banners. For detailed syntax information about `configutil`, see the *Sun Java System Messaging Server Administration Reference* (`http://docs.sun.com/doc/817-6267`).

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 121, "To Configure IMAP Services" on page 123, or "To Configure HTTP Services" on page 125. 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 606.

Certificate-Based Login

In addition to password-based authentication, Sun Java System 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 608.

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 121, "To Configure IMAP Services" on page 123, or "To Configure HTTP Services" on page 125.

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. 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 593.

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 593.

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 593.

Client Access Controls

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 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 612 and "Configuring Client Access to SMTP Services" on page 627.

To Configure POP Services

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

For more information, see also:

- "Enabling and Disabling Services" on page 114
- "To Set the Login Separator for POP Clients" on page 116
- "Specifying Port Numbers" on page 114
- "Number of Connections per Process" on page 119
- "Dropping Idle Connections" on page 120
- "Number of Threads per Process" on page 120

- “Number of Processes” on page 118

Console To configure the POP service using Console:

1. From Sun ONE 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 119.
   - Set the maximum idle time for connections. For more information, see “Dropping Idle Connections” on page 120.
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 120.
   - Set the maximum number of processes. For more information, see “Number of Processes” on page 118.
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 enable POP over SSL:

```
configutil -o service.pop.enablesslport -v 1
configutil -o service.pop.sslport -v 995
```

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 Sun ONE Console. Some of the more common IMAP services options are given in this section. A complete listing can be found in the *Sun Java System Messaging Server Administration Reference*.For more information, see also:

- "Enabling and Disabling Services" on page 114
- "Specifying Port Numbers" on page 114
- "Password-Based Login" on page 117
- "Number of Connections per Process" on page 119
- "Dropping Idle Connections" on page 120
- "Number of Threads per Process" on page 120
- "Number of Processes" on page 118

Console To configure the IMAP service from the Console:

1. From Sun ONE 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 119.
   - Set the maximum idle time for connections. For more information, see “Dropping Idle Connections” on page 120.
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 120.
   - Set the maximum number of processes. For more information, see “Number of Processes” on page 118.
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.imap.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
```

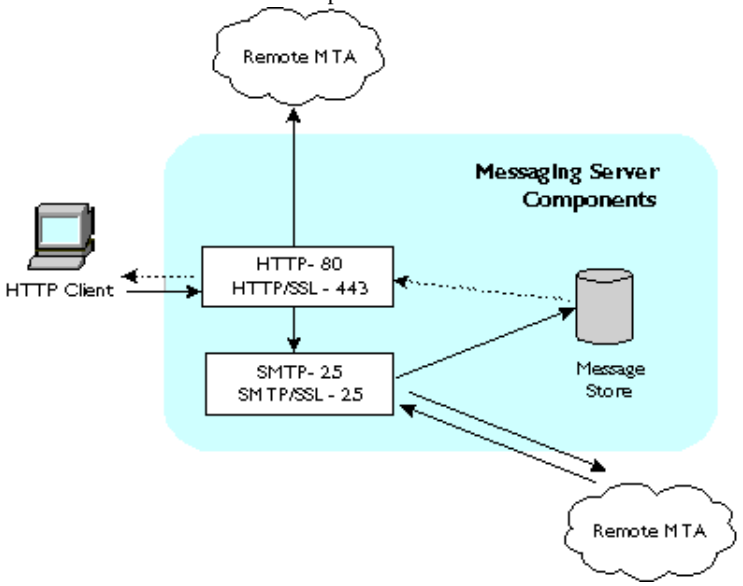
To specify a protocol welcome banner:

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

To Configure HTTP Services

POP and IMAP clients send mail directly to Messaging Server MTA for routing or delivery. In contrast, HTTP clients send mail to a specialized web server that is part of 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 5-1. If Messaging Server is used only for web-based email, disable both POP and IMAP.

Figure 5-1 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 *Sun Java System Messaging Server Administration Reference*. For more information, see also:

- “Enabling and Disabling Services” on page 114
- “Specifying Port Numbers” on page 114
- “Password-Based Login” on page 117
- “Number of Connections per Process” on page 119
- “Dropping Idle Connections” on page 120
- “Logging Out HTTP Clients” on page 121
- “Number of Threads per Process” on page 120
- “Number of Processes” on page 118

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 Sun ONE Console:

1. From Sun ONE 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 119.
   - Set the maximum idle time for connections. For more information, see “Dropping Idle Connections” on page 120.
   - Set the maximum idle time for client sessions. For more information, see “Logging Out HTTP Clients” on page 121.
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 120.
   - Set the maximum number of processes. For more information, see “Number of Processes” on page 118.
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 127.

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 127.
11. Click Save.

Command Line You can set values for the HTTP attributes at the command line as follows (see *Sun Java System Messaging Server Administration Reference* at http://docs.sun.com/doc/817-6267 for more info):

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
```

Enabling Single Sign-On (SSO)

Single sign-on is the ability for an end user to authenticate once (that is, log on with user ID and password) and have access to multiple applications. The Sun Java System Identity Server is the official gateway used for SSO for Sun Java System servers. That is, users must log into Identity Server to get access to other SSO configured servers.

For example, when properly configured, a user can sign in at the Sun Java System Identity Server login screen and have access to Messenger Express, in another window without having to sign in again. Similarly, if the Sun Java System Calendar Server is properly configured, a user can sign in at the Sun Java System Identity Server login screen, then have access to their Calendar in another window without having to sign in again.

It should be noted that Messaging Server provides two methods of deploying SSO. The first way is through the Sun Java System Identity Server, the second way is through communications servers trusted circle technology. Using a trusted circle is the legacy method of implementing SSO. Though this method provides some features not available with Identity Server SSO, we do not recommend using it since all future development will be with the Identity Server. Both methods, however, are described in the following sections:

- "Identity Server SSO for Sun Java System Servers" on page 131
- "Trusted Circle SSO (Legacy)" on page 134

Identity Server SSO for Sun Java System Servers

This section describes SSO using Identity Server. It consists of the following sections:

- "SSO Limitations and Notices" on page 132

SSO Limitations and Notices

- The Messenger Express session is only valid for as long as the Identity Server session is valid. If the user logs out of Identity Server the webmail session is automatically closed (single sign-off).
- SSO Applications working together must be in the same DNS domain. (Also known as cookie domain).
- SSO Applications must have access to Identity Server verification URL (naming service).
- Browsers must have cookies.

Configuring Messaging Server to Support SSO

Four `configutil` parameters support Messaging Server SSO. Of these four, only one, `local.webmail.sso.amnamingurl` is required to enable SSO with Messaging Server. To enable SSO, set this parameter to the URL where Identity Server runs the naming service. Typically this URL is http:/ /*server* / amserver / namingservice. Example:

```
configutil -o local.webmail.sso.amnamingurl -v
http://sca-walnut:88/amserver/namingservice
```

NOTE Identity Server SSO does not look at `local.webmail.sso.enable` which enables the older SSO mechanism. `local.webmail.sso.enable` should be left to `off` or unset otherwise warning messages will be logged about missing configuration parameters which are needed for the old SSO mechanism.

You can modify the SSO configuration parameters shown in Table 6-3, by using the `configutil` command.

Table 6-1 Identity Server Single Sign-On Parameters

Parameter	Description
`local.webmail.sso.amnamingurl`	The URL where Identity Server runs the naming service. Mandatory variable for single sign-on through Identity Server. Typically this URL is `http://<server>/amserver/namingservice.` Default: Not set.
`local.webmail.sso.amcookiename`	Identity Server cookie name. If Identity Server is configured to use another cookie name, then that name needs to be configured in Messaging Server as `local.webmail.sso.amcookiename` so that Messaging Server knows what to look for when doing single-sign on. Default value is `iPlanetDirectoryPro` and must not be changed if Identity Server has the default configuration. Default: `iPlanetDirectoryPro`
`local.webmail.sso.amloglevel`	AMSDK logging level. The SSO library used by Messaging Server has its own logging mechanism separate from Messaging Server. Its messages are logged in a file called `http_sso` under *msg_svr_basel*`log`. By default only messages with *info* or higher are logged, but it is possible to increase the logging level by setting the logging level to a value from 1 to 5 (1 = errors, 2 = warnings, 3 = info, 4 = debug, 5 = maxdebug). Be aware that the library doesn't have the same notion of message importances as Messaging Server and that setting the level to *debug* can result in a lot of meaningless data. Also the `http_sso` log file is not managed by common Messaging Server logging code and is never cleaned up or rolled over. It is the responsibility of the system administrator to clean it up when setting the log level higher than the default. Default: 3
`local.webmail.sso.singlesignoff`	Single sign-off from Messaging Server to Identity Server. Identity Server is the central authentication authority, and single sign-off is always enabled from Identity Server to Messaging Server. This option allows a site to configure whether the *logout* button in webmail should also log the user out of Identity Server (saving some customization work). By default this is enabled. If this is disabled, a user logging out of the default webmail client is automatically logged back in since logout refers to the root document and the root document refers to the inbox display as long as the Identity Server cookie exists and is valid. Therefore, a site choosing to disable this option needs to customize what happens at webmail logout. Default: yes

Troubleshooting SSO

If there is a problem with SSO, the first thing to do is check the webmail log file *msg_svr_base*/log/http for errors. Increasing the logging level may be helpful (`configutil -o logfile.http.loglevel -v debug`). If this does not help, then check the amsdk messages in *msg_svr_base*/log/http_sso, then increase the amsdk logging level (`configutil -o local.webmail.sso.amloglevel -v 5`). Note that new logging levels only take effect after server restart.

If SSO is still having problems, make sure you are using fully qualified host names of both Identity Server and Messaging Server during log in. Cookies are only shared between servers of the same domain, and browsers do not know what the domain is for local server names, so one must use the fully qualified names in the browser for SSO to work.

Trusted Circle SSO (Legacy)

This section describes trusted circle SSO. We do not recommend using this method of SSO since all future development will be with the Identity Server. However, there is some functionality available with trusted circle SSO that is not available with Identity Server SSO at this time.This section consists of the following sections:

Trusted Circle SSO Overview and Definitions

Before deploying SSO it is important to understand the following terminology.

- **SSO**: Single Sign-On. The ability to sign on to one application and be able access the other applications. The user identification is the same throughout all applications.

- **Trusted applications**. Applications sharing the SSO scheme (*SSO Prefix*) and trusting each other's cookies and verifications. Also known as *Peer SSO applications*.
- **Trusted circle**. The circle of trusted applications. They share the same SSO Prefix.
- **SSO Prefix**. A string defined by the person deploying SSO and made known to applications so they can use it to find cookies generated by other applications in the same trusted circle. Applications with different prefixes are not in the same circle and the user needs to re-authenticate when moving between these applications. The prefix sometimes, but not always, explicitly contains the trailing - ("-") in the configuration setting.
- **Application ID**. (appid). A unique string defined by the person deploying SSO for each application in the SSO circle.
- **SSO Cookie**. A token that the browser uses to remember that the user has authenticated to some application. The name of the cookie is of the form *SSO_prefix-application ID*. The value of the cookie is the SSO key, usually a session ID generated by the application.
- **Cookie Domain**. A domain within which the application is restricted to send cookies. This is a domain in the DNS sense.
- **Verification URL**. A URL used by one application to verify the cookie it found to another application.

Trusted Circle SSO Applications

Before implementing SSO, you must first consider which applications will be in this trusted circle. The applications which can be in this trusted circle are Messenger Express (with or without Messenger Express Multiplexor), Calendar Express, and the old iPlanet Delegated Administrator for Messaging (not recommended because it only supports Sun LDAP Schema 1).

Table 6-2 shows which applications can be accessed from each other via SSO. From a user's point of view, logging into one of the applications on the first column, SSO works if the user is able to access application across the top row without having to re-enter user id and passwords.

Table 6-2 SSO Interoperability

To: / From:	Calendar Express	Messenger Express	Messenger Express Multiplexor	Delegated Administrator
Calendar Express	SSO	SSO	SSO	SSO
Messenger Express	SSO	N/A	N/A	SSO
Messenger Express Multiplexor	SSO	N/A	N/A	SSO
Delegated Administrator	SSO	SSO	SSO	N/A

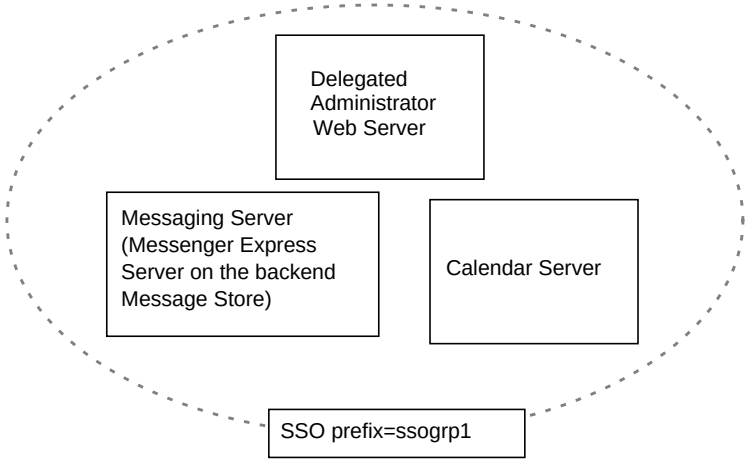
Trusted Circle SSO Limitations

- SSO Applications working together must be in the same domain.
- SSO Applications must have access to each other's SSO verification URL.
- Browsers must support cookies.
- For security purposes, SSO should not be used on machines where the browser is run.
- To switch to a different identity, the browser needs to be restarted.
- Assuming single sign-off is enabled in both Messenger Express and for Sun Java System Calendar Server, if you log out of Sun Java System Calendar Server, then you are supposed to have to login again to Messenger Express. If you log out of Messenger Express, then you would have to login again into Sun Java System Calendar Server. However, it currently does not work this way.You may stay logged into one after logging out of another.

Example Trusted Circle SSO Deployment Scenarios

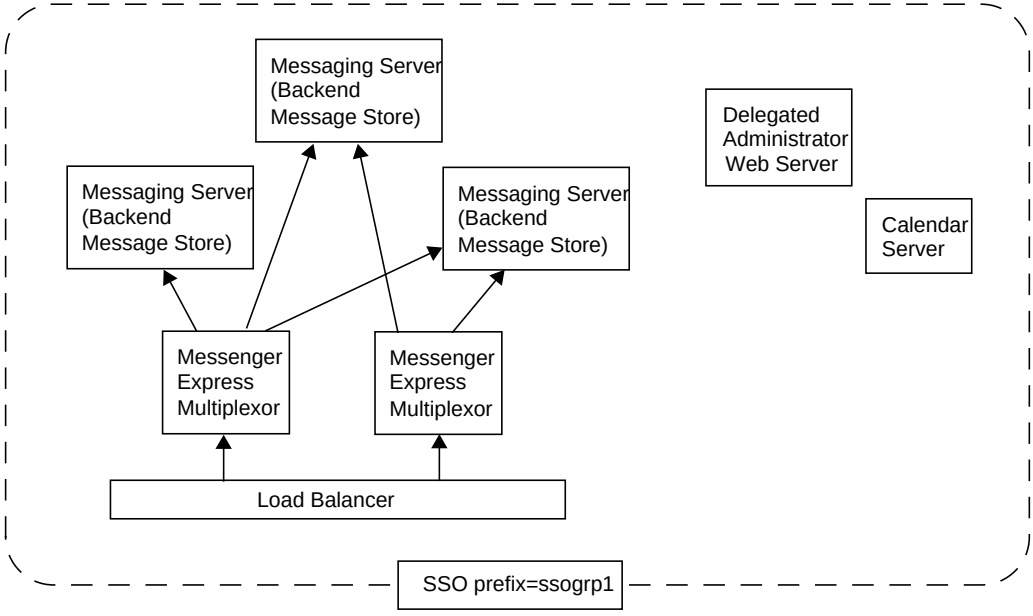
The simplest SSO deployment scenario consists of only Messenger Express and iPlanet Delegated Administrator for Messaging. A more complicated scenario can be created by adding Calendar Express—on the same machine or a different machine—using the same SSO prefix so they are in the same trusted circle. This is shown in Figure 6-1.

Figure 6-1 Simple SSO Deployment



An even more complex deployment would include Messenger Express Multiplexors and load balancers.

Figure 6-2 Complex SSO Deployment



Setting Up Trusted Circle SSO

This section describes setting up SSO for Messenger Express, iPlanet Delegated Administrator for Messaging, and Calendar Manager.

1. Configure Messenger Express for SSO.

   a. Set the appropriate SSO `configutil` parameters.

      To enable single sign-on for Messenger Express with Delegated Administrator, set the configuration parameters as follows (assumes your default domain is siroe.com). These parameters are described in Table 6-3. You must be root user. `cd` to *instance_root*

`configutil -o local.webmail.sso.enable -v 1`
`configutil -o local.webmail.sso.prefix -v ssogrp1`
`ssogrp1` is the default SSO Prefix used by iDA, although you can choose a different prefix, using the default would save a little typing when configuring iDA and iCS.
`configutil -o local.webmail.sso.id -v ims5`
ims5 is a name you pick to identify Messenger Express (ME) to other applications.
`configutil -o local.webmail.sso.cookiedomain -v ".siroe.com"`
The above domain must match the domain used by the ME/browser client to connect to the servers. Thus, although the hosted domain on this server may be called xyz.com, we must use a real domain in the DNS. This value must start with a period.
`configutil -o local.webmail.sso.singlesignoff -v 1`
`configutil -o local.sso.`*`ApplicationID`*`.verifyurl -v \`
`"http://`*`ApplicationHost:port`*`/verifySSO?"`
`ApplicationID` is a name we give to the SSO application (example: ida for Delegated Administrator, ics50 for Calendar Server). ApplicationHost:port is the host and port number of the application. You will have one of these lines for each non-Messaging Server applcation. Example:

```
configutil -o local.sso.ida.verifyurl -v \
   "http://siroe.com:8080/verifySSO?"
```

   b. Restart Messenger Express http server after changing the configuration.

```
cd instance_root
./stop-msg http
./start-msg http
```

2. Configure Directory Server for SSO.

a. Create a proxy user account in the directory.

The proxy user account allows the Delegated Administrator to bind to the Directory Server for proxy authentication. Using the following LDIF code (`proxy.ldif`) you could create a proxy user account entry using `ldapadd`.

```
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
```

`ldapadd -h mysystem.siroe.com -D "cn=Directory Manager" -w` *password* **`-v -f proxy.ldif`**

b. Create the appropriate ACIs for proxy user account authentication.

Using the `ldapmodify` utility, create an ACI for each of the suffixes you created at the time you installed the Delegated Administrator.

`osiroot` - The suffix you entered to store the user data (the default is `o=isp`). `osiroot` is the root of the Organization Tree.

`dcroot` - The suffix you entered to store the domain information. (The default is `o=internet`.)

`osiroot` - The suffix you entered to store the configuration information, it should have been the same value you entered to store the user data.

The following is an example of an ACI entry (`aci1.ldif`) for the `osiroot` for the proxy user created earlier:

```
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";)
```

`ldapmodify -h siroe.com -D "cn=Directory Manager" -w` *password* **`-v -f aci1.ldif`**

Create a similar ACI entry (`aci2.ldif`)for the `dcroot`:

```
dn: o=internet
changetype: modify
add: aci
aci: (target="ldap:///o=internet")(targetattr="*")(version 3.0; acl
"proxy";allow (proxy) userdn="ldap:///uid=proxy, ou=people, o=siroe.com,
o=isp";)
```

`ldapmodify -h siroe.com -D "cn=Directory Manager" -w` *password* **`-v -f aci2.ldif`**

3. Configure the Delegated Administrator

a. Add the proxy user credentials and cookie name for context to the Delegated Administrator `resource.properties` file.

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
NDAAuth-singleSignOnId=SSO_Prefix-
NDAAuth-applicationId=DelAdminID
```

For example:

```
LDAPDatabaseInterface-ldapauthdn=
   uid=proxy, ou=people, o=siroe.com, o=isp
LDAPDatabaseInterface-ldapauthpw=proxypassword
NDAAuth-singleSignOnId=ssogrp1-
NDAAuth-applicationId=ida
```

b. Add the participating server's verification URL.

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

Following the example, assume Messenger Express is installed and its application ID is `msg5`. Edit the Delegated Administrator *iDA_server_root*`/nda/classes/netscape/nda/servlet/resource.properties` file and add an entry such as:

```
verificationurl-ssogrp1-msg5=http://webmail_hostname:port/VerifySSO?
verificationurl-ssogrp1-ida=http://iDA_hostname:port/VerifySSO?
verificationurl-ssogrp1-ics50=http://iCS_hostname:port/VerifySSO?
```

4. Add Delegated Administrator single sign-on cookie information and enable UTF8 Parameter Encoding.

a. Define a context identifier for Delegated Administrator.

Edit *Web_Server_Root*`/https-`*instancename*`/config/servlets.properties` and uncomment all lines containing the text `servlet.*.context=ims50`. Where `*` is any string.

b. Specify a cookie name for the context in the Enterprise Server configuration.

Edit the Enterprise Server file *Web_Server_Root*/https-*instancename*/config/contexts.properties and add the following line to the bottom of the file before the #IDACONF-Start line:

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

c. Enable UTF8 parameter encoding for ims5 contexts.

To Enable UTF8 Parameter Encoding for ims5 Contexts in the Enterprise Server configuration add the following entry to the Enterprise Server *WebServer_Root*/https-*instancename*/config/contexts.properties file:

```
context.ims50.parameterEncoding=utf8
```

5. Restart Messenger Express.

After you've made the configuration changes described in steps 1a through 2c, you must restart Messenger Express for the changes to take effect:

WebServer_Root/**https-i***instance_name*/**stop**
WebServer_Root/**https-***instancename*/**start**

6. If you are deploying Calendar in this SSO group, configure Calendar Server.

Edit ics.conf and add the following:

```
sso.appid = "ics50"
sso.appprefix = "ssogrp1"
sso.cookiedomain = ".red.iplanet.com"
sso.enable = "1"
sso.singlesignoff = "true"
sso.userdomain = "mysystem.red.iplanet.com"
sso.ims5.url="http://mysystem.red.iplanet.com:80/VerifySSO?"
sso.ida.url=http://mysystem.red.iplanet.com:8080/VerifySSO?
```

7. Restart Calendar Server

start-cal

8. Restart the Messenger Express http server:

msg_svr_base/sbin/**stop-msg http**
msg_svr_base/sbin/**start-msg http**

Messenger Express Trusted SSO Configuration Parameters

You can modify the single sign-on configuration parameters for Messenger Express, shown in Table 6-3, by using the `configutil` command. For more information about `configutil`, see the *Messaging Server Reference Manual*.

Table 6-3 Trusted Circle 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 Messenger Express 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. This string must match what's used by the iPlanet Delegated Administrator for Messaging in its `resource.properties` file without the trailing -. For example, if: `NDAAuth-singleSignOnID=ssogrp1-` Then this value should be set here to `ssogrp1.`
`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. This is an arbitrary string. Its value must match what you specify for the Delegated Administrator in its `resource.properties` file. The corresponding entry in `resource.properties` would be: `Verifycationurl-XXX-YYY=http://webmailhost:webmailport/VerifySSO?` Where XXX is the `local.webmail.sso.prefix` value set above, and YYY is the value of `local.webmail.sso.id` set here.

Table 6-3 Trusted Circle Single Sign-On Parameters

Parameter	Description
`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. This domain must match the DNS domain used by the Messenger Express browser to access the server. It is not the hosted domain name.
`local.webmail.sso.singlesignoff`	The integer value of this parameter, if set to any non-zero value, clears all SSO cookies on the client with prefix values matching the value configured in local.webmail.sso.prefix when the client logs out. If set to zero, Messenger Express will clear its own SSO cookie when the client logs out. The default value is zero.
`local.sso.`*`appid`*`.verifyurl`	Sets the verify URL values for peer SSO applications. *appid* is the application ID of a peer SSO application whose SSO cookies are to be honored. For example, the default *appid* for Delegated Administrator is `nda45`. Its actual value is specified by the Delegated Administrator `resource.properties` file entry `NDAAuth-applicationID`. 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?` If you are using a load balancer in front of multiple Messenger Express Multiplexors and Message Store servers (running Messenger Express) or Calendar front ends, be sure to assign a *different* `appid` for each physical system with the real host names in the `verifyurl`. This will ensure that the correct system will be used to verify the cookie

Chapter 7

Configuring and Administering Multiplexor Services

This chapter describes two multiplexors that are included with 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:

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 Messaging Server, the Messenger Express Multiplexor is built-in to the HTTP service (`mshttpd`) that is included with the 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 Messaging Multiplexor

The Sun Java System 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 Messaging Server. You can install the MMP at the same time you install the Messaging Server or other Sun Java System servers, or you can install the MMP separately at a later time.

The MMP supports:

- Both unencrypted and encrypted (SSL) communications with mail clients.

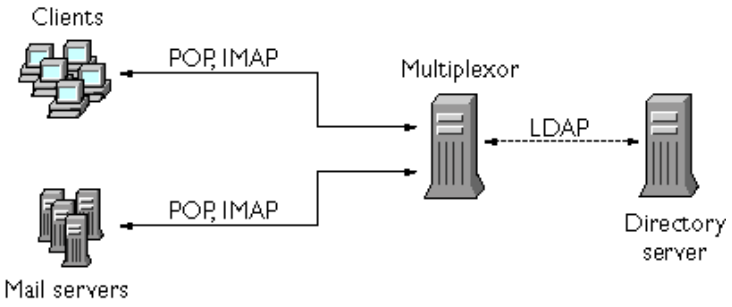
- Client certificate-based authentication, described in “Certificate-Based Client Authentication” on page 150.
- User pre-authentication, described in “User Pre-Authentication” on page 151.
- Virtual domains that listen on different IP addresses and automatically append domain names to user IDs, described in “MMP Virtual Domains” on page 151.
- Multiple installations of the MMP on either the same server (see “Multiple Messaging Multiplexor Installs” on page 153) or different servers (see the *Messaging Server Installation Guide*). Separate installations on the same server can have separate configurations for 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 623.

How the Messaging Multiplexor Works

The 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 7-1 shows how servers and clients relate to each other in an MMP installation.

Figure 7-1 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

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

When SSL is enabled, the MMP supports 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. See "Configuring MMP with SSL" on page 159 for details.

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* (`http://docs.sun.com/doc/817-6267`).

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 149.

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 or POP 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 149.
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 604.

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 the time configured on the server 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 user IDs 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 `inetDomainBaseDN` 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.

When Sun LDAP Schema 2 is enabled (see the *Sun ONE Messaging Server Installation Guide* and *Sun Java System Communcations Services Schema Reference Manual*), the user subtree for the specified domain will be all the users in the subtree below the organization node for that domain.

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`
`RestrictPlainPasswords`

StoreAdmin and StoreAdminPass
SearchFormat
TCPAccess
TCPAccessAttr

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

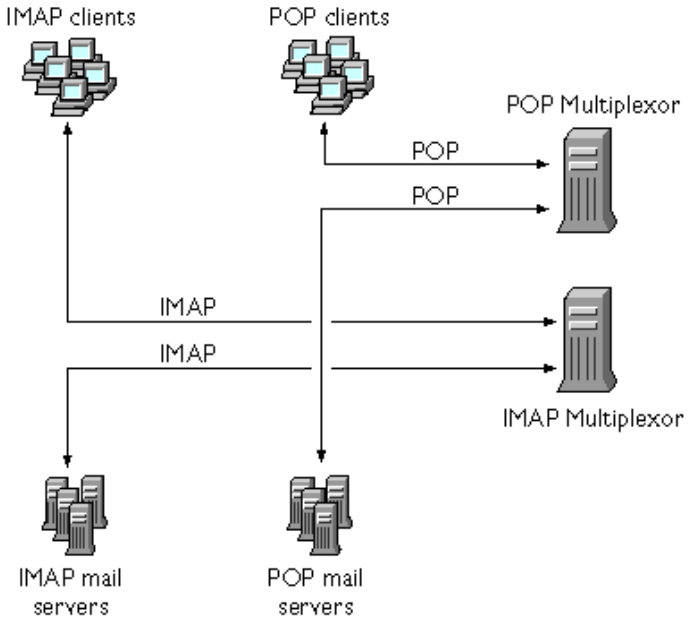
For detailed descriptions of these configuration parameters, see the *Messaging Server Reference Manual*.

Multiple Messaging Multiplexor Installs

You can do multiple installs of the MMP on a single server. Each of these installs will run as a separate process and can have different configuration files. Multiple installs 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 install of the MMP to support both POP, IMAP, and SMTP protocols (as shown in Figure 7-1), or you can create separate MMP installs for each protocol, as shown in Figure 7-2. By splitting messaging services across different machines, you can tune the resources on each computer for maximum performance.

Figure 7-2 Separate MMP installs for Each Protocol



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 623. 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 609.

Setting Up the Messaging Multiplexor

During the initial runtime configuration of Messaging Server, you determined if you wanted to configure the MMP on a machine. You could either set it up on the same machine as your Messaging Server or set it up on a separate machine.

NOTE MMP does not cache DNS results. A high quality caching DNS server on the local network is a requirement for a production deployment of Messaging Server.

The following sections describe how to set up the MMP:

- "Before You Configure MMP" on page 155
- "Multiplexor Configuration" on page 156
- "Multiplexor Files" on page 157
- "Starting the Multiplexor" on page 158
- "Modifying an Existing MMP" on page 158

More information about the MMP can be found in the following:

- *Sun Java System Messaging Server Administration Reference*: MMP Syntax and Structure (http://docs.sun.com/doc/817-6267)

Before You Configure MMP

Before configuring the MMP:

1. Choose the machine on which you will configure the MMP. It is best to use a dedicated machine set aside for the MMP.

NOTE It is recommended that the MMP not be enabled on a machine that is also running either the POP or IMAP servers.

If you install MMP on the same machine as Messaging Server, you must make sure that the POP and IMAP servers are set to non-standard ports. That way, the MMP and Messaging Server ports will not conflict with one another.

2. On the machine that the MMP is to be configured, create a UNIX system user to be used by the MMP. This new user must belong to a UNIX system group. See "To Create UNIX System Users and Groups" on page 32

3. Set up the Directory Server and its host machine for use with Messaging Server, if they are not already set up. See "To Prepare Directory Server for Messaging Server Configuration" on page 33

4. If the MMP is upgraded before the backend servers, customers should set the `Capability` option in `ImapProxyAService.cfg` to match the response to the `capability` command from the older backend. This would be:

   ```
   IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ NAMESPACE UIDPLUS CHILDREN LANGUAGE
   XSENDER X-NETSCAPE XSERVERINFO
   ```

   Note that line break is for editorial clarity, and that the configuration value must be on one line.

Multiplexor Configuration

To configure the MMP, you must use the Messaging Server configure program, which gives you the option of enabling the Messaging Multiplexor. For detailed information about the configure program see "To Create the Initial Messaging Server Runtime Configuration" on page 43.

To configure the MMP:

1. Put Sun Java System Messaging Server on the machine where you are installing and configuring the MMP.

   You must install the Administration Server, Java, and Messaging Server packages.

2. Configure the MMP by creating the Messaging Server Initial Runtime Configuration. See "To Create the Initial Messaging Server Runtime Configuration" on page 43.

   Note the following exception: when installing Messaging Server, check only the Messaging Multiplexor option.

Multiplexor Files

The Messaging Multiplexor files are stored in the *msg_svr_base*/config configuration file directory. You must manually edit the configuration parameters in the Messaging Multiplexor configuration files listed in Table 7-1.

Table 7-1 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.
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 623.
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 *msg_svr_base*/config directory, where *msg_svr_base* is the directory where you installed the Messaging Server.

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 configured or upgraded, the configuration template files will be overwritten.

For a complete description of all MMP configuration parameters, see the *Sun Java System Messaging Server Administration Reference*.

Starting the Multiplexor

To start, stop, or refresh an instance of the Messaging Multiplexor, use the one of the following commands in Table 7-2 located in the *msg_svr_base*/sbin directory:

Table 7-2 MMP Commands

Option	Description
start-msg mmp	Starts the MMP (even if one is already running).
stop-msg mmp	Stops the most recently started MMP.
refresh mmp	Causes an MMP that is already running to refresh its configuration without disrupting any active connections.

Modifying an Existing MMP

To modify an existing instance of the MMP, edit the ImapProxyAService.cfg and/or PopProxyAService.cfg configuration files as necessary. These configuration files are located in the *msg_svr_base*/config subdirectory.

Configuring MMP with SSL

To configure the MMP to use SSL, do the following:

NOTE It is assumed that the MMP is installed on a machine that does not have a Message Store or MTA.

1. Administration Server must be installed and configured.
2. Go to your Administration Server installation directory and run `mpsadmserver startconsole` to login to the Sun ONE Console:

   ```
   /usr/sbin/mpsadmserver startconsole
   ```

3. Use the Admin Console to install an SSL server certificate.

   See http://docs.sun.com/db/doc/816-5572-10

4. From the command line, make the following symbolic links to simplify things:

   ```
   cd msg_svr_base/config
   ln -s /var/mps/serverroot/alias/admin-serv-instance-cert7.db cert7.db
   ln -s /var/mps/serverroot/alias/admin-serv-instance-key3.db key3.db
   ```

   Also, make sure that those files are owned by the system ID under which the MMP will run.

5. Since the `sslpassword.conf` file is set during the initial Messaging Server runtime configuration, you do not need to set one up. See "To Create the Initial Messaging Server Runtime Configuration" on page 43.

NOTE An alternative approach to steps 1-8 is to copy the following files: `cert7.db`, `key3.db`, `secmod.db`, and `sslpassword.conf` from an existing Messaging or Directory Server. These servers must have a server certificate and a key appropriate for the same domain already installed.

6. Edit the `ImapProxyAService.cfg` file and uncomment the relevant SSL settings.
7. If you want SSL and POP, edit the `PopProxyAService.cfg` file and uncomment the relevant SSL settings.

   Additionally, you must edit the `AService.cfg` file and add `|995` after the `110` in the `ServiceList` setting.

8. Make sure that the `BindDN` and `BindPass` options are set in the `ImapProxyAService.cfg` and `PopProxyAService.cfg` files.

   You should also set the `DefaultDomain` option to your default domain (the domain to use for unqualified user names).

If you just want server-side SSL support, you are finished. Start the MMP with the following command in the *msg_svr_base*/`sbin` directory:

```
start-msg mmp
```

If you want client certificate based login, do the following:

1. Get a copy of a client certificate and the CA certificate which signed it.
2. Start the Sun ONE Console as before (on the same machine as the MMP), but this time import the CA certificate as a Trusted Certificate Authority.
3. Use the Store Administrator you created during your Messaging Server installation.

   For more information, see the "Specifying Administrator Access to the Store" on page 519.
4. Create a `certmap.conf` file for the MMP. For example:

   ```
   certmap default  default
   default:DNComps
   default:FilterComps  e=mail
   ```

   This means to search for a match with the e field in the certificate DN by looking at the mail attribute in the LDAP server.
5. Edit your `ImapProxyAService.cfg` file and:

   a. Set `CertMapFile` to `certmap.conf`

   b. Set `StoreAdmin` and `StorePass` to values from Step 3.

   c. Set `UserGroupDN` to the root of your Users and Groups tree.
6. If you want client certificates with POP3, repeat Step 5 for the `PopProxyAService.cfg` file.
7. If the MMP is not already running, start it with the following command in the *msg_svr_base*/`sbin` directory:

   ```
   start-msg mmp
   ```

8. Import the client certificate into your client. In Netscape™ Communicator, click on the padlock (Security) icon, then select Yours under Certificates, then select Import a Certificate... and follow the instructions.

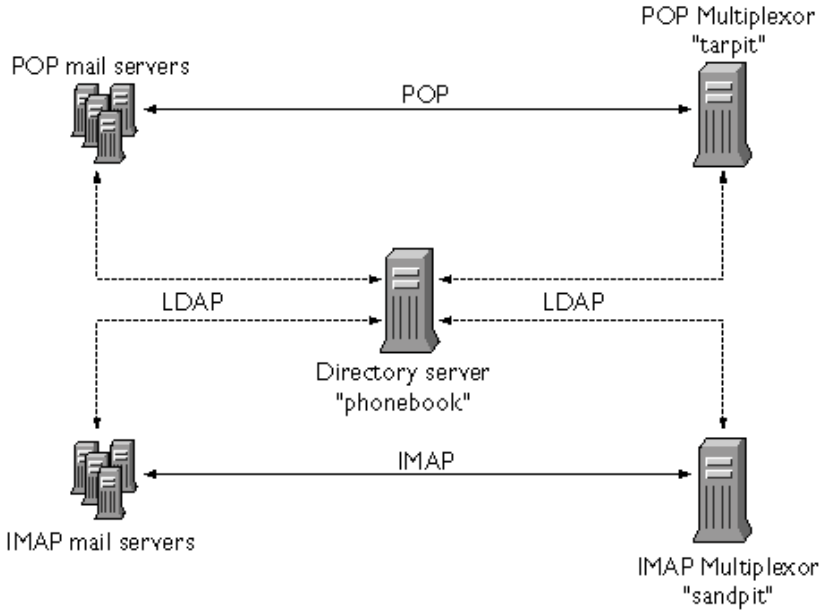
NOTE All your users will have to perform this step if you want to use client certificates everywhere.

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 7-3.

Figure 7-3 Multiple MMPs Supporting Multiple Messaging Servers



IMAP Configuration Example

The IMAP Messaging Multiplexor in Figure 7-3 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.siroe.com/o=internet
default:LogDir              /opt/SUNWmsgsr/config/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
BINARY LANGUAGE XSENDER X-NETSCAPE XSERVERINFO"
default:SearchFormat        (uid=%s)
default:SSLEnable           yes
default:SSLPorts            993
default:SSLSecmodFile       /opt/SUNWmsgsr/config/secmod.db
default:SSLCertFile         /opt/SUNWmsgsr/config/cert7.db
default:SSLKeyFile          /opt/SUNWmsgsr/config/key3.db
default:SSLKeyPasswdFile    ""
default:SSLCipherSpecs      all
default:SSLCertNicknames    Siroe.com Server-Cert
default:SSLCacheDir         /opt/SUNWmsgsr/config
default:SSLBacksidePort     993
default:VirtualDomainFile   /opt/SUNWmsgsr/config/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          /opt/SUNWmsgsr/config/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 7-3 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.siroe.com/o=internet
default:LogDir              /opt/SUNWmsgsr/config/log
default:LogLevel            5
default:BindDN              "cn=Directory Manager"
default:BindPass            password
default:BacksidePort        110
default:Timeout             1800
default:SearchFormat        (uid=%s)
default:SSLEnable           no
default:VirtualDomainFile   /opt/SUNWmsgsr/config/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          /opt/SUNWmsgsr/config/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
```

MMP Tasks

This section describes miscellaneous MMP configuration tasks. These include:

To Configure Mail Access with MMP

The MMP is not configured automatically, it has to be explicitly configured. In addition, the MMP does not make use of the `PORT_ACCESS` mapping table. If you wish to reject SMTP connections from certain IP addresses and you are using the MMP, you must use the `TCPAccess` option. The syntax of this option is the same as `mailDomainAllowedServiceAccess` (see *Sun Java System Communcations Services Schema Reference Manual* `http://docs.sun.com/doc/817-5702`). It is also described at "Filter Syntax" on page 614.

To Set a Failover MMP LDAP Server

It is possible to specify more than one LDAP server for the MMP so that if one fails another takes over. Modify your `PopProxyAservice.cfg` or `IMAPProxyAservice.cfg` to the following:

`default:LdapUrl "ldap://`*ldap01.yourdomain ldap02.yourdomain*`/o=INTERNET"`

About Messenger Express Multiplexor

The Sun Java System 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 Sun Java System 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 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 19, "Configuring Security and Access Control".

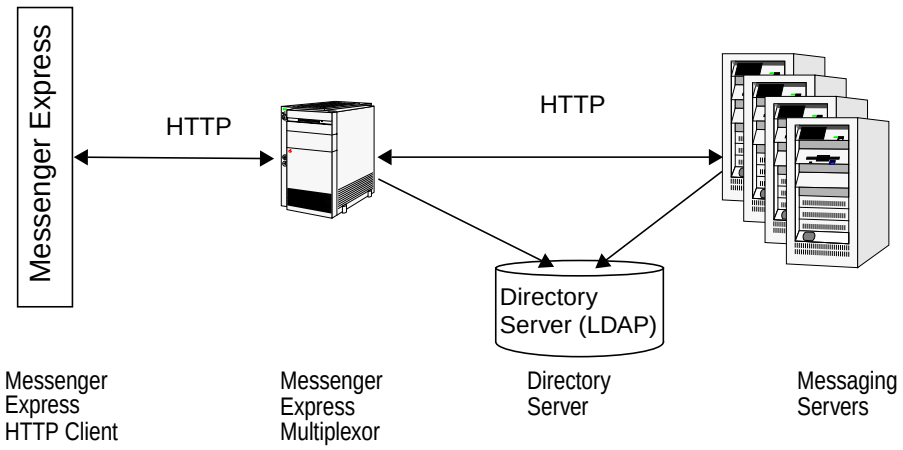
- Hosted Domains

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 Messaging Server (Messenger Express). The Messenger Express Multiplexor facilitates distributing mailboxes across multiple server machines. Clients connect to the Multiplexor when logging onto 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 7-4 on page 166 describes where the Messenger Express Multiplexor resides in an Messaging Server installation.

Figure 7-4 Overview of iPlanet Messenger Express Multiplexor



The Messenger Express Multiplexor interfaces between the Messenger Express client and 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.

Setting 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:

- "To Install Messaging Server on Proxy Machine" on page 167
- "To Configure Messenger Express Multiplexor Parameters" on page 167
- "To Enable Messenger Express Multiplexor" on page 169

To 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 *Sun ONE 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 Messaging Server through the Messenger Express Multiplexor.

To Configure Messenger Express 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 *msg_svr_base*`/sbin/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.

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

- "LDAP Parameters" on page 168
- "dcroot" on page 169
- "Default Domain" on page 169
- "Login Separator" on page 169

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
```

To 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 *msg_svr_base*`/sbin/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.

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.

To 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 browser 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 *msg_svr_base*/`log` 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 Customer Support.

Administering Your Messenger Express Multiplexor

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

To 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 606.

To Set Up Multiple Proxy Servers

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.

To Manage Different Versions of Messaging Server and Messenger Express Multiplexor

If you use different versions of 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 the user's mail host. The Multiplexor finds these files in the *msg_svr_base*/`config/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 *msg_svr_base*/`config/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 Messaging Server 6 2003Q4 and you have installed Messaging Server 6 2004Q2 as the Messenger Express Multiplexor, you need to replace the entire contents of the directory *msg_svr_base*/`config/html` of the Messenger Express Multiplexor with the contents of the same directory from the Messaging Server 6 2003Q4 back-end server. When you eventually upgrade Messaging Server 6 2003Q4 to Messaging Server 6 2004Q2, you can update these static files in directory *msg_svr_base*/`config/html` for the Messenger Express Multiplexor server as well.

To Configure the Port of the Back-end Messaging Server with the Messenger Express Multiplexor

If you want to configure the port of the back-end HTTP Messaging Server with the Messenger Express Multiplexor, use the following `configutil` command on your Multiplexor machine:

`local.service.http.proxy.port.`*`hostname`*

where *hostname* is the host of the back-end HTTP Messaging Server.

For example, if the host name of the back-end messaging server is `sesta.com` and the port number is `8888`, the command would be in the following format:

```
configutil -o local.service.http.proxy.port.store.sesta.com -v 8888
```

`local.service.proxy.port` applies to all back-end message stores except those which have their own port (same as `local.service.proxy.admin`).

To Configure Single Sign-on

Single sign-on must be configured on the Messenger Express Multiplexor machine in the same way as the Messaging (HTTP) server, with the following additional configurations:

`configutil -o local.service.http.proxy.admin -v` *`store_administrator`*

where *store_administrator* is the back-end store administrator specified during your back-end Messaging Server installation.

`configutil -o local.service.http.proxy.adminpass -v` *`store_admin_password`*

where *store_admin_password* is the back-end store administrator password specified during your back-end Messaging Server installation.

If you are using multiple back-end Messaging Servers that use different store administrators and passwords, you can configure them individually by appending the fully qualified host name to each configuration variable in Messenger Express Multiplexor:

`configutil -o local.service.http.proxy.admin.`*`hostname`* `-v` *`store_administrator`*

`configutil -o local.service.http.proxy.adminpass.`*`hostname`* `-v` *`store_admin_password`*

where *hostname* is the host of the back-end HTTP Messaging Server, *store_administrator* and *store_admin_password* are the back-end store administrator and password specified during your back-end Messaging Server installation.

To log the user into the back-end servers, Messenger Express Multiplexor uses the `proxyauth` login command. To enable `proxyauth`, use the following `configutil` parameter on the back-end message store:

```
configutil -o service.http.allowadminproxy -v 1
```

NOTE If Single sign-on is enabled through the Messenger Express Multiplexor, it does not need to be configured on the back-end HTTP Messaging Servers.

Chapter 8

MTA Concepts

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

The MTA Functionality

The Message Transfer Agent, or *MTA* (Figure 8-2 on page 177), is a component of the Messaging Server (Figure 8-1 on page 176)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 8-1 Messaging Server, Simplified Components View (Messenger Express not Shown)

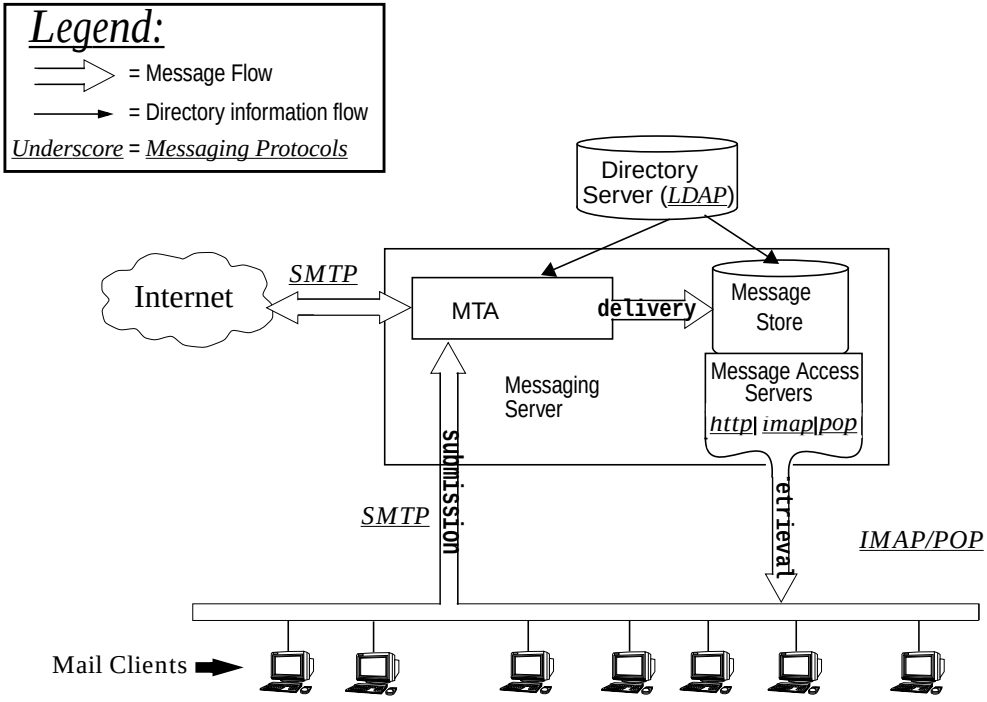
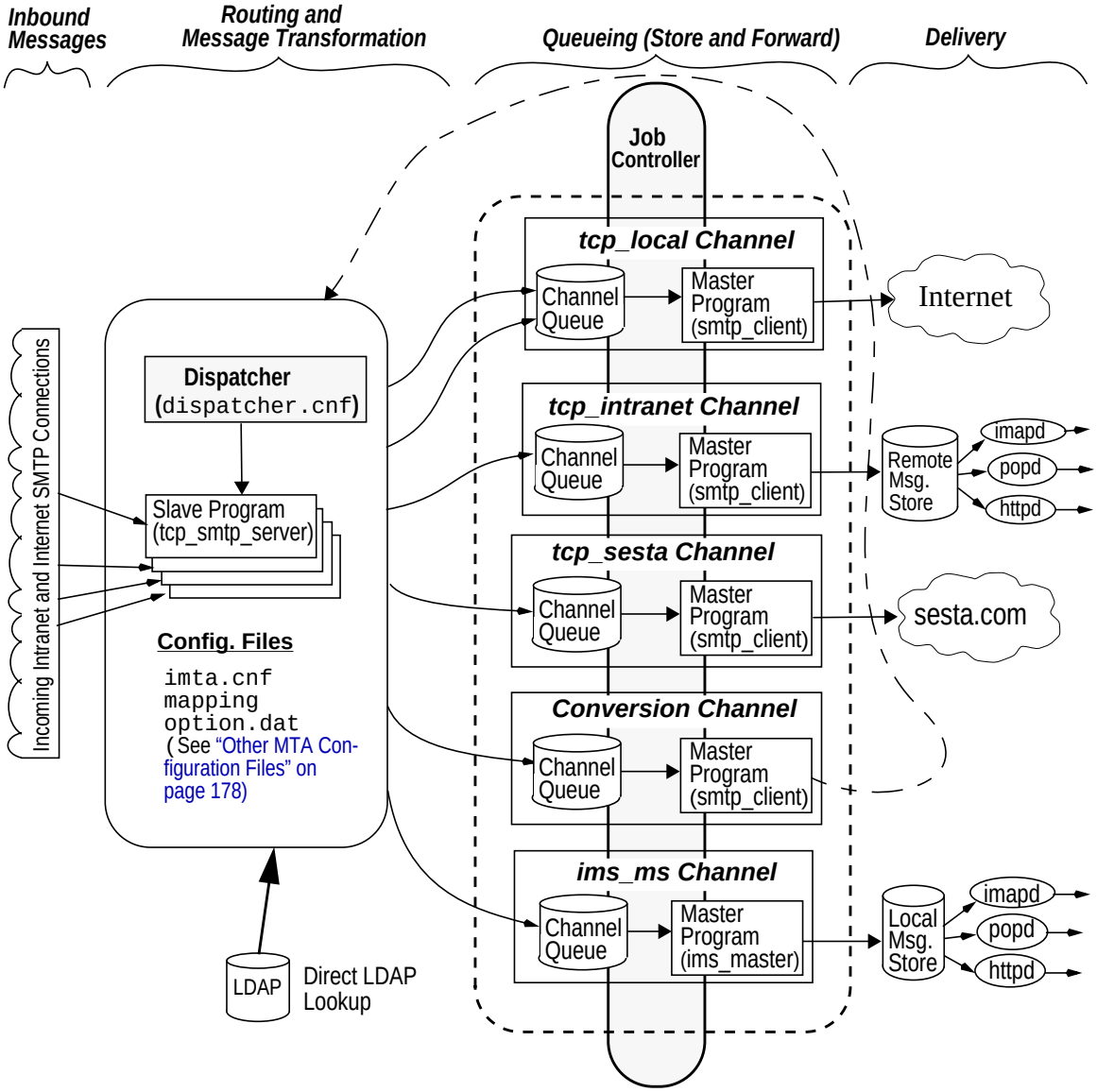


Figure 8-2 MTA Architecture



MTA Architecture and Message Flow Overview

This section provides a short overview of MTA architecture and message flow (Figure 8-2). Note that the MTA is a highly complex component and that Figure 8-2 is *simplified* depiction of messages flowing through the system. In fact, this picture is not a perfectly accurate depiction of all messages flowing through the system. For purposes of conceptual discussion, however, it must suffice.

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. The dispatcher maintains pools of multithreaded processes for each service. As additional sessions are requested, the dispatcher activates an SMTP server program to handle each session. A process in the Dispatcher's process pool may concurrently handle many connections. 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 17, "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 180

Routing

SMTP servers enqueue messages, but so can a number of other channels including, the conversion channel and reprocess channel. During this phase of delivery, however, 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 181).

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 *Sun Java System Messaging Server Administration Reference* and Chapter 12, "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 240 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 240.

To Start and Stop the Dispatcher

To start the Dispatcher, execute the command:

```
start-msg dispatcher
```

This command subsumes and makes obsolete any other `start-msg` 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:

```
stop-msg 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`). `$U` specifies that the rewritten address use the same user name. % specifies that the rewritten address use the same domain separator. `$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 11, “Configuring Rewrite Rules”” for more details.

For more information about configuring rewrite rules, see “The MTA Configuration File” on page 222 and Chapter 11, “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.

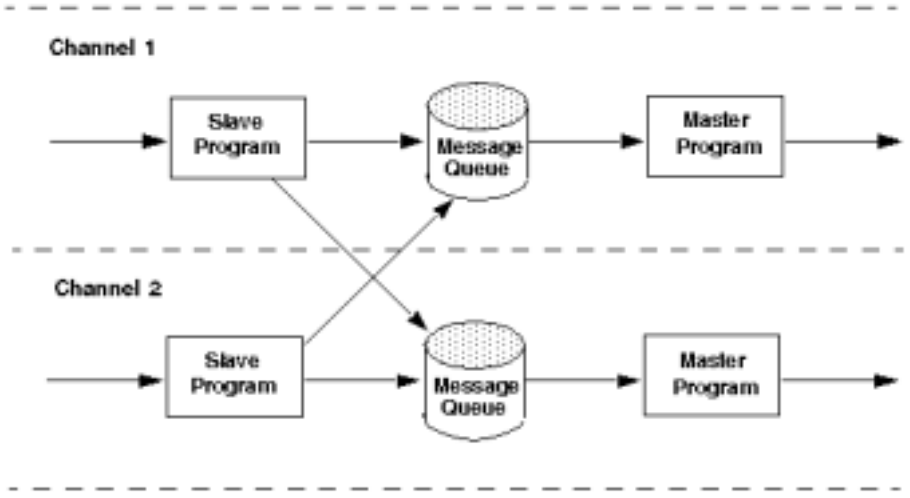
The slave channel program typically accepts incoming connections where the MTA is responding to an external request. The slave channel program:

- 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 8-3 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.

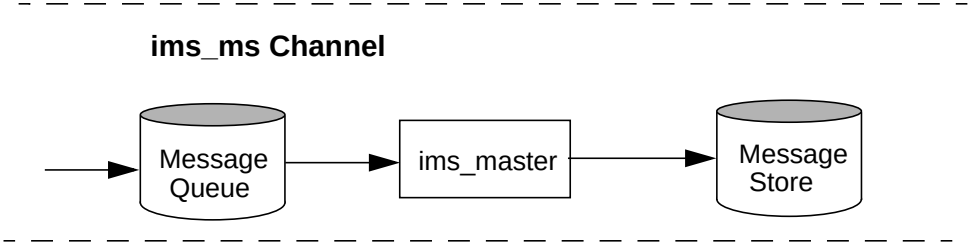
The master program dequeues the message from the queue and initiates network transport of the message. Note that the master program can only dequeue messages from its own channel queue.

Figure 8-3 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 8-4.

Figure 8-4 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: *msg_svr_base*/data/queue/*channel*/*.

CAUTION Do not add any files or directories in the MTA queue directory (that is, the value of IMTA_QUEUE in the imta_tailor file) as this will cause problems. When using a separate file system for the MTA queue directories, create a subdirectory under that mount point and specify that subdirectory as the value of IMTA_QUEUE.

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 222). 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 the example below, the channel host table contains three channel definitions or blocks.

```
! 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 *Sun Java System Messaging Server Administration Reference* and Chapter 12, “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 241, and "TCP/IP (SMTP) Channel Option Files" on page 239. For details on configuring channels, see Chapter 12, "Configuring Channel Definitions"." For more information about creating MTA channels, see "The MTA Configuration File" on page 222.

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 directly accesses the LDAP directory. This is fully described in Chapter 9, "MTA Address Translation and Routing."

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 242 and "Processing Pools for Channel Execution Jobs" on page 348.)

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.

For information about pools and configuring the Job Controller, see "Job Controller File" on page 242 and "Configuring Message Processing and Delivery" on page 343.

To Start and Stop the Job Controller

To start the Job Controller, execute the command:

```
start-msg job_controller
```

To shut down the Job Controller, execute the command:

```
stop-msg 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 9

MTA Address Translation and Routing

Prior to Messaging Server 6 2003Q4, Messaging Server used to access all user, domain, and group data from a database that was compiled from information stored in an LDAP server. When directory information was updated in the LDAP server, the database information was synchronized with a program called `dirsync`. The Messaging Server MTA now accesses the LDAP directory directly. This chapter describes the flow of data through the MTA using direct LDAP data access. It contains the following sections:

The Direct LDAP Algorithm and Implementation

The following sections describe direct LDAP processing.

Domain Locality Determination

Starting with an address of the form *user@domain*, the address translation and routing process first checks to see if *domain* is local.

Rewrite Rule Machinery

A facility has been added to the MTA rewrite rule machinery to check a given string to see if it is a domain we need to handle locally. This new facility is activated by a $V or $Z metacharacter. These new metacharacters are syntactically similar to the existing $N, $M, $Q, and $C metacharacters, that is, they are followed by a pattern string. In the case of $N, $M, $Q, and $C the pattern is matched against either the source or destination channel. In the case of $V and $Z the pattern is a domain and the check is to see whether or not it is local. $V causes a rule failure for a nonlocal domain and $Z causes a rule failure for a local domain.

The handling of these metacharacters is implemented as the following procedure:

1. Messaging Server verifies that the current domain is local.
2. If the base DN determination succeeds, the attribute specified by the LDAP_DOMAIN_ATTR_ROUTING_HOSTS MTA option (default mailRoutingHosts) is retrieved. If this attribute is present, it lists the set of hosts able to handle users in this domain. This list is compared against the host specified by the local.hostname configutil parameter and the list of hosts specified by the local.imta.hostnamealiases configutil parameter. These options can be overridden by the LDAP_LOCAL_HOST and LDAP_HOST_ALIAS_LIST MTA options, respectively. If there is a match or the attribute is not present on the domain, the domain is local. If no match occurs, the domain is nonlocal.

   The handling of domains considered to be nonlocal because of the mailRoutingHosts attribute depends on the setting of the ROUTE_TO_ROUTING_HOST MTA option. If the option is set to 0 (the default), the address is simply treated as nonlocal and MTA rewrite rules are used to determine routing. If the option is set to 1, a source route consisting of the first value listed in the LDAP_DOMAIN_ATTR_ROUTING_HOSTS MTA option is prepended to the address.
3. If the base DN determination fails, remove a component from the left hand side of the domain and go to Step 1. If no components remain continue to Step 4.

   A consequence of this backtracking up the domain tree is that if domain.com is recognized as local, any subdomain of domain.com will be recognized as local. Situations may arise where this is undesirable, so an MTA option, DOMAIN_UPLEVEL, is provided to control this behavior. Specifically, bit 0 (value = 1) of DOMAIN_UPLEVEL, if clear, disables retries with domain components removed. The default value of DOMAIN_UPLEVEL is 0.
4. Vanity domain checking now needs to be performed. This is done by instituting an LDAP search using the LDAP URL specified by the DOMAIN_MATCH_URL MTA option. The value of this option should be set to:

```
ldap:///$B?msgVanityDomain?sub?(msgVanityDomain=$D)
```

$B substitutes the value of the `local.ugldapbasedn` configutil parameter; this is the base of the user tree in the directory. The `LDAP_USER_ROOT` MTA option can be used to override the value of this configutil option specifically for the MTA. (A new $C metacharacter has also been added to substitute in the base of the domain tree.

The actual return value from this search does not matter. What matters is if there is a value to return. If there is a return value the domain is considered to be local, if not it is considered to be nonlocal.

Domain Map Determination of Domain Locality

It is also instructive to note what steps are performed to determine domain locality. The steps are schema-level-specific. In the case of Sun LDAP Schema 1, they are:

1. Convert the domain to a base DN in the domain tree. This is done by converting the domain into a series of dc components and then adding a domain root suffix. The default suffix is obtained from the `service.dcroot` configutil parameter. The default suffix is `o=internet`. So a domain of the form `a.b.c.d` would typically be converted into `dc=a,dc=b,dc=c,dc=d,o=internet`. The `service.dcroot` configutil parameter can be overridden by setting the `LDAP_DOMAIN_ROOT` MTA option.
2. Look for an entry with the base DN found in Step 1 and an object class of either `inetDomain` or `inetDomainAlias`. The search filter used for this purpose can be overridden by setting the `LDAP_DOMAIN_FILTER_SCHEMA1` MTA option which defaults to `(|(objectclass=inetDomain)(objectclass=inetdomainalias))`.
3. Exit with a failure if nothing is found.
4. If the object class of the entry found is `inetDomain` return the value of the `inetDomainBaseDn` attribute for the domain entry. Messaging Server checks for the presence of the `inetDomainBaseDn` attribute. If it is present it is used If it is not present, the entry is assumed to be a domain alias. The MTA option `LDAP_DOMAIN_ATTR_BASEDN` can be used to override the use of `inetDomainBaseDN`.
5. If the object class of the entry found is `inetDomainAlias`, look up the entry referenced by the `aliasedObjectName` attribute. This new entry must have an object class of the `inetDomainBaseDN` attribute. An alternative to the use of `aliasedObjectName` attribute can be specified with the MTA option `LDAP_DOMAIN_ATTR_ALIAS`.
6. If the alias entry lookup is successful return the value of the `inetDomainBaseDn` attribute for the new entry.

In Sun LDAP Schema 2, the action taken is much simpler: The directory is searched for an entry with the object class sunManagedOrganization where the domain appears as a value of either the sunPreferredDomain or associatedDomain attribute. If need be the use of the sunPreferredDomain and associatedDomain attributes for this purpose can be overridden with the respective MTA options LDAP_ATTR_DOMAIN1_SCHEMA2 and LDAP_ATTR_DOMAIN2_SCHEMA2. The search is done under the root specified by the service.dcroot configutil parameter. The service.dcroot configutil parameter can be overridden by setting the LDAP_DOMAIN_ROOT MTA option.

Caching Of Domain Locality Information

Due to the frequency with which domain rewrite operations are performed and the expense of the directory queries (especially the vanity domain check) both negative and positive indications about domains need to be cached. This is implemented with an in-memory open-chained dynamically-expanded hash table. The maximum size of the cache is set by the DOMAIN_MATCH_CACHE_SIZE MTA option (default 100000) and the timeout for entries in the cache is set by the DOMAIN_MATCH_CACHE_TIMEOUT MTA option (default 600 seconds).

Error Handling

Temporary server failures during this process have to be handled carefully, since when they occur, it is impossible to know whether or not a given domain is local. Basically two outcomes are possible in such a case:

1. Return a temporary (4*xx*) error to the client telling it to try the address again later.
2. Accept the address but queue it to the reprocessing channel so it can be retried locally later.

Neither of these options is appropriate in all cases. For example, outcome 1 is appropriate when talking to a remote SMTP relay. But outcome 2 is appropriate when dealing with an SMTP submission from a local user.

While it would be possible in theory to handle temporary failures by using multiple rules with the same pattern, the overhead of repeating such queries, even with a cache in place, is unacceptable. For these reasons, the simple success/fall-through-to-the-next-rule matching model of domain rewriting is inadequate. Instead, a special template, specified by the MTA option DOMAIN_FAILURE, is used in the event of a domain lookup failure. When a $V operation fails, this template replaces the remainder of the current rewrite rule template being processed.

In addition to $V and $Z, several other new metacharacters have been added to the rewrite rule facility:

- $? accepts a numeric argument which, if present, specifies the enhanced status code to be returned by SMTP. For example, "$5001001?message" specifies an enhanced status code value of 5.1.1.
- $1M checks to see whether the internal reprocessing flag was set by the source channel.
- $1N checks to see whether the internal reprocessing flag was cleared by the source channel.
- $1~ performs any pending channel match checks and if the checks have failed terminates processing of the current rewrite rule template successfully.

Pattern for Domain Check Rewrite Rule

This domain check needs to be performed before other rewrite rules have a chance to operate. This ordering is insured by using the special $* on the left hand side in the rule. The $* pattern is checked prior to any other rules.

Putting It All Together

Taking all the machinery described so far into account, the new rewrite rule we need in the imta.cnf is:

```
$*          $E$F$U%$H$V$H@localhost
```

and the value of the DOMAIN_FAILURE MTA option in the option.dat file needs to be:

```
reprocess-daemon$Mtcp_local$1M$1~-error$4000000?Temporary lookup failure
```

In this rewrite rule, localhost is the host name associated with the local channel. The value of the DOMAIN_FAILURE option shown here is the default value so it does not need to appear in option.dat under normal circumstances.

The ordering here is especially tricky. The MTA checks $V after the address is rebuilt but before the route is added. This lets the MTA to change the route in the event of a temporary lookup failure. Pending channel match checks are applied any time the insertion point changes, so the @ after the second $H invokes the check. If the check succeeds, the remainder of the template applies and rewrite processing concludes. If the check fails, the rewrite fails and rewriting continues with the next applicable rewrite rule. If the check cannot be performed due to a temporary failure, template processing continues with the value specified by the DOMAIN_FAILURE MTA option. The value of this template first sets the routing host to reprocess-daemon. Then the template checks to see whether or not the MTA is dealing with a reprocessing channel of some sort or tcp_local. If the MTA is

dealing with such a channel, the rule continues, making the routing host illegal and specifying a temporary failure as the outcome. If the MTA is not dealing with such a channel, the rule is truncated and successfully terminated, thereby rewriting the address to the reprocess channel.

Alias expansion of local addresses

Once an address has been determined to be associated with the local channel it automatically undergoes alias expansion. The alias expansion process examines a number of sources of information, including:

1. The alias file (part of the compiled configuration).
2. The alias database.
3. Alias URLs.

The exact alias sources that are checked and the order in which they are checked depends on the setting of the `ALIAS_MAGIC` MTA option in the `option.dat` file. For direct LDAP, set the option to 8764. This means that the URL specified by the `ALIAS_URL0` MTA option is checked first, then the URL specified by the `ALIAS_URL1` MTA option, then the URL specific by the `ALIAS_URL2` MTA option, and finally, the alias file. The alias database is not checked when this setting is active.

Alias Checking with LDAP URLs

Checking of aliases in LDAP is implemented by specifying two special LDAP URLs as alias URLs. The first of these handles regular users and groups; vanity domains are handled by subsequent alias URLs. The first URL is specified as `ALIAS_URL0`:

```
ALIAS_URL0=ldap:///$V?*?sub?$R
```

The $V Metacharacter

Metacharacter expansion occurs prior to URL lookup. The two metacharacters used in the `ALIAS_URL0` value are `$V` and `$R`.

The `$V` metacharacter converts the domain part of the address into a base DN. This is similar to the initial steps performed by the `$V` rewrite rule metacharacter described previously in the section entitled "Rewrite Rule Machinery." `$V` processing consists of the following steps:

1. Get the base DN for the current domain.

2. Get the canonical domain associated with the current domain. In Sun LDAP Schema 1, the canonical domain name is given by the `inetCanonicalDomainName` attribute of the domain entry if the attribute is present. If the attribute is absent the canonical name is the name constructed in the obvious way from the DN of the actual domain entry. This will differ from the current domain when the current domain is an alias. The name attribute used to store the canonical name may be overridden with the `LDAP_DOMAIN_ATTR_CANONICAL` MTA option in the `option.dat` file.

   In Sun LDAP Schema 2, the canonical name is simply the value of the `SunPreferredDomain` attribute.

3. If the base DN exists, substitute it into the URL in place of the `$V`.

4. Any applicable hosted domain for this entry is now determined. This is done by comparing either the canonical domain (if bit 2 (value = 4) of `DOMAIN_UPLEVEL` is clear) or the current domain (if bit 2 (value = 4) of `DOMAIN_UPLEVEL` is set) with the `service.defaultdomain` configutil parameter. If they do not match, the entry is a member of a hosted domain. The `service.defaultdomain` configutil parameter can be overridden by setting the `LDAP_DEFAULT_DOMAIN` MTA option in the `option.dat` file.

5. If the base DN determination fails, remove a component from the left hand side of the domain and go to Step 1. The substitution fails if no components remain.

`$V` also accepts an optional numeric argument. If it is set to 1 (for example, `$1V`), a failure to resolve the domain in the domain tree will be ignored and the base of the user tree will be returned.

If the attempt to retrieve the domain's base DN succeeds, the MTA also retrieves several useful domain attributes which will be needed later. The names of the attributes retrieved are set by the following MTA options in the `option.dat` file:

- `LDAP_DOMAIN_ATTR_UID_SEPARATOR` (default `domainUidSeparator`)
- `LDAP_DOMAIN_ATTR_SMARTHOST` (default `mailRoutingSmartHost`)
- `LDAP_DOMAIN_ATTR_CATCHALL_ADDRESS` (default `mailDomainCatchallAddress`)
- `LDAP_DOMAIN_ATTR_BLOCKLIMIT` (default `mailDomainMsgMaxBlocks`)
- `LDAP_DOMAIN_ATTR_REPORT_ADDRESS` (default `mailDomainReportAddress`)
- `LDAP_DOMAIN_ATTR_STATUS` (default `inetDomainStatus`)
- `LDAP_DOMAIN_ATTR_MAIL_STATUS` (default `mailDomainStatus`)
- `LDAP_DOMAIN_ATTR_CONVERSION_TAG` (default `mailDomainConversionTag`)

- LDAP_DOMAIN_ATTR_FILTER (default mailDomainSieveRuleSource)
- LDAP_DOMAIN_ATTR_DISK_QUOTA (no default)
- LDAP_DOMAIN_ATTR_MESSAGE_QUOTA (no default)
- LDAP_DOMAIN_ATTR_AUTOREPLY_TIMEOUT (no default)
- LDAP_DOMAIN_ATTR_OPTIN (no default)
- LDAP_DOMAIN_ATTR_PRESENCE (no default)
- LDAP_DOMAIN_ATTR_RECIPIENTLIMIT (no default)
- LDAP_DOMAIN_ATTR_RECIPIENTCUTOFF (no default)
- LDAP_DOMAIN_ATTR_SOURCEBLOCKLIMIT (no default)

Calling a Mapping from a URL

There might be unusual cases where the mapping from domain to base DN is done some other way. In order to accommodate such setups, the URL resolution process has the ability to call an MTA mapping. This is done with a metacharacter sequence of the general form:

$|/*mapping-name*/*mapping-argument*|

The double quotation (“) 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 does not collide with the expected character values used in either the mapping name or argument.

The $R Metacharacter

The $R metacharacter provides an appropriate filter for the URL. The intent is to produce a filter that searches all the attributes that might contain an email address for a particular user or group. The list of attributes to search comes from the configutil parameter local.imta.mailaliases. If this parameter is not set, the local.imta.schematag configutil parameter is examined, and depending on its value, an appropriate set of default attributes is chosen as follows:

sims401	mail, rfc822mailalias
nms41	mail,mailAlternateAddress
ims50	mail,mailAlternateAddress,mailEquivalentAddress

The value of local.imta.schematag can be a comma-separated list. If more than one schema is supported, the combined list of attributes with duplicates eliminated is used. The LDAP_SCHEMATAG MTA option can be used to override the setting of local.imta.schematag specifically for the MTA.

Additionally, the filter searches not only for the address that was originally supplied, but also for an address with the same local part but the domain that was actually found in the domain tree, which was saved in Step 2 in the section entitled "The $V Metacharacter." The iterative nature of the domain tree lookup means the two addresses may be different. This additional check is controlled by bit 1 (value = 2) of the DOMAIN_UPLEVEL MTA option in the option.dat file. Setting the bit enables the additional address check. The default value of DOMAIN_UPLEVEL is 0.

For example, suppose that the domain siroe.com appears in the domain tree. Assuming Sun LDAP Schema 1 is in force, a lookup of the address

```
u@host1.siroe.com
```

The filter that results from the expansion of $R and an ims50 schematag looks like:

```
(|(mail=u@siroe.com)
     (mail=u@host1.siroe.com)
     (mailAlternateAddress=u@siroe.com)
     (mailAlternateAddress=u@host1.siroe.com)
     (mailEquivalentAddress=u@siroe.com)
     (mailEquivalentAddress=u@host1.siroe.com))
```

If, on the other hand, DOMAIN_UPLEVEL was set to 1 rather than 3, the filter would be:

```
(|(mail=u@host1.siroe.com)
     (mailAlternateAddress=u@host1.siroe.com)
     (mailEquivalentAddress=u@host1.siroe.com))
```

The ALLOW_UNQUOTED_ADDRS_VIOLATE_RFC2798 MTA Option

Some MTAs have been exceptionally lax about address quoting and canonicalization. In particular, they would allow illegal addresses such as a..b@siroe.com. Even worse, they failed to add the necessary quotes to make the address into the legal "a..b"@siroe.com prior to searching for it in the directory.

The ALLOW_UNQUOTED_ADDRS_VIOLATE_RFC2798 MTA option in the option.dat file accommodates this particular violation of the standards. If set to 1 it will add additional filter terms to search on the syntactically invalid dequoted form of quoted addresses. For example, a search for the address "a..b"@siroe.com might produce a filter of the form:

```
(|(mail="a..b"@siroe.com)
     (mail=a..b@siroe.com)
     (mailAlternateAddress="a..b"@siroe.com)
     (mailAlternateAddress=a..b@siroe.com)
     (mailEquivalentAddress="a..b"@siroe.com)
     (mailEquivalentAddress=a..b@siroe.com))
```

This option does not solve the problems caused by the use of illegal addresses. The relevant email and directory standards are specific about what is allowed in the local part of an address. When various messaging components are presented with syntactically illegal addresses, they may behave in different ways. They may quote such addresses to make them legal, they may pass them through unchanged, they may reject them, or they may do something entirely unexpected. Therefore, if end users are given such an illegal address they are likely to find it does not work when it hits other messaging systems provided by other vendors.

Determining the Attributes to Fetch

If the URL specifies an * for the list of attributes to return, we replace the asterisk with the list of attributes the MTA is able to use.

Handling LDAP Errors

At this point the resulting URL is used to perform an LDAP search. If an LDAP error of some kind occurs, processing terminates with a temporary failure indication (4xx error in SMTP). If the LDAP operation succeeds but fails to produce a result, the catchall address attribute for the domain retrieved from the LDAP_DOMAIN_ATTR_CATCHALL_ADDRESS MTA option is checked. If it is set, its value replaces the current address.

If no catchall address attribute is set the smarthost attribute for the domain retrieved from the LDAP_DOMAIN_ATTR_SMARTHOST MTA option is checked. If it is set, an address of the form

@smarthost:*user@domain*

is created and alias processing terminates successfully with this result. Additionally, the conversion tag for the domain obtained from the LDAP_DOMAIN_ATTR_CONVERSION_TAG MTA option will, if present, be attached to the address so that conversions can be done prior to forwarding to the smarthost. If no catchall address or smarthost exists for the domain, processing of this alias URL terminates unsuccessfully.

Sanity Checks on the LDAP Result

After the LDAP search has returned a result, it is checked to verify that there is only one entry in it. If there are more than one, each entry is checked to see if it has the right object class for a user or a group, a non-deleted status, and for users, a UID. Entries that do not pass this check are ignored. If the list of multiple entries is reduced to one by this check, processing proceeds. If not, a duplicate or ambiguous directory error is returned.

Support for Vanity Domains

The `ALIAS_URL0` check is for a conventional user or a user in a hosted domain. If this fails a vanity domain check is also made. This is done with the following alias URL:

```
ALIAS_URL1=ldap:///$B?*?sub?(&(msgVanityDomain=$D)$R)
```

Support for Catchall Addresses

Finally, a check for a catchall address of the form *@host* needs to be made in the `mailAlternateAddress` attribute. This form of wildcarding is allowed in both hosted and vanity domains, so the proper alias URL for it is:

```
ALIAS_URL2=ldap:///$1V?*?sub?(mailAlternateAddress=@$D)
```

Processing the LDAP Result

LDAP alias result processing is done in a number of order-dependent stages. These stages are described in the following sections.

Object Class Check

If the alias search succeeds, the object class of the entry is checked to make sure it contains an appropriate set of object classes for a user or a group. The possible sets of required object classes for users and groups is normally determined by what schemata are active. This is determined by the `local.imta.schematag` setting.

Table 9-1 shows the user and group object classes that result from various `schematag` values.

Table 9-1 Object Classes Resulting from Various `schematag` Values

`schematag`	**User Object Classes**	**Group Object Classes**
sims40	`inetMailRouting+inetmailuser`	`inetMailRouting+inetmailgroup`

Table 9-1 Object Classes Resulting from Various schematag Values

schematag	User Object Classes	Group Object Classes
nms41	mailRecipient + nsMessagingServerUser	mailGroup
ims50	inetLocalMailRecipient+inetmailuser	inetLocalMailRecipient+inetmailgroup

The information in this table, like the rest of schema tag handling, is hard coded. However, there are also two MTA options in the option.dat file, LDAP_USER_OBJECT_CLASSES and LDAP_GROUP_OBJECT_CLASSES, which can be set to specify different sets of object classes for users and groups respectively.

For example, a schema tag setting of ims50,nms41 would be equivalent to the following option settings:

```
LDAP_USER_OBJECT_CLASSES=inetLocalMailRecipient+inetmailuser,
mailRecipient+nsMessagingServerUser
```

```
LDAP_GROUP_OBJECT_CLASSES=inetLocalMailRecipient+inetmailgroup,mailGroup
```

The LDAP result is simply ignored if it does not have a correct set of object classes appropriate for a user or a group. The MTA also determines if it is dealing with a user or a group and saves this information. This saved information will be used repeatedly later.

Note that the object class settings described here are also used to construct an actual LDAP search filter that can be used to check to see that an entry has the right object classes for a user or a group. This filter is accessible through the $K metacharacter. It is also stored internally in the MTA's configuration for use by channel programs and is written to the MTA option file, option.dat, as the LDAP_UG_FILTER option when the command imsimta cnbuild -option is used. This option is only written to the file. The MTA never reads it from the option file.

Entry Status Checks

Next the entry's status is checked. There are two status attributes, one for the entry in general and another specifically for mail service.

Table 9-2 shows the general and mail-specific user or group attributes in the schematag entry to check against depending on what schemata are in effect

Table 9-2 Attributes to Check Against

schematag	Type	General	Mail-specific
sims40	users	inetsubscriberstatus	mailuserstatus
sims40	groups	none	inetmailgroupstatus
nms41	users	none	mailuserstatus
nms41	groups	none	none
Messaging Server 5.0	users	inetuserstatus	mailuserstatus
Messaging Server 5.0	groups	none	inetmailgroupstatus

If necessary the LDAP_USER_STATUS and LDAP_GROUP_STATUS MTA options in the option.dat file can be used to select alternate general status attributes for users and groups respectively. The mail-specific user and group status attributes are controlled by the LDAP_USER_MAIL_STATUS and LDAP_GROUP_MAIL_STATUS MTA options.

Another factor that plays into this are the statuses for the domain itself (LDAP_DOMAIN_ATTR_STATUS and LDAP_DOMAIN_ATTR_MAIL_STATUS). All in all there are four status attributes. They are combined by considering them in the following order:

1. Domain status
2. Domain mail status
3. User or group status
4. Mail user or group status

The first of these that specifies something other than “active” takes precedence over all the others. The other permissible status values are “inactive,” “deleted,” “removed,” “disabled,” “hold,” and “overquota.” “Hold,” “disabled,” and “removed” statuses may only be specified for mail domains, mail users, or mail groups. “Overquota” status can only be specified as a mail domain or mail user status.

All statuses default to “active” if a particular status attribute is not present. Unknown status values are interpreted as “inactive.”

When the four statuses are combined, the following statuses for a user or group are possible: “active,” “inactive,” “deleted,” “removed,” “disabled,” “hold,” and “overquota.” Active status causes alias processing to continue. Inactive or overquota status results in immediate rejection of the address with a 4xx

(temporary) error. Deleted, removed, and disabled statuses results in immediate rejection of the address with a 5xx (permanent) error. Hold status is treated as active as far as status handling is concerned but it sets an internal flag so that when delivery options are considered later on, any options that are there are overridden with an option list containing a single “hold” entry.

UID Check

The next step is to consider the entry's UID. UIDs are used for a variety of purposes and must be part of all user entries and may be included in group entries. A user entry without a UID is ignored and processing of this alias URL terminates unsuccessfully. UIDs for entries in a hosted domain can consist of the real UID, a separator character, and then a domain. The MTA only wants the real UID so the rest is removed if present using the domain separator character obtained with the `LDAP_DOMAIN_ATTR_UID_SEPARATOR` MTA option in the `option.dat` file.

In the unlikely event that an attribute other than `uid` is used to store UIDs, the `LDAP_UID` MTA option can be used to force use of a different attribute.

Message Capture

Next the LDAP attribute used to specify one or more message capture addresses is checked. The attribute used for this purpose must be specified with the `LDAP_CAPTURE` MTA option. There is no default. Values of this attribute are treated as addresses and a special “capture” notification is generated and sent to these address containing the current message as an attachment. Additionally, the capture addresses are used to seed the address reversal cache in the likely event the address will subsequently appear as an envelope from: address.

Seeding the Reversal Cache

Next the primary address and any aliases attached to the user entry are considered. This information is used to seed the address reversal cache. It plays no part in the current address translation process. First, the primary address, personal name, recipient limit, recipient cutoff, and source block limit attributes are considered. The primary address is normally stored in the “mail” attribute; another attribute can be specified by setting the `LDAP_PRIMARY_ADDRESS` MTA option appropriately. (The primary address reverses to itself, of course.) There is no default attribute for any of the other attributes. If you want to use them, you must specify them with the `LDAP_PERSONAL_NAME`, `LDAP_RECIPIENTLIMIT`, `LDAP_RECIPIENTCUTOFF`, and `LDAP_SOURCEBLOCKLIMIT` MTA options. The corresponding domain-level recipient limit, recipient cutoff, and source block limit attributes are also considered at this point. User-level settings completely override any domain-level setting.

Next, any secondary addresses are considered and a cache entry is made for each one. There are two sorts of secondary addresses: Those that undergo address reversal and those that do not. Both must be considered in order to properly seed the address reversal cache because of the need to check for message capture requests in all cases.

Secondary addresses that undergo reversal are normally stored in the `mailAlternateAddress` attribute. Another attribute can be specified by setting the `LDAP_ALIAS_ADDRESSES` MTA option. Secondary addresses that do not undergo reversal are normally stored in the `mailEquivalentAddress` attribute. Another attribute can be specified with the `LDAP_EQUIVALENCE_ADDRESSES` MTA option.

Mail Host and Routing Address

It is now time to consider the `mailhost` and `mailRoutingAddress` attributes. The actual attributes considered can be overridden with the `LDAP_MAILHOST` and `LDAP_ROUTING_ADDRESS` MTA options, respectively. These attributes work together to determine whether or not the address should be acted on at this time or forwarded to another system.

The first step is to decide whether or not `mailhost` is meaningful for this entry. A preliminary check of the delivery options active for the entry is done to see whether or not the entry is mailhost-specific. If it is not, `mailhost` checking is omitted. See the description of "Delivery Options Processing" on page 207, and the # flag in particular, to understand how this check is done.

In the case of a user entry, the `mailhost` attribute must identify the local system in order to be acted on. The `mailhost` attribute is compared with the value of the `local.hostname` configutil parameter and against the list of values specified by the `local.imta.hostnamealiases` configutil parameter. The `mailhost` attribute is deemed to identify the local host if any of these match.

A successful match means that the alias can be acted on locally and alias processing continues. An unsuccessful match means that the message needs to be forward to the mailhost to be acted on. A new address of the form

@mailhost:*user@domain*

is constructed and becomes the result of the alias expansion operation.

The handling of a missing `mailhost` attribute is different depending on whether the entry is a user or a group. In the case of a user, a mailhost is essential, so if no `mailhost` attribute is present a new address of the form

@smarthost:*user@domain*

is constructed using the smart host for the domain determined by the LDAP_DOMAIN_ATTR_SMARTHOST MTA option. An error is reported if no smart host exists for the domain.

Groups, on the other hand, do not require a mailhost, so a missing mailhost is interpreted as meaning that the group can be expanded anywhere. So alias processing continues.

The mailRoutingAddress attribute adds one final wrinkle. If it is present, alias processing terminates with the mailRoutingAddress as the result. However, if a mailhost is present, it is added to the mailRoutingAddress as a source route.

Miscellaneous Attribute Support

Next the mailMsgMaxBlocks attribute is considered. First it is minimized with the domain block limit returned from the LDAP_DOMAIN_ATTR_BLOCKLIMIT MTA option. If the size of the current message is known to exceed the limit, alias processing terminates with a size-exceeded error. If the size is not known or does not exceed the limit, the limit is nevertheless stored and will be rechecked when the message itself is checked later. The use of mailMsgMaxBlocks can be overridden with the LDAP_BLOCKLIMIT MTA option.

Next a number of attributes are accessed and saved. Eventually these will be written into the queue file entry for use by the ims_master channel program, which will then use them to update the store's user information cache. If the attributes are not found for individual users, domain-level attributes can be used to set defaults.

This step is skipped if the LDAP entry is for a group rather than a user or if the LDAP entry came from the alias cache and not from the LDAP directory. The logic behind the latter criteria is that frequent updates of this information are unnecessary and using the alias cache offers a reasonable criteria for when updates should be done. The names of the attributes retrieved are set by various MTA options.

Table 9-3 shows the MTA options which set the retrieved disk quota and message quota attributes.

Table 9-3 MTA Options Which Set the Retrieved Disk Quota and Message Quota Attributes

MTA option	Attribute
LDAP_DISK_QUOTA	mailQuota
LDAP_DOMAIN_ATTR_DISK_QUOTA	mailQuota
LDAP_DOMAIN_ATTR_MESSAGE_QUOTA	mailMsgQuota
LDAP_MESSAGE_QUOTA	mailMsgQuota

Next a number of attributes are stored for possible use in conjunction with metacharacter substitutions later.

Table 9-4 shows the MTA options, the default attribute, and metacharacters.

Table 9-4 MTA Options, Default Attributes, and Metacharacters

MTA Option	Default Attribute	Metacharacters
LDAP_PROGRAM_INFO	mailProgramDeliveryInfo	$P
LDAP_DELIVERY_FILE	mailDeliveryFileURL	$F
LDAP_SPARE_1	no default	$1E $1G $E
LDAP_SPARE_2	no default	$2E $2G $G
LDAP_SPARE_3	no default	$3E $3G
LDAP_SPARE_4	no default	$4E $4G
LDAP_SPARE_5	no default	$5E $5G

Spare slots for additional attributes are included so that you can use them to build customized address expansion facilities.

Next any values associated with the mailConversionTag attribute are added to the current set of conversion tags. The name of this attribute can be changed with the LDAP_CONVERSION_TAG MTA option. If any values were associated with the domain's mailDomainConversionTag attribute, they are attached as well.

Delivery Options Processing

Next the mailDeliveryOption attribute is checked. The name of this attribute can be changed with the LDAP_DELIVERY_OPTION MTA option. This is a multi valued option and its values determine the addresses produced by the alias translation process. Additionally, the permissible values are different for users and groups. Common permissible values are program, forward, and hold." User-only values are mailbox, native, unix, and autoreply. The group-only values are members, members_offline, and file.

The conversion of the mailDeliveryOption attribute into appropriate addresses is controlled by the DELIVERY_OPTIONS MTA option. This option not only specifies what addresses are produced by each permissible mailDeliveryOption value, but also what the permissible mailDeliveryOption values are and whether or not each one is applicable to users, groups, or both.

The value of this option consists of a comma-separated list of deliveryoption=template pairs, each pair with one or more optional single character prefixes.

The default value of the DELIVERY_OPTIONS option is:

```
DELIVERY_OPTIONS=*mailbox=$M%$\\$2I$_+$2S@ims-ms-daemon,
     &members=*,
     *native=$M@native-daemon,
     /hold=@hold-daemon:$A,
     *unix=$M@native-daemon,
     &file=+$F@native-daemon,
     &@members_offline=*,
     program=$M%$P@pipe-daemon,
     #forward=**,
     *^!autoreply=$M+$D@bitbucket
```

Each delivery option corresponds to a possible mailDeliveryOption attribute value and the corresponding template specifies the resulting address using the same metacharacter substitution scheme used by URL processing.

Table 9-5 shows the single character prefixes available for the DELIVERY_OPTIONS options.

Table 9-5 Single-Character Prefixes for options in the DELIVERY_OPTIONS MTA option.

Character Prefix	Description
*	Delivery option applies to users.
&	Delivery option applies to groups.
$	Sets a flag saying expansion of this user or group is to be deferred.
^	Sets a flag saying that the vacation start and end times should be checked to see if this delivery option really is in effect.
#	Sets a flag saying expansion of this delivery option does not need to take place on the entry's designated mailhost.
/	Sets a flag that causes all addresses produced by this delivery option to be held. Message files containing these recipient addresses will have a .HELD extension.
!	Sets a flag that says that autoreply operations should be handled internally by the MTA. It only makes sense to use this prefix on an autoreply delivery option. The option's value should direct the message to the bitbucket channel

If neither * nor & are present, the delivery option is taken to apply to both users and groups.

Additional Metacharacters for Use in Delivery Options

Several additional metacharacters have been added to support this new use of the MTA's URL template facility. These include:

Table 9-6 shows additional metacharacters and their descriptions for use in delivery options.

Table 9-6 Additional Metacharacters for Use in Delivery Options

Metacharacter	Description
$\\	Force subsequent text to lower case.
$^	Force subsequent text to upper case.
$_	Perform no case conversion on subsequent text.
$*n*A	Insert the *n*th character of the address. The first character is character 0. The entire address is substituted if *n* is omitted. This is intended to be used to construct autoreply directory paths.
$D	Insert the domain part of the address.
$*n*E	Insert the value of the *n*th spare attribute. If *n* is omitted the first attribute is used.
$F	Insert the name of the delivery file (`mailDeliveryFileURL` attribute).
$*n*G	Insert the value of the *n*th spare attribute. If *n* is omitted the second attribute is used.
$*n*H	Insert the *n*th component of the domain from the original address counting from 0. The default is 0 if *n* is omitted.
$*n*I	Insert hosted domain associated with alias. This metacharacter accepts an integer parameter *n* whose semantics are described in Table 9-7.
$*n*J	Insert the *n*th part of the host domain counting from 0. The default for *n* is 0.
$K	Insert an LDAP filter that matches the object classes for a user or group. See the description of the `LDAP_UG_FILTER` output-only MTA option.
$L	Insert the local part of the address.
$M	Insert the UID associated with the current alias.
$P	Insert the program name (`mailProgramDeliveryInfo` attribute).
$*n*S	Insert subaddress associated with the current address. This metacharacter accepts an integer parameter n whose semantics are described in Table 9-7.
$*n*U	Insert the nth character of the dequoted form of the mailbox part of the current address. The first character is character 0. The entire dequoted mailbox is substituted if n is omitted.
$*n*X	Insert the *n*th component of the mailhost. The entire mailhost is inserted if *n* is omitted.

Table 9-7 shows how the integer parameter modifies the behavior of the $nI and $nS metacharacters.

Table 9-7 Integers Controlling Behavior Modification of the $nI and $nS Metacharacters

Integer	Description of Behavior
0	Fail if no value is available (default).
1	Insert value if one is available. Insert nothing if not.
2	Insert value if one is available. Insert nothing and delete preceding character is one is not (this particular behavior is needed by the `ims-ms` channel).
3	Insert value if one is available. Insert nothing and ignore following character if one is not.

In addition to the metacharacters, Table 9-8 shows two special template strings.

Table 9-8 Special Template Strings

Special Template String	Description
*	Perform group expansion. This value is not valid for user entries.
**	Expand the attribute named by the `LDAP_FORWARDING_ADDRESS` MTA option. This defaults to `mailForwardingAddress`.

With group expansion, for example, if a user's `mailDeliveryOption` value is set to `mailbox`, we form a new address consisting of the stripped UID, a percent sign followed by the hosted domain if one is applicable, a plus sign followed by the subaddress if one was specified, and finally `@ims-ms-daemon`.

Delivery Option Defaults

If the list of active delivery options is empty at this point, the first option on the list (usually mailbox) is activated for users and the second option on the list (usually members) is activated for groups.

Start and End Date Checks

Start and end dates are checked after the delivery option list has been read. There are two attributes whose names are controlled by the LDAP_START_DATE (default vacationStartDate) and LDAP_END_DATE (default vacationEndDate) MTA options, respectively. If one or more of the active delivery options specified the ^ prefix character, the values of these options are checked against the current date. If the current date is outside the range specified by these options, the delivery options with the ^ prefix are removed from the active set.

Optin and Presence Attributes

The LDAP_OPTIN MTA option can be used to specify an LDAP attribute containing a list of spam filter opt-in values. If the option is specified and the attribute is present, it is appended to the current spam filter opt-in list. Any values set by the domain level attribute set by the LDAP_DOMAIN_ATTR_OPTIN MTA option will also be appended to the list.

The LDAP_PRESENCE MTA option can be used to specify a URL that can be resolved to return presence information about the user. If the option is specified and the attribute is present, its value is saved for possible use in conjunction with Sieve presence tests. The domain level attribute set by the LDAP_DOMAIN_ATTR_PRESENCE MTA option is used as source for this URL if no value exists for the user entry.

Sieve Filter Handling

Next the mailSieveRuleSource attribute is checked for a Sieve filter that applies to this entry. If this attribute exists, it is parsed and stored at this point. The two possible forms for the value of this attribute are a single value that contains a complete Sieve script or multiple values where each value contains a piece of a Sieve script. The latter form is produced by the web filter construction interface. Special code is used to order the values and glue them together properly.

The use of the mailSieveRuleSource attribute specifically can be overridden by using the LDAP_FILTER MTA option.

Deferred Processing Control

Next the mailDeferProcessing attribute is checked. This attribute can be changed by using the LDAP_REPROCESS MTA option. Processing continues normally if this attribute exists and is set to no. But if this attribute is set to yes and the current source channel is not the reprocess channel, expansion of this entry is aborted and the original *user@domain* address is simply queued to the reprocess channel. If this attribute does not exist, the setting of the deferred processing character prefix associated with delivery options processing is checked. (See the section Delivery

Options Processing for examples.) Processing is deferred if it is set. If it is not set, the default for users is no. The default for groups is controlled by the MTA option DEFER_GROUP_PROCESSING, which defaults to 1 (yes). Alias processing concludes at this point for user entries.

Group Expansion Attributes

A number of additional attributes are associated with group expansion and must be dealt with at this point. The names of these attributes are all configurable via various MTA options.

Table 9-9 lists the default attribute names, the MTA option to set the attribute name, and the way the attribute is processed by the MTA. The ordering of the elements in the table shows the order in which the various group attributes are processed. This ordering is essential for correct operation.

Table 9-9 Group Expansion Attributes

Default Attribute	MTA option to Set Attribute Name	How the Attribute is Processed
mgrpMsgRejectAction	LDAP_REJECT_ACTION	Single valued attribute that controls what happens if any of the subsequent access checks fail. Only one value is defined: TOMODERATOR, which if set instructs the MTA to redirect any access failures to the moderator specified by the mgrpModerator attribute. The default (and any other value of this attribute) causes an error to be reported and the message rejected.
mailRejectText	LDAP_REJECT_TEXT	The first line of text stored in the first value of this attribute is saved. This text will be returned if any of the following authentication attributes cause the message to be rejected. This means the text can appear in SMTP responses so value has to be limited to US-ASCII to comply with current messaging standards.

Table 9-9 Group Expansion Attributes *(Continued)*

Default Attribute	MTA option to Set Attribute Name	How the Attribute is Processed
mgrpBroadcasterPolicy	LDAP_AUTH_POLICY	Specifies level of authentication needed to send to the group. Possible tokens are SMTP_AUTH_REQUIRED or AUTH_REQ, both of which mean that the SMTP AUTH command must be used to identify the sender in order to send to the group; PASSWORD_REQUIRED, PASSWD_REQUIRED, or PASSWD_REQ, all of which mean the password to the list specified by the mgrpAuthPassword attribute must appear in an Approved: header field in the message; OR, which changes the OR_CLAUSES MTA option setting to 1 for this list; AND, which changes the OR_CLAUSES MTA option setting to 0 for this list; and NO_REQUIREMENTS, which is a no-op. Multiple values are allowed and each value can consist of a comma-separated list of tokens. If SMTP AUTH is called for it also implies that any subsequent authorization checks will be done against the email address provided by the SASL layer rather than the MAIL FROM address.
mgrpAllowedDomain	LDAP_AUTH_DOMAIN	Domains allowed to submit messages to this list. Can be multi valued.
mgrpDisallowedDomain	LDAP_CANT_DOMAIN	Domains not allowed to submit messages to this list. Can be multi valued.
mgrpAllowedBroadcaster	LDAP_AUTH_URL	URL identifying mail addresses allowed to send mail to this group. Can be multi valued. Each URL is expanded into a list of addresses and each address is checked against the current envelope from address. A match means the message is allowed.
mgrpDisallowedBroadcaster	LDAP_CANT_URL	URL identifying mail addresses not allowed to send mail to this group. Can be multi valued. Each URL is expanded into a list of addresses and each address is checked against the current envelope from address. A match means the message is not allowed.
mgrpMsgMaxSize	LDAP_ATTR_MAXIMUM_MESSAGE_SIZE	Maximum message size in bytes that can be sent to the group. This attribute is obsolete but still supported for backwards compatibility; the new mailMsgMaxBlocks attribute should be used instead.

Table 9-9 Group Expansion Attributes *(Continued)*

Default Attribute	MTA option to Set Attribute Name	How the Attribute is Processed
mgrpAuthPassword	LDAP_AUTH_PASSWORD	Specifies a password needed to post to the list. The presence of this attribute forces a reprocessing pass. As the message is enqueued to the reprocessing channel, the password is taken from the header and placed in the envelope. Then while reprocessing, the password is taken from the envelope and checked against this attribute. Additionally, only passwords that actually are used are removed from the header field.
mgrpModerator	LDAP_MODERATOR_URL	The list of URLs given by this attribute to be expanded into a series of addresses. The interpretation of this address list depends on the setting of the LDAP_REJECT_ACTION MTA option. If LDAP_REJECT_ACTION is set to TOMODERATOR, this attribute specifies the moderator address(es) the message is to be sent to should any of the access checks fail. If LDAP_REJECT_ACTION is missing or has any other value, the address list is compared with the envelope from address. Processing continues if there is a match. If there is no match, the message is again sent to all of the addresses specified by this attribute. Expansion of this attribute is implemented by making the value of this attribute the list of URLs for the group. Any list of RFC822 addresses or DNs associated with the group is cleared, and the delivery options for the group are set to members. Finally, subsequent group attributes listed in this table are ignored.
mgrpDeliverTo	LDAP_GROUP_URL1	List of URLs which, when expanded, provides a list of mailing list member addresses.
memberURL	LDAP_GROUP_URL2	Another list of URLs which, when expanded, provides another list of mailing list member addresses.
uniqueMember	LDAP_GROUP_DN	List of DNs of group members. DNs may specify an entire subtree. Unique member DNs are expanded by embedding them in an LDAP URL. The exact URL to use is specified by the GROUP_DN_TEMPLATE MTA option. The default value for this option is: ldap:///$A?mail?sub?(mail=*) $A specified the point where the uniqueMember DN is inserted.
mgrpRFC822MailMember	LDAP_GROUP_RFC822	Mail addresses of members of this list.
rfc822MailMember	LDAP_GROUP_RFC822	rfc822MailMember is supported for backwards compatibility. Either rfc822MailMember or mgrpRFC822MailMember, but not both, can be used in any given group.

Table 9-9 Group Expansion Attributes *(Continued)*

Default Attribute	MTA option to Set Attribute Name	How the Attribute is Processed
mgrpErrorsTo	LDAP_ERRORS_TO	Sets the envelope originator (MAIL FROM) address to whatever the attribute specifies.
mgrpAddHeader	LDAP_ADD_HEADER	Turns the headers specified in the attribute into header trimming ADD options.
mgrpRemoveHeader	LDAP_REMOVE_HEADER	Turns the headers specified into header trimming MAXLINES=-1 options.
mgrpMsgPrefixText	LDAP_PREFIX_TEXT	Adds the specified text to the beginning of the message text, if any.
mgrpMsgSuffixText	LDAP_SUFFIX_TEXT	Adds the specified text to the ending of the message text, if any.

One final attribute is checked in the special case of group expansion as part of an SMTP EXPN command: mgmanMemberVisibility or expandable. The LDAP_EXPANDABLE MTA option can be used to select different attributes to check. Possible values are: anyone, which means that anyone can expand the group, all or true, which mean that the user has to successfully authenticate with SASL before expansion will be allowed, and none, which means that expansion is not allowed. Unrecognized values are interpreted as none. If the attribute is not present, the EXPANDABLE_DEFAULT MTA option controls whether the expansion is allowed.

Alias entries are cached in a fashion similar to domain entries. The MTA options controlling the alias cache are ALIAS_ENTRY_CACHE_SIZE (default 1000 entries) and ALIAS_ENTRY_CACHE_TIMEOUT (default 600 seconds). The entire LDAP return value for a given alias is retained in the cache.

Negative caching of alias entries is controlled by the ALIAS_ENTRY_CACHE_NEGATIVE MTA option. A nonzero value enables caching of alias match failures. A zero value disables it. Negative caching of alias entries is disabled by default. The theory is that repeated specification of an invalid address is unlikely to occur very often in practice. In addition, negative caching may interfere with timely recognition of new users added to the directory. However, sites should consider re enabling negative caching of aliases in situations where vanity domains are heavily used. The search performed by the URL specified in ALIAS_URL0 is less likely to be successful.

Address Reversal

Address reversal with direct LDAP starts with a USE_REVERSE_DATABASE value of 4, which disables the use of any reverse database. It then builds on the routing facilities previously discussed. In particular, in the previous version, it began with a reverse URL specification of the form:

```
REVERSE_URL=ldap:///$V?mail?sub?$Q
```

The $V metacharacter has already been described in the context of alias URLs However, the $Q metacharacter, while quite similar in function to the $R metacharacter used in alias URLs, is specifically intended for use in address reversal. Unlike $R, it produces a filter that searches attributes containing addresses that are candidates for address reversal. The list of attributes to search comes from the MTA option LDAP_MAIL_REVERSES. If this option is not set, the local.imta.schematag configutil parameter is examined, and depending on its value, an appropriate set of default attributes is chosen.

Table 9-10 shows the local.imta.schematag values and the default attributes chosen.

Table 9-10 local.imta.schematag Values and Attributes

Schema Tag Value	Attributes
sims40	mail,rfc822mailalias
nms41	mail,mailAlternateAddress
ims50	mail,mailAlternateAddress

The use of $Q is no longer appropriate, however. In order for message capture and other facilities to work correctly, address reversal has been enhanced to pay attention to the attribute that matched in addition to the fact that a match occurred. This means that $R should be used to specify the filter instead of $Q. Additionally, the $N metacharacter has been added, which returns a list of the attributes of interest to address reversal. The resulting option value is:

```
REVERSE_URL=ldap:///$V?$N?sub?$R
```

As always, local.imta.schematag can be a comma-separated list. If more than one schema is supported, the combined list of attributes, with duplicates eliminated, is used.

Additionally, the filter searches not only for the address that was originally supplied, but also for an address with the same local part but the domain that was actually found in the domain tree (which was saved in Step 2 on page 192. The iterative nature of the domain tree lookup means the two addresses may be different.

For example, suppose that the domain `siroe.com` appears in the domain tree and the MTA looks the address:

```
u@host1.siroe.com
```

The filter that results from the expansion of $R and an ims50 schematag will be something like:

```
(|(|(mail=u@siroe.com)(mail=u@host1.siroe.com))
    (|(mailAlternateAddress=u@siroe.com)
     (mailAlternateAddress=u@host1.siroe.com)))
```

Note that the reverse URL explicitly specifies the attribute containing the canonicalized address. Normally this will be the mail attribute.

After the URL is constructed an LDAP search is performed. If the search is successful the first attribute value returned replaces the original address. Unsuccessful searches or errors leave the original address unchanged.

Due to the frequency with which address reversal operations are performed, especially given the number of addresses that can appear in a message header, and the expense of the directory queries involved, both negative and positive results need to be cached. This is implemented with an in-memory, open-chained, dynamically-expanded hash table. The maximum size of the cache is set by the `REVERSE_ADDRESS_CACHE_SIZE` MTA option (default 100000) and the timeout for entries in the cache is set by the `REVERSE_ADDRESS_CACHE_TIMEOUT` MTA option (default 600 seconds). The cache actually stores addresses themselves, not the LDAP URLs and LDAP results.

Asynchronous LDAP Operations

Asynchronous lookups avoid the need to store an entire large LDAP result in memory, which can cause performance problems in some cases. The MTA provides the ability to perform various types of lookups done by the MTA asynchronously.

Use of asynchronous LDAP lookups is controlled by the MTA option LDAP_USE_ASYNC. This option is a bit-encoded value. Each bit, if set, enables the use of asynchronous LDAP lookups in conjunction with a specific use of LDAP within the MTA.

Table 9-11 shows the bit and value settings for the LDAP_USE_ASYNC MTA option in the option.dat file.

Table 9-11 Settings for the LDAP_USE_ASYNC MTA Option

Bit	Value	Specific Use of LDAP
0	1	LDAP_GROUP_URL1 (mgrpDeliverTo) URLs
1	2	LDAP_GROUP_URL2 (memberURL) URLs
2	4	LDAP_GROUP_DN (UniqueMember) DNs
3	8	auth_list, moderator_list, sasl_auth_list, and sasl_moderator_list nonpositional list parameter URLs
4	16	cant_list, sasl_cant_list nonpositional list parameter URLs
5	32	originator_reply nonpositional list parameter URLs
6	64	deferred_list, direct_list, hold_list, nohold_list nonpositional list parameter URLs
7	128	username_auth_list, username_moderator_list, username_cant_list nonpositional list parameter URLs
8	256	alias file list URLs
9	512	alias database list URLs
10	1024	LDAP_CANT_URL (mgrpDisallowedBroadcaster) outer level URLs
11	2048	LDAP_CANT_URL inner level URLs
12	4096	LDAP_AUTH_URL (mgrpAllowedBroadcaster) outer level URLs
13	8192	LDAP_AUTH_URL inner level URLs
14	16384	LDAP_MODERATOR_URL (mgrpModerator) URLs

The default value of the LDAP_USE_ASYNC MTA option is 0, which means that asynchronous LDAP lookups are disabled by default.

Settings Summary

In order to enable direct LDAP, the following MTA options need to be set:

```
ALIAS_MAGIC=8764
ALIAS_URL0=ldap:///$V?*?sub?$R
USE_REVERSE_DATABASE=4
USE_DOMAIN_DATABASE=0
REVERSE_URL=ldap:///$V?mail?sub?$Q
```

If vanity domains are to be supported, the following additional options must 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)
```

Note that the last of these options also handle the case of wild carded local parts in hosted as well as vanity domains. If wild carded local part support is desired but vanity domain support is not, the following option should be used instead:

```
ALIAS_URL1=ldap:///$V?*?sub?&(mailAlternateAddress=@$D)
```

The `filter ssrd:$A` clause needs to be removed from the ims-ms channel definition in the MTA configuration file (`imta.cnf`).

Chapter 10

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:

Compiling the MTA Configuration

Whenever an MTA configuration file such as `imta.cnf`, `mappings`, `aliases`, or `option.dat` is modified, you must recompile the configuration using the `imsimta refresh` command (see the *Sun Java System Messaging Server Administration Reference*). This command compiles the configuration files into a single image in shared memory (on UNIX) or a dynamic link library (NT).

The compiled configuration has a static and dynamic reloadable part. If the dynamic part is changed, and you run an `imsimta reload`, a running program will reload the dynamic data. The dynamic parts are mapping tables, aliases, and lookup tables.

The main reason for compiling configuration information is performance. Another feature of using a compiled configuration is that configuration changes can be tested more conveniently since the configuration files themselves are not “live” when a compiled configuration is in use.

Whenever a component of the MTA (such as a channel program) must read the configuration file, it first checks to see if a compiled configuration exists. If it does, the image is attached to the running program. If the image attach operation fails, the MTA falls back on the old method of reading the text files instead.

The MTA Configuration File

The primary MTA configuration file is `imta.cnf`. By default, this file is found at *msg_svr_base*`/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. The system will typically work well using default `imta.cnf`.

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 11, “Configuring Rewrite Rules”, and Chapter 12, “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.

```
! 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

</opt/SUNWmsgsr/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 10-1 shows how some example addresses would be routed by the preceding configuration.

Table 10-1 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 181, "Channel Definitions" on page 185, and Chapter 11, "Configuring Rewrite Rules"" for more information on the MTA configuration file.

NOTE Whenever changes are made to the `imta.cnf` file, the MTA configuration must be recompiled. See "Compiling the MTA Configuration" on page 222.

Mappings 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 `mappings` 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.

Mapping tables are kept in the MTA `mappings` file. This is the file specified with the `IMTA_MAPPING_FILE` option in the MTA `tailor` file; by default, this is *msg_svr_base*`/config/mappings`. The contents of the `mappings` file is incorporated into the compiled configuration as part of the reloadable section (see“Compiling the MTA Configuration” on page 222). The `mappings` file should be world readable. Failure to allow world-read access leads to erratic behavior. Whenever changes are made to the `mappings` file, the MTA configuration must be recompiled. See “Compiling the MTA Configuration” on page 222

Table 10-2 lists the mapping tables described in this book.

Table 10-2 Messaging Server Mapping Tables

Mapping Table	Page	Description
`CHARSET-CONVERSION`	409	Used to specify what sorts of channel-to-channel character set conversions and message reformatting should be done.
`COMMENT_STRINGS`	362	Used to modify address header comments (strings enclosed in parentheses).
`CONVERSIONS`	393	Used to select message traffic for the conversion channel.
“domain lookup″	221	Used for finding the base of the tree in which to search for aliases in direct LDAP mode.

Table 10-2 Messaging Server Mapping Tables

Mapping Table	Page	Description
FORWARD	256	Used to perform forwarding similar to that performed using the alias file or alias database.
FROM_ACCESS	482	Used to filter mail based on envelope From addresses. Use this table if the To address is irrelevant.
INTERNAL_IP	493	Used to recognize systems and subnets that are internal.
MAIL_ACCESS	482	Used to block incoming connections based on combined information found in SEND_ACCESS and PORT_ACCESS table.
NOTIFICATION_LANGUAGE	259	Used to customize or localize notification messages.
ORIG_MAIL_ACCESS	482	Used to block incoming connections based on combined information found in ORIG_SEND_ACCESS and PORT_ACCESS tables
ORIG_SEND_ACCESS	482	Used to block incoming connections based on envelope From address, envelope To address, source and destination channels.
PERSONAL_NAMES	363	Used to modify personal names (strings preceding angle-bracket-delimited addresses).
PORT_ACCESS	482	Used to block incoming connections based on IP number.
REVERSE	252	Used to convert addresses from an internal form to a public, advertised form.
SEND_ACCESS	482	Used to block incoming connections based on envelope From address, envelope To address, source and destination channels.
*SMS_Channel*_TEXT	773	Used for site-defined text conversions.
X-ATT-NAMES	400	Used to retrieve a parameter value from a mapping table.
X-REWRITE-SMS-ADDRESS	772	Used for local SMS address validity checks.

File Format in the Mappings File

The mappings 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 `mappings` file.

Including Other Files in the Mappings File

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

```
<file-spec
```

This effectively substitutes the contents of the file file-spec into the mappings 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 mappings file. Includes can be nested up to three levels deep. Include files are loaded at the same time the mappings 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 mappings 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 10-3 Mapping Pattern Wildcards

Wildcard	Description
%	Match exactly one character.
*	Match zero or more characters, with maximal or “greedy” left-to-right matching

Table 10-3 Mapping Pattern Wildcards *(Continued)*

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, _, $.
$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.

Table 10-3 Mapping Pattern Wildcards *(Continued)*

${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 10-4 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 10-4 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.

Table 10-4 Mapping Template Substitutions and Metacharacters *(Continued)*

Substitution sequence	Substitutes
$|...|	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.
$^	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 mappings 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 Lookup Table or Database Substitutions (${...})

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

If you are using the general lookup table you need to set the low order bit of the MTA option use_text_databases. That is, set it to an odd number. Changes to the general.txt need to be compiled into the MTA configuration using the imsimta cnbuild to compile and imsimta reload to reload the reloadable data.

If you are using a general database, 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 table substitution, the processing control metacharacter should be placed to the left of the general table substitution in the mapping table template; otherwise the “failure” of a general table 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*. 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, Messaging Server provides several other configuration files to help you configure MTA services. These files are summarized in Table 10-5. Note that whenever an MTA configuration file such as `imta.cnf`, `mappings`, `aliases`, or `option.dat` is modified, you must recompile the configuration (see the `imsimta refresh` command in the *Sun Java System Messaging Server Administration Reference*).

Table 10-5 MTA Configuration Files

File	Description
Alias File (mandatory)	Implements aliases not present in the directory. *msg_svr_base*`config/aliases`
TCP/IP (SMTP) Channel Option Files (also called SMTP Option Files)	Sets channel-specific options. *msg_svr_base*`config/`*channel*`_option`
Conversion File	Used by the conversion channel to control message body part conversions. *msg_svr_base*`config/conversions`
Dispatcher Configuration File (mandatory)	Configuration file for the Dispatcher. *msg_svr_base*`/config/dispatcher.cnf`
Job Controller File (mandatory)	Configuration file used by the `Job Controller`. `/`*msg_svr_base*`/config/job_controller.cnf`
MTA Configuration File (mandatory)	Used for address rewriting and routing as well as channel definition. `/`*msg_svr_base*`/config/imta.cnf`
Mappings File (mandatory)	Repository of mapping tables. `/`*msg_svr_base*`/config/mappings`
Option File	File of global MTA options. `/`*msg_svr_base*`/config/option.dat`
Tailor File (mandatory)	File to specify locations and some tuning parameters. `/`*msg_svr_base*`/config/imta_tailor`
General Lookup Table (optional)	General lookup facility is equivalent to the general database. Part of reloadable compiled configuration. File to specify locations and some tuning parameters. `/`*msg_svr_base*`/config/general.txt`
Forward Lookup Table (optional)	Lookup facility for To: addresses. Equivalent to forward database. Part of reloadable compiled configuration. `/`*msg_svr_base*`/config/forward.txt`

Table 10-5 MTA Configuration Files

File	Description
Reverse Lookup Table (optional)	Reverse lookup facility for From: addresses. Equivalent to reverse database. Part of reloadable compiled configuration */msg_svr_base*/config/reverse.txt

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 249.

After making changes to the aliases file, you must restart the MTA or issue the command imsimta reload.

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, *msg_svr_base*/config/imta/tcp_local_option. For more information refer to "Configuring SMTP Channel Options" on page 324. 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 390.

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 180.)

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=msg_svr_base/lib/tcp_smtp_server
LOGFILE=msg_svr_base/log/tcp_smtp_server.log
```

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

Mappings File

The `mappings` file 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 `mappings` 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 `mappings` file and the `test -mapping` command, see the "Mappings File" on page 225 and the *Messaging Server Reference Manual*.

After making changes to the `mappings` file, you must restart the MTA or issue the command `imsimta reload`.

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.

In `option.dat`, lines starting with `#`, `!` or `;` are treated as comment lines, even if the preceding line has a trailing `\` meaning that it is being continued. This means that long options that may contain these characters, in particular delivery options, care has to be taken.

For delivery options, where a natural layout would lead to continuation lines starting with a # or !, there is a safe and neat work around.

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 *msg_svr_base*`/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 187, “Processing Pools for Channel Execution Jobs” on page 348, and “Service Job Limits” on page 348.)

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.

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 10-1 shows a sample Job Controller configuration file. Table 10-6 shows the available options.

Code Example 10-1 Sample Job Controller Configuration File in UNIX

```
!MTA Job Controller configuration file
!
!Global defaults
tcp_port=27442          (1)
secret=never mind
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=msg_svr_base/lib/l_master
!
[CHANNEL=ims-ms]
master_command=msg_svr_base/lib/ims_master
!
[CHANNEL=tcp_*]          (7)
anon_host=0
master_command=msg_svr_base/lib/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 348. For more information about the syntax of the Job Controller file, see the *Messaging Server Reference Manual*.

Table 10-6 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.
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, *msg_svr_base*/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.

Table 10-6 Job Controller Configuration File Options *(Continued)*

Option	Description
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.
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. For a discussion on how alias resolution is handled see "The $V Metacharacter" on page 196.

The Alias Database

Use of the Alias Database is discouraged. Use the `aliases` file instead, especially since it can be dynamically reloaded using the `imsimta reload` command.

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.

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. Changes can be activated by compiling the configuration with `imimta cnbuild` and issuing the `imsimta reload` command (or restarting the MTA). 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 11, “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

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. 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 17, “Mail Filtering and Access Control”,” and Chapter 19, “Configuring Security and Access Control”.”

Log Files

All MTA specific log files are kept in the log directory, (*msg_svr_base*/log). 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 20, "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 (also called the *reverse 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 11, "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 generally located in the MTA database directory. The database is the files whose names are specified with the IMTA_REVERSE_DATABASE option in the *msg_svr_base*/config/imta_tailor file, which by default are the files *msg_svr_base*/data/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 mappings 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 mappings 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 10-7.

Table 10-7 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.

Table 10-7 REVERSE mapping table flags *(Continued)*

Flags	Description
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 *Sun Java System Messaging Server Administration Reference* 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 *Sun Java System Messaging Server Administration Reference* 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 *Sun Java System Messaging Server Administration Reference* 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.)

The Forward Lookup Table and FORWARD Address Mapping

Address reversals are not applied to envelope To: addresses. The reasons for this omission are fairly obvious—envelope To: 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 canonicalization functions of address reversal are entirely inappropriate for envelope To: addresses.

In any case, plenty of machinery is available in the MTA to perform substitutions on envelope To: addresses. The alias file, alias database and general lookup table, provide precisely this functionality.

The MTA also provides the forward lookup table and `FORWARD` mapping, used for special sorts of forwarding purposes, such as pattern-based forwarding, source-specific forwarding, or auto-registration of addresses. Note that the forward lookup table and `FORWARD` mapping are intended for use primarily for certain special sorts of address forwarding; most sorts of address forwarding, however, are better performed using one of the MTA's other forwarding mechanisms.

The various substitution mechanisms for envelope To: addresses provide functionality equivalent to the reversal lookup table, but none yet discussed provide functionality equivalent to the reverse mapping. And circumstances do arise where mapping functionality for envelope To: addresses is useful and desirable.

The FORWARD Mapping Table

The `FORWARD` mapping table provides this functionality of pattern based forwarding, and also provides a mechanism for source specific forwarding. 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 will replace the envelope To: address if the entry specifies a `$Y`; a `$N` will discard the result of the mapping. See Table 10-8 for a list of additional flags.

Table 10-8 FORWARD mapping table flags Flags Description

Flag	Description
$Y	Use output as new address

Table 10-8 FORWARD mapping table flags Flags Description

Flag	Description
$N	Address remains unchanged
$D	Run output through the rewriting process again
$G	Run output through the forward lookup table, if forward lookup table use has been enabled
$H	Disable further forward lookup table or FORWARD mapping lookups
$I	Hold the message as a .HELD file

The FORWARD mapping, if present, is consulted before any forward lookup table lookup. If a FORWARD mapping matches and has the flag $G, then the result of the FORWARD mapping will be checked against the forward lookup table, if forward lookup table use has been enabled via the appropriate setting of USE_FORWARD_DATABASE. (Note that if channel specific forward lookup table use has been specified, then the source address and source channel will be prefixed to the result of the FORWARD mapping before looking up in the forward lookup table.) If a matching FORWARD mapping entry specifies $D, then the result of the FORWARD mapping (and optional forward table lookup) will be run through the MTA address rewriting process again. If a matching FORWARD mapping entry specifies $H, then no further FORWARD mapping or database lookups will be performed during that subsequent address rewriting (that resulting from the use of $D).

The example below illustrates the use of a complex REVERSE and FORWARD mapping. Suppose that a system or pseudo domain named am.sigurd.innosoft.com associated with the mr_local channel produces RFC 822 addresses of the general form:

```
"lastname, firstname"@am.sigurd.example.com
```

or

```
"lastname,firstname"@am.sigurd.example.com
```

Although these addresses are perfectly legal they often confuse other mailers which do not fully comply with RFC 822 syntax rules—mailers which do not handle quoted addresses properly, for instance. Consequently, an address format which does not require quoting tends to operate with more mailers. One such format is

```
firstname.lastname@am.sigurd.example.com
```

Example of a complex FORWARD and REVERSE mapping:

```
REVERSE

 *|mr_local|"*,$ *"@am.sigurd.innosoft.com $Y"$1,$ $2"@am.sigurd.innosoft.com
 *|mr_local|"*,*"@am.sigurd.innosoft.com   $Y"$1,$ $2"@am.sigurd.innosoft.com
 *|*|"*,$ *"@am.sigurd.innosoft.com        $Y$3.$2@am.sigurd.innosoft.com
 *|*|"*,*"@am.sigurd.innosoft.com          $Y$3.$2@am.sigurd.innosoft.com
 *|mr_local|*.*@am.sigurd.innosoft.com     $Y"$2,$ $1"@am.sigurd.innosoft.com
 *|*|*.*@am.sigurd.innosoft.com            $Y$2.$3@am.sigurd.innosoft.com


FORWARD

 "*,$ *"@am.sigurd.innosoft.com           $Y"$0,$ $1"@am.sigurd.innosoft.com
 "*,*"@am.sigurd.innosoft.com             $Y"$0,$ $1"@am.sigurd.innosoft.com
 *.*@am.sigurd.innosoft.com               $Y"$1,$ $0"@am.sigurd.innosoft.com
```

So the goals of the sample mapping tables in the above example are threefold. (1) Allow any of these three address formats above to be used. (2) Present only addresses in the original format to the mr_local channel, converting formats as necessary. (3) Present only addresses in the new unquoted format to all other channels, converting formats as necessary. (The REVERSE mapping shown assumes that bit 3 in the MTA option USE_REVERSE_DATABASE is set.

The Forward Lookup Table

In cases where address forwardings need to be auto-registered or source specific, the forward lookup table is available. Note that use of the Forward lookup table for simple forwarding of messages is generally not appropriate; the aliases file or alias lookup table is a more efficient way to perform such forwarding. By default, the forward lookup table is not used at all; its use must be explicitly enabled via the USE_FORWARD_DATABASE option. Forward table lookups are performed after address rewriting and after alias expansion is performed, and after any FORWARD mapping is checked. If a forward table lookup succeeds, the resulting substituted address is then run through the MTA address rewriting process all over again.

There are two mechanisms available for the forward lookup table, an in-memory hash table or conventional database. Unless the size of the table is prohibitively large then hash table is recommended. (1,000 is not prohibitively large, 100,000 is). The hash table is enabled by setting bit 3 (value 34) in the use_text_database option

as well as setting use_forward_database. The hash table is read from *msg_svr_base*/configure/forward.txt, compiled into the reloadable part of the configuration, and can be forced to be reloaded into the active MTA processes by the imsimta reload command.

The forward database is an MTA crdb database, created using the crdb utility from a source text file. The format of the source text file by default is expected to be:

```
user1@domain1    changedmailbox1@changeddomain1
user2@domain2    changedmailbox@changeddomain2
```

But if source specific use of the forward database has been enabled by setting bit 3 of the USE_FORWARD_DATABASE option, then the source text file format expected is:

```
source-channel|source-address|original-address  changed-address
```

For instance, an entry such as

```
tcp_limited|bob@blue.com|helen@red.com  “helen of  troy”@siroe.com
```

will map the To: address helen@red.com to “helen of troy”@siroe.com if and only if the message is coming from bob@blue.com and the enqueuing channel is tcp_limited.

Controlling Delivery Status Notification Messages

Delivery status notifications or *status notifications* 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 status notifications for messages originating from specific channels.

By default, status notifications are stored in the *msg_svr_base*/config/locale/C directory (specified by the IMTA_LANG setting in the *msg_svr_base*/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 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 Status Notifications

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.

For double-byte languages such as Japanese, be sure to construct your text in Japanese, then view the text as if it was ASCII to check for % characters. If there are accidental % characters, then replace them with %%.

The format and structure of status notification message 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 status 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 10-9.

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 10-9 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.
%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 status 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 Delivery Status Notification Messages

Delivery Status 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 status 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 Status Notifications" on page 260.)
2. Set up a `NOTIFICATION_LANGUAGE` mapping table.

The `NOTIFICATION_LANGUAGE` mapping table (located in *msg_svr_base*`/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 *msg_svr_base*`/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.

Code Example 10-2 Notification Language Mapping Table—Example

```
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 Status Notification Message Features

The essential procedures for setting up status 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 Status 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 status 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 you set `RETURN_UNITS=1`, then you will need to schedule the return job to run hourly as well to get hourly notices. When the return job runs every hour it will also roll over the `mail.log*` files every hour. To prevent the hourly rollover of the `mail.log` file, set the `IMTA_RETURN_SPLIT_PERIOD` tailor file option in the `imta.tailor` file to 24. Return job scheduling is controlled by the local.schedule.`return_job` `configutil` parameter.

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 Status 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 status 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 status 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 Status Notification Messages to the Postmaster

By default, a copy of failure and warning status 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 10-10.

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 10-10.)

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 10-10.)

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 10-10.)

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 status 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 10-10 Keywords for Sending Notification Messages to the Postmaster and Sender

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.
nosendpost	Disables sending a copy of all failed messages to the postmaster.

Table 10-10 Keywords for Sending Notification Messages to the Postmaster and Sender

Keyword	Description
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.
Returned Message Content	**Specifies Addresses on Notifications**

Table 10-10 Keywords for Sending Notification Messages to the Postmaster and Sender

Keyword	Description
`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.

Controlling Message Disposition Notifications

Message Disposition Notifications (MDN) are email reports sent by the MTA to a sender and / or postmaster reporting on a message's delivery disposition. For example, if a message is rejected by a Sieve filter, an MDN will be sent to the sender. MDNs are also known as read receipts, acknowledgements, receipt notifications, or delivery receipts. The Sieve scripting language is typically used for messaging filtering and vacation messages.

To Customize and Localize Message Disposition Notification Messages

The instructions for modifying and localizing MDNs parallel those described in customizing and localizing delivery status notification messages with some minor differences as described here. (See "To Customize and Localize Delivery Status Notification Messages" on page 262.)

The mapping (called the `DISPOSITION_LANGUAGE` mapping) parallels the `notification_language` mapping table (Code Example 10-2 on page 264) used to internationalize status notifications.

However, probes for MDNs to this mapping take the following form:

```
type|modifiers|source-channel|header-language|return|recipient
```

Where:

`type` is disposition type, which can be one of the following: `displayed`, `dispatched`, `processed`, `deleted`, `denied`, or `failed`.

modifiers is a comma-separate list of disposition modifiers. The current list is: error, warning, superseded, and expired.

source-channel is the source channel producing the MDN.

header-language is the language specified in one of the following: accept-language, preferred-language, or x-accept-language. (MTA uses the first of these options that is present.)

return is the address to which the notification is being returned.

recipient is the address that the disposition is about.

The result of the disposition mapping consists of two or three pieces of information separated by vertical bars (|). The first piece of information is the directory where the template files for the disposition notification can be found. The second piece of information is the character set into which the standalone disposition text should be forced. (This information is required because some dispositions—notably the dispositions produced by autoreply echo or the use of the :mime parameter to the vacation Sieve action—do not employ template files and consequently cannot inherit the character set from those files.) Finally, the third piece of information is an override subject line for the notification. This information is only used if the $T flag is also set by the mapping.

The following additional template files are used to construct MDNs:

```
disposition_deleted.txt disposition_failed.txt
disposition_denied.txt disposition_prefix.txt
disposition_dispatched.txt disposition_processed.txt
disposition_displayed.txt disposition_suffix.txt
disposition_option.opt
```

The use of these template files parallels that of the various return_*.txt files for status notification messages.

Chapter 11

Configuring Rewrite Rules

This chapter describes how to configure rewrite rules in the `imta.cnf` file. If you have not already read Chapter 10, "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 restart` 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. The example below shows the rewrite rule section of a partial configuration file.

```
! 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 Table 11-3 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 276.

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*]

where

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 Table 11-4, the $U substitution sequence is used. For more information, see "Template Substitutions and Rewrite Rule Control Sequences" on page 288.

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 288.

For more information about template syntax, see "Rewrite Rule Templates" on page 279.

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` (*msg_svr_base*`/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 11-1, and discussed in the following subsections.

Table 11-1 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.

Table 11-1 Summary of Special Patterns for Rewrite Rules

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

In addition to these special patterns, Messaging Server also has the concept of *tags*, which may appear in rewrite rule patterns. These tags are used in situations where an address may be rewritten several times and, based upon previous rewrites, distinctions must be made in subsequent rewrites by controlling which rewrite rules match the address. For more information, see "Tagged Rewrite Rule Sets" on page 279.

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 279.) 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 11-2 summarizes the template formats.

Table 11-2 Summary of Template Formats for Rewrite Rules

Template	Page	Usage
A%B	280	A becomes the new user/mailbox name, B becomes the new host/domain specification, rewrite again.
A@B	280	Treated as A%B@B.

Table 11-2 Summary of Template Formats for Rewrite Rules

Template	Page	Usage
A%B@C	280	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	280	Treated as A@B@C@C.
A@B@C@D	280	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 11-3.

Table 11-3 Extracted Addresses and Host Names

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

Table 11-3 Extracted Addresses and Host Names *(Continued)*

Address	First Host Domain Specification	Comments
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. This section consists of the following subsections:

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

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

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

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

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

Table 11-4 summarizes the template substitutions and control sequences.

Table 11-4 Summary of Rewrite Rule Template Substitutions and Control Sequences

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

Table 11-4 Summary of Rewrite Rule Template Substitutions and Control Sequences *(Continued)*

Substitution Sequence	Substitutes
$[...]	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
$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

Table 11-4 Summary of Rewrite Rule Template Substitutions and Control Sequences *(Continued)*

Substitution Sequence	Substitutes
$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 11-5 lists the LDAP URL Substitution Sequences.

Table 11-5 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 imsimta 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 298 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 280), 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 11-6.

Table 11-6 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.
$*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 non-repeating 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 *msg_svr_base*`/data/db/domaindb.db`.

NOTE DO NOT EDIT THIS FILE BY HAND.

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 the following 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 11-7 shows some sample addresses and how they would be rewritten and routed according to the rewrite rules.

Table 11-7 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

Table 11-7 Sample Addresses and Rewrites

Initial address	Rewritten as	Routed to
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 12

Configuring Channel Definitions

This chapter describes how to use channel keyword definitions in the MTA configuration file `imta.cnf`. Please read Chapter 10, “About MTA Services and Configuration”,” as well as “Channel Definitions” on page 185 and “The MTA Configuration File” on page 222 before reading this chapter. This chapter contains the following sections:

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 restart` 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 12-1 Channel Keywords Alphabetized

Keyword	Page	Keyword	Page	Keyword	Page	Keyword	Page
733	354	822	354	addreturnpath	361	addrsperfile	378
aliaslocal	363	aliaspostmaster	268	allowetrn	328	allowswitchchannel	339
alternatechannel	376	alternateblocklimit	376	alternatelinelimit	376	alternaterecipientlimit	376
authrewrite	341	backoff	346	bangoverpercent	356	bangstyle	354
bidirectional	346	blocketrn	328	blocklimit	375	cacheeverything	335
cachefailures	335	cachesuccesses	335	channelfilter	382	charset7	330
charset8	330	charsetesc	330	checkehlo	327	commentinc	362
commentmap	362	commentomit	362	commentstrip	362	commenttotal	362
connectalias	357	connectcanonical	357	copysendpost	267	copywarnpost	267
daemon	340	datefour	368	datetwo	368	dayofweek	369
defaulthost	358	defaultmx	338	defaultnameservers	338	deferred	346
defragment	372	dequeue_removeroute	365	destinationfilter	382	disableetrn	328
domainetrn	328	domainvrfy	329	dropblank	360	ehlo	327
eightbit	330	eightnegotiate	330	eightstrict	330	errsendpost	267
errwarnpost	267	expandchannel	352	expandlimit	352	expnallow	330
`expndisable`	330	`expndefault`	330			exproute	356
fileinto	382	filesperjob	348	filter	382	forwardcheckdelete	336

Table 12-1 Channel Keywords Alphabetized

Keyword	Page	Keyword	Page	Keyword	Page	Keyword	Page
forwardchecknone	336	forwardchecktag	336	header_733	354	header_822	354
header_uucp	354	headerlabelalign	370	headerlinelength	370	headerread	367
headertrim	367	holdexquota	378	holdlimit	352	identnone	337
identnonelimited	337	identnonenumeric	337	identnonesymbolic	337	identtcp	337
identtcplimited	337	identtcpsymbolic	337	ignoreencoding	372	immnonurgent	
improute	356	includefinal	266	indenttcpnumeric	337	inner	366
innertrim	367	interfaceaddress	335	interpretencoding	372	language	371
lastresort	339	linelength	374	linelimit	375	localvrfy	329
logging	380	loopcheck	381	mailfromdnsverify	330	master	346
master_debug	381	maxblocks	373	maxheaderaddrs	369	maxheaderchars	369
maxjobs	348	maxlines	373	maxprocchars	370	maysaslserver	341
maytls	342	maytlsclient	342	maytlsserver	342	missingrecipientpolicy	359
msexchange	342	multiple	378	mustsaslserver	341	musttls	342
musttlsclient	342	musttlsserver	342	mx	338	nameservers	338
noaddreturnpath	361	nobangoverpercent	356	noblocklimit	375	nocache	335
nochannelfilter	382	nodayofweek	369	nodefaulthost	358	nodeferred	346
nodefragment	372	nodestinationfilter	382	nodropblank	360	noehlo	327
noexproute	356	noexquota	378	nofileinto	382	nofilter	382
noheaderread	367	noheadertrim	367	noimproute	356	noinner	366
noinnertrim	367	nolinelimit	375	nologging	380	noloopcheck	381
nomailfromdnsverify	330	nomaster_debug	381	nomsexchange	341	nomx	338
nonrandomemx	338	nonurgentbackoff	346	nonurgentblocklimit	351	nonurgentnotices	265
noreceivedfor	361	noreceivedfrom	361	noremotehost	358	norestricted	360
noreturnaddress	268	noreturnpersonal	268	noreverse	360	normalbackoff	346
normalblocklimit	351	normalnotices	265	norules	365	nosasl	341
nosaslserver	341	nosaslswitchchannel	341	nosendetrn	328	nosendpost	267
noservice	353	noslave_debug	381	nosmtp	327	nosourcefilter	382
noswitchchannel	339	notices	265	notls	342	notlsclient	342
notlsserver	342	novrfy	329	nowarnpost	267	nox_env_to	368

Table 12-1 Channel Keywords Alphabetized

Keyword	Page	Keyword	Page	Keyword	Page	Keyword	Page
percentonly	356	percents	354	personalinc	363	personalmap	363
personalomit	363	personalstrip	363	pool	348	port	335
postheadbody	268	postheadonly	268	randommx	338	receivedfor	361
receivedfrom	361	remotehost	358	restricted	360	returnaddress	268
returnenvelope	267	returnpersonal	268	reverse	360	routelocal	357
rules	365	rules	365	saslswitchchannel	341	sendetrn	328
sendpost	267	sensitivitycompanyconfidential	371	sensitivitynormal	371	sensitivitypersonal	371
sensitivityprivate	371	service	353	sevenbit	330	silentetrn	328
single	378	single_sys	340	slave	346	slave_debug	381
smtp	327	smtp_cr	327	smtp_crlf	327	smtp_crorlf	327
smtp_lf	327	sourceblocklimit	375	sourcecommentinc	362	sourcecommentmap	362
sourcecommentomit	362	sourcecommentstrip	362	sourcecommenttotal	362	sourcefilter	382
sourcepersonalinc	363	sourcepersonalmap	363	sourcepersonalomit	363	sourcepersonalstrip	363
sourceroute	354	streaming	332	subaddressexact	364	subaddressrelaxed	364
subaddresswild	364	subdirs	380	submit	382	suppressfinal	266
switchchannel	339	threaddepth	351	tlsswitchchannel	342	transactionlimit	350
unrestricted	360	urgentbackoff	346	urgentblocklimit	351	urgentnotices	265
useintermediate	266	user	382	uucp	354	viaaliasoptional	365
viaaliasrequired	365	vrfyallow	329	vrfydefault	329	vrfyhide	329
warnpost	267	x_env_to	368				

Channel Keywords Categorized by Function

The following table is a categorized list of keywords.

Table 12-2 Channel Keywords Categorized by Function

Keyword	Page	Definition
Address Handling		
733	354	Use % routing in the envelope; synonymous with `percents`.
822	354	Use source routes in the envelope; same as `sourceroute`.
addreturnpath	361	Add `Return-path:` header to messages enqueued to this channel.
aliaslocal	363	Look up rewritten addresses in the alias file and alias database.
authrewrite	341	Used on a source channel to have the MTA propagate authenticated originator information, if available, into the headers.
bangoverpercent	356	Group A!B%C as A!(B%C)
bangstyle	354	Use UUCP ! routing in the envelope; synonymous with `uucp`.
defaulthost	358	Specify a domain name to use to complete addresses
dequeue_removeroute	365	Removes source routes from envelope `To:` addresses.
exproute	356	Require explicit routing when addresses passed to remote systems.
holdlimit	352	Hold message when number of envelope recipient addresses exceeds this limit.
improute	356	Implicit routing for this channel's addresses
missingrecipientpolicy	359	Set policy for how to legalize (which header to add) messages that are lacking any recipient headers.
noaddreturnpath	361	Do not add `Return-path:` header when enqueuing message.
nobangoverpercent	356	Group A!B%C as (A!B)%C
nodefaulthost	358	Do not specify a domain name to use to complete addresses
noexproute	356	No explicit routing for this channel's addresses
noimproute	356	No implicit routing for this channel's addresses
noreceivedfrom	361	Construct `Received:` header lines without including the original envelope From: address.
noremotehost	358	Use local host's domain name as the default domain name to complete addresses
norestricted	360	Same as `unsrestricted`.
noreverse	360	Exempts addresses in messages from address reversal processing
norules	365	Do not enforce channel-specific rewrite rule checks for this channel.

Table 12-2 Channel Keywords Categorized by Function

Keyword	Page	Definition
percentonly	356	Ignores bang paths. Use % routing in the envelope.
percents	354	Use % routing in the envelope; synonymous with 733.
remotehost	358	Use remote host's name as the default domain name to complete addresses
restricted	360	The channel connects to mail systems that require encoding.
reverse	360	Checked addresses against address reversal database or REVERSE mapping
routelocal	357	Causes the MTA, when rewriting an address to the channel, to attempt to "short circuit" any explicit routing in the address.
rules	365	Enforce channel-specific rewrite rule checks for this channel.
sourceroute	354	Synonymous with 822.
subaddressexact	364	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	364	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	364	After looking for an exact match including the entire subaddress, the MTA should next look for an entry of the form name+*.
unrestricted	360	Tells the MTA not to perform RFC 1137 encoding and decoding.
uucp	354	Use UUCP! routing in the envelope; synonymous with bangstyle.
viaaliasoptional	365	Final recipient addresses that match the channel are not required to be produced by an alias.
viaaliasrequired	365	Final recipient address that matches the channel must be produced by an alias.
Attachments and MIME Processing		
defragment	372	Partial messages queued to the channel are placed in the defragmentation channel queue instead.
ignoreencoding	372	Ignore `Encoding:` header on incoming messages.
interpretencoding	372	Interpret `Encoding:` header on incoming messages, if the need arises.
nodefragment	372	Disables defragmention.
Character Sets and Eight Bit Data		
charset7	330	Default character set to associate with 7-bit text messages
charset8	330	Default character set to associate with 8-bit text messages
charsetesc	330	Default character set to associate with 7-bit text containing the escape character
eightbit	330	Channel supports eight-bit characters.
eightnegotiate	330	Channel should negotiate use of eight-bit transmission if possible.

Table 12-2 Channel Keywords Categorized by Function

Keyword	Page	Definition
eightstrict	330	Reject messages with headers that contain unnegotiated eight-bit data.
sevenbit	330	Do not support 8-bit characters; 8-bit characters must be encoded.
File Creation in the MTA Queue Area		
addrsperfile	378	Limit on the maximum number of recipients that can be associated with a single message file in a channel queue
expandchannel	352	Specifies channel in which to perform deferred expansion due to application of expandlimit.
expandlimit	352	Processes an incoming message “off-line” when the number of addressees exceeds this limit.
multiple	378	No limit on the number of recipients in a message file, however the SMTP channel defaults to 99.
single	378	A separate copy of the message will be created for each destination address on the channel.
single_sys	378	Create a single message copy for each destination system used.
subdirs	380	Specifies the number of subdirectories across which to spread messages for the channel queues.
Headers		
authrewrite	341	Used on a source channel to have the MTA propagate authenticated originator information, if available, into the headers.
commentinc	362	Leave comments in message header lines intact.
commentmap	362	Runs comment strings in message header lines through the `COMMENT_STRINGS` mapping table.
commentomit	362	Remove comments from message header lines.
commentstrip	362	Remove problematic characters from comment fields in message header lines.
commenttotal	362	Strip comments (material in parentheses) from all header lines, except Received: header lines. Not recommended.
datefour	368	Expand all year fields to four digits.
datetwo	368	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	369	Retain day of the week information and adds this information to date and time headers if it is missing.
defaulthost	358	Specify a domain name to use to complete addresses
dropblank	360	Strip illegal blank headers from incoming messages.
header_733	354	Use % routing in the message header.

Table 12-2 Channel Keywords Categorized by Function

Keyword	Page	Definition
header_822	354	Use source routes in the message header.
headerlabelalign	370	Controls the alignment point for message headers enqueued on this channel; it takes an integer-valued argument.
headerlinelength	370	Controls the length of header lines enqueued on this channel.
headerread	367	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	367	Applies header trimming rules from an options file to the message headers after the original message headers are processed.
header_uucp	354	Use ! routing in the header
inner	366	Parse messages and rewrite inner headers.
innertrim	367	Apply header trimming rules from an options file to inner message headers (use with caution).
language	371	Specifies the default language in headers.
maxheaderaddrs	369	Controls how many addresses can appear on a single line.
maxheaderchars	369	Controls how many characters can appear on a single line.
missingrecipientpolicy	359	Set policy for how to legalize (which header to add) messages that are lacking any recipient headers.
nodayofweek	369	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	358	Do not specify a domain name to use to complete addresses
nodropblank	360	Do not strip illegal blank headers from incoming messages.
noheaderread	367	Do not apply header trimming rules from option file.
noheadertrim	367	Do not apply header trimming rules from options file.
noinner	366	Do not to rewrite inner message header lines.
noinnertrim	367	Do not apply header trimming to inner message headers.
noreceivedfor	361	Construct `Received:` header lines without including any envelope recipient information.
noreceivedfrom	361	Construct `Received:` header lines without including the original envelope From: address.
noremotehost	358	Use local host's domain name as the default domain name to complete addresses
noreverse	360	Exempts addresses in messages queued to the channel from address reversal processing

Table 12-2 Channel Keywords Categorized by Function

Keyword	Page	Definition
norules	365	Do not enforce channel-specific rewrite rule checks for this channel.
nox_env_to	368	Remove `X-Envelope-to` header lines.
personalinc	363	Leave personal name fields in message header lines intact.
personalmap	363	Run personal names through PERSONAL_NAMES mapping table.
personalomit	363	Remove personal name fields from message header lines.
personalstrip	363	Strip problem characters from personal name fields in header lines.
receivedfor	361	If a message is addressed to just one envelope recipient, to include that envelope `To:` address in the Received: header line it constructs.
receivedfrom	361	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	358	Use remote host's name as the default domain name to complete addresses
restricted	360	Channel connects to mail systems that require this encoding.
reverse	360	Check addresses against address reversal database or REVERSE mapping
rules	365	Enforce channel-specific rewrite rule checks for this channel.
sensitivitycompanyconfidential	371	`Companyconfidential` is the upper sensitivity limit of messages accepted.
sensitivitynormal	371	`Normal` is the upper sensitivity limit of messages accepted.
sensitivitypersonal	371	`Personal` is the upper sensitivity limit of messages accepted.
sensitivityprivate	371	`Private` is the upper sensitivity limit of messages accepted.
sourcecommentinc	362	Leave comments in incoming message header lines.
sourcecommentmap	362	Runs comment strings in header lines through source channels.
sourcecommentomit	362	Remove comments from incoming message header lines, for example, To:, From:, and Cc: headers.
sourcecommentstrip	362	Remove problematic characters from comment field in incoming header lines.
sourcecommenttotal	362	Strip comments (material in parentheses) in incoming messages.
sourcepersonalinc	363	Leave personal names in incoming message header lines intact.
sourcepersonalmap	363	Run personal names through source channels.
sourcepersonalomit	363	Remove personal name fields from incoming message header lines.
sourcepersonalstrip	363	Strip problematic characters from personal name fields in incoming message header lines.
unrestricted	360	Tells the MTA not to perform RFC 1137 encoding and decoding.

Table 12-2 Channel Keywords Categorized by Function

Keyword	Page	Definition
x_env_to	368	Enables generation of `X-Envelope-to` header lines.
Incoming channel matching and switching		
allowswitchchannel	339	Allows switching to this channel from a `switchchannel` channel
nosaslswitchchannel	341	No switching to this channel upon successful SASL authentication
noswitchchannel	339	No channel switching should be done to or from the channel.
switchchannel	339	Switches from the server channel to the channel associated with the originating host.
saslswitchchannel	341	Cause incoming connections to be switched to a specified channel upon a client's successful use of SASL.
tlsswitchchannel	342	Switches to another channel upon successful TLS negotiation.
Logging and debugging		
logging	380	Log message enqueues and dequeues into the log file and activates logging for a particular channel.
loopcheck	381	Places a string into the SMTP EHLO response banner in order for the MTA to check if it is communicating with itself.
master_debug	381	Create debugging output in the channel's master program output.
nologging	380	Do not log message enqueues and dequeues into the log file.
noloopcheck	381	No string into the SMTP EHLO response banner.
nomaster_debug	381	No debugging output in the channel's master program output.
noslave_debug	381	Do not generate slave debugging output.
slave_debug	381	Generate slave debug output.
Long Address Lists or Headers		
expandchannel	352	Specifies channel in which to perform deferred expansion due to application of expandlimit.
expandlimit	352	Processes an incoming message "off-line" when the number of addressees exceeds this limit.
holdlimit	352	Holds a message when the number of addresses exceeds this limit.
maxprocchars	370	Maximum length header that can be processed and rewritten.
Mailbox filters		
channelfilter	382	Location of channel filter file; same as `destinationfilter`.
destinationfilter	382	Location of channel filter file that applies to outgoing messages.
fileinto	382	Specify effect on address when a mailbox filter `fileinto` operation is applied.
filter	382	Specify the location of user filter files.

Table 12-2 Channel Keywords Categorized by Function

Keyword	Page	Definition
nochannelfilter	382	No channel filtering for outgoing messages. Also known as `nodestinationfilter`.
nodestinationfilter	382	Do not perform channel filtering for outgoing messages.
nofileinto	382	Mailbox filter fileinto operator has no effect.
nofilter	382	Do not perform user mailbox filtering.
nosourcefilter	382	Do not perform channel filtering for incoming messages.
sourcefilter	382	Specify the location of channel filter file for incoming messages.
Notification and Postmaster Messages (See page 259 for complete notification procedures)		
aliaspostmaster	268	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).
copysendpost	267	Sends a copy of the failure notice to the postmaster unless the originator address on the failing message is blank.
copywarnpost	267	Sends a copy of the warning message to the postmaster unless the originator address on the undelivered message is blank.
errsendpost	267	Sends a copy of the failure notice to the postmaster only when the notice cannot be returned to the originator.
errwarnpost	267	Sends a copy of the warning message to the postmaster when the notice cannot be returned to the originator.
includefinal	266	Include final form of recipient address in delivery notifications.
nonurgentnotices	265	Specifies the amount of time that may elapse before notices are sent and messages returned for messages of non-urgent priority.
noreturnaddress	268	Use `RETURN_ADDRESS` option value as postmaster address name.
noreturnpersonal	268	Use `RETURN_PERSONAL` option value as postmaster personal name.
normalnotices	265	Specifies the amount of time that may elapse before notices are sent and messages returned for messages of normal priority.
nosendpost	267	Disables sending a copy of all failed messages to the postmaster.
notices	265	Specifies the amount of time that may elapse before notices are sent and messages returned.
nowarnpost	267	Disables sending a copy of warning messages to the postmaster.
postheadbody	268	Returns both the headers and the contents of the message.
postheadonly	268	Returns only headers to the postmaster.
returnaddress	268	Specifies the return address for the local postmaster.
returnenvelope	267	Control use of blank envelope return addresses.

Table 12-2 Channel Keywords Categorized by Function

Keyword	Page	Definition
returnpersonal	268	Set the personal name for the local postmaster.
sendpost	267	Enables sending a copy of all failed messages to the postmaster.
suppressfinal	266	Suppress the final address form from notification messages, if an original address form is present, from notification messages.
urgentnotices	265	Specify the amount of time which may elapse before notices are sent and messages returned for messages of urgent priority.
useintermediate	266	Uses an intermediate form of the address produced after list expansion, but prior to user mailbox name generation.
warnpost	267	Enables sending a copy of warning messages to the postmaster.
Processing Control and Job Submission (See Table 12-7 on page 344 for greater functional granularity)		
backoff	346	Frequency of attempted redelivery of unsuccessfully delivered messages. Can be overridden by the keywords `normalbackoff`, `nonurgentbackoff`, `urgentbackoff`.
bidirectional	346	Channel served by a master and slave program.
deferred	346	Recognize and honor of the `Deferred-delivery:` header line.
expandchannel	352	Specifies channel in which to perform deferred expansion due to application of expandlimit.
expandlimit	352	Processes an incoming message “off-line” when the number of addressees exceeds this limit.
filesperjob	348	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	346	Channel served by a master program (`master`).
maxjobs	348	Maximum number of jobs that can be running concurrently for the channel.
nodeferred	346	Specifies that `Deferred-delivery:` header line not be honored.
nonurgentbackoff	346	The frequency for attempted redelivery of nonurgent messages.
nonurgentblocklimit	351	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	346	The frequency for attempted redelivery of normal messages.
normalblocklimit	351	Forces messages above this size to nonurgent priority.
noservice	353	Service conversions for messages coming into this channel must be enabled via `CHARSET-CONVERSION`.

Table 12-2 Channel Keywords Categorized by Function

Keyword	Page	Definition
pool	348	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	353	Unconditionally enables service conversions regardless of `CHARSET-CONVERSION` entry.
slave	346	Channel served by a master program (slave).
threaddepth	**351**	Number of messages triggering new thread with multithreaded SMTP client.
transactionlimit		Limits the number of messages allowed per connection.
urgentbackoff	346	The frequency for attempted redelivery of urgent messages.
urgentblocklimit	351	Forces messages above this size to normal priority.
user	382	Used on pipe channels to indicate under what user name to run.
Sensitivity limits		
sensitivitycompanyconfidential	371	Upper sensitivity limit of messages accepted.
sensitivitynormal	371	`Normal` is the upper sensitivity limit of messages accepted.
sensitivitypersonal	371	`Personal` is the upper sensitivity limit of messages accepted.
sensitivityprivate	371	`Private` is the upper sensitivity limit of messages accepted.
Size Limits on Messages, and User Quotas and Privileges		
alternatechannel	376	Alternate destination channel for alternateblocklimit, alternatelinelimit, and alternaterecipientlimit
alternateblocklimit	376	Specifies limit on the number of blocks in a message before it will be sent to alternativechannel.
alternatelinelimit	376	Specifies limit on the number of lines in a message before it will be sent to alternativechannel.
alternaterecipientlimit	376	Specifies limit on the number of recipients in a message before it will be sent to alternativechannel.
blocklimit	375	Maximum number of MTA blocks allowed per message.
holdexquota	378	Hold messages for users that are over quota.
holdlimit	352	Holds an incoming message when the number of addresses exceeds this limit.
linelength	374	Limits the maximum permissible message line length on a channel-by-channel basis.
linelimit	375	Maximum number of lines allowed per message.
maxblocks	373	Specifies the maximum number of blocks allowed in a message.
maxlines	373	Specifies the maximum number of lines allowed in a message.

Table 12-2 Channel Keywords Categorized by Function

Keyword	Page	Definition
noblocklimit	375	No limit for the number of MTA blocks allowed per message.
noexquota	378	Return to originator any messages to users who are over quota.
nolinelimit	375	No limit specified for the number of lines allowed per message.
nonurgentblocklimit	351	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	351	Forces messages above this size to nonurgent priority.
sourceblocklimit	375	Maximum number of MTA blocks allowed per incoming message.
urgentblocklimit	351	Forces messages above this size to normal priority.
SMTP Authentication, SASL and TLS (See 341 for greater functional granularity)		
authrewrite	341	Used on a source channel to have the MTA propagate authenticated originator information, if available, into the headers.
maysaslserver	341	Permit clients to attempt to use SASL authentication.
maytls	342	Causes the MTA to offer TLS to incoming connections and to attempt TLS upon outgoing connections.
maytlsclient	342	The MTA SMTP client will attempt TLS use when sending outgoing messages, if sending to an SMTP server that supports TLS.
maytlsserver	342	The MTA SMTP server will advertise support for the STARTTLS extension and will allow TLS use when receiving messages.
msexchange	342	Used on TCP/IP channels to tell the MTA that this is a channel that communicates with Microsoft Exchange gateways and clients.
mustsaslserver	341	SMTP server does not accept messages unless remote client successfully authenticates.
musttls	342	Insist upon TLS in both outgoing and incoming connections.
musttlsclient	342	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	342	The MTA SMTP server will advertise support for the STARTTLS extension and will insist upon TLS use when receiving incoming messages.
nomsexchange	341	Default.
nosasl	341	SASL authentication is not permitted or attempted.
nosaslserver	341	SASL authentication is not permitted.
notls	342	TLS will not be permitted or attempted.
notlsclient	342	TLS use will not be attempted by the MTA SMTP client on outgoing connections (the STARTTLS command will not be issued during outgoing connections).

Table 12-2 Channel Keywords Categorized by Function

Keyword	Page	Definition
notlsserver	342	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	341	Cause incoming connections to be switched to a specified channel upon a client's successful use of SASL.
tlsswitchchannel	342	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 12-4 on page 325 for greater functional granularity)		
allowetrn	328	Honors ETRN commands.
blocketrn	328	Blocks ETRN commands.
checkehlo	327	Checks the SMTP response banner to determine whether to use EHLO or HELO.
disableetrn	328	Disable support for the ETRN SMTP command.
domainetrn	328	Honors only those ETRN commands that specify a domain.
domainvrfy	329	Issues VRFY commands using a full address.
ehlo	327	Uses the SMTP EHLO command on initial connections.
eightbit	330	Channel supports eight-bit characters.
eightnegotiate	330	Channel should negotiate use of eight-bit transmission if possible.
eightstrict	330	Reject messages with headers that contain unnegotiated eight-bit data.
expnallow	330	Allows `EXPN` even if it has been disabled at the SMTP server level with the `DISABLE_EXPAND` SMTP channel option.
expndisable	330	Disables `EXPN` unconditionally.
expndefault	330	Allows `EXPN` if the SMTP server is set to allow it.
localvrfy	329	Issues VRFY commands using a local address.
mailfromdnsverify	330	Verifies domain used on `MAIL FROM:` command exists in the DNS.
noehlo	327	Does not use the EHLO command.
nomailfromdnsverify	330	Does not verify that the domain used on the `MAIL FROM:` command exists in the DNS.
nosendetrn	328	Does not send ETRN commands.
nosmtp	327	Does not support the SMTP protocol. This is the default.
novrfy	329	Does not issue VRFY commands.
sendetrn	328	Sends ETRN commands.
sevenbit	330	Do not support 8-bit characters; 8-bit characters must be encoded.

Table 12-2 Channel Keywords Categorized by Function

Keyword	Page	Definition
silentetrn	328	Honors ETRN commands without echoing channel information.
smtp	327	Supports the SMTP protocol. The keyword `smtp` is mandatory for all SMTP channels. (This keyword is equivalent to `smtp_crorlf`.)
smtp_cr	327	Accepts lines terminated with a carriage return (CR) without a following line feed (LF).
smtp_crlf	327	Lines must be terminated with a carriage return (CR) line feed (LF) sequence.
smtp_crorlf	327	Lines may be terminated with any of a carriage return (CR), or a line feed (LF) sequence, or a full CRLF.
smtp_lf	327	Accepts lines terminated with linefeed (LF) without preceding CR.
streaming	**332**	Controls the degree of protocol streaming used in the protocol associated with a channel.
vrfyallow	329	Provides informative responses to VRFY commands.
vrfydefault	329	Provides default responses to VRFY command, according to channel's HIDE_VERIFY option setting.
vrfyhide	329	Provides obfuscatory responses to SMTP VRFY command.
TCP/IP Connection and DNS Lookup Support (See Table 12-5 on page 333 for greater functional granularity)		
cacheeverything	335	Caches all connection information.
cachefailures	335	Caches only connection failure information.
cachesuccesses	335	Caches only connection success information.
connectalias	357	Deliver to whatever host is listed in the recipient address.
connectcanonical	357	Connect to the host alias for the system to which the MTA would be connected.
daemon	340	Connects to a specific host system regardless of the envelope address.
defaultmx	338	Channel determines whether to do MX lookups from network.
defaultnameservers	338	Consults TCP/IP stack's choice of nameservers.
forwardcheckdelete	336	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	336	Does not perform a forward lookup after a DNS reverse lookup.
forwardchecktag	336	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	337	No perform IDENT lookups; performs IP to hostname translation; includes both hostname and IP address in `Received:` header.

Table 12-2 Channel Keywords Categorized by Function

Keyword	Page	Definition
identnonelimited	337	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	337	Does not perform IDENT lookups or IP to hostname translation.
identnonesymbolic	337	Does not perform IDENT lookups; does perform IP to hostname translation; includes only the hostname in `Received:` header.
identtcp	337	Performs IDENT lookups on incoming SMTP connections and IP to hostname translation; include both hostname and IP address in `Received:` header
identtcplimited	337	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	337	Performs IDENT lookups on incoming SMTP connections, but does not perform IP to hostname translation.
identtcpsymbolic	337	Performs IDENT lookups on incoming SMTP connections and IP to hostname translation; includes only hostname in `Received:` header.
interfaceaddress	335	Binds to the specified TCP/IP interface address.
lastresort	339	Specifies a last resort host.
mailfromdnsverify	330	Verifies that the domain used on the `MAIL FROM:` command exists in the DNS.
mx	338	TCP/IP network and software supports MX records lookup.
nameservers	338	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	335	Does not cache any connection information.
nomailfromdnsverify	330	Does not verify that the domain used on the `MAIL FROM:` command exists in the DNS.
nomx	338	TCP/IP network does not support MX lookups.
nonrandomemx	338	Does MX lookups; does not randomize returned entries with equal precedence.
port	335	Specifies the default port number for SMTP connections. The standard port is 25.
randommx	338	Does MX lookups; randomizes returned entries with equal precedence.
single	340	Specifies that a separate copy of the message should be created for each destination address on the channel.
single_sys	340	Creates single copy of message for each destination system used.
threaddepth	**351**	Number of messages triggering new thread with multithreaded SMTP client.
Miscellaneous		

Table 12-2 Channel Keywords Categorized by Function

Keyword	Page	Definition
submit	382	Used to mark a channel as a submit-only channel.
user	382	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 12-3 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 (*msg_svr_base*/`config`) and named *x*`_option`, where *x* is the name of the channel. Refer to the *Sun Java System Messaging Server Administration Reference* for details.

This section is divided into the following subsections:

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, */msg_svr_base*/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, EXPN 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:

- “DNS Domain Verification” on page 330
- “Character Set Labeling and Eight-Bit Data” on page 330
- “Protocol Streaming” on page 332

Table 12-4 summarizes the keywords described in this section.

Table 12-4 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.
`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.

Table 12-4 SMTP Command and Protocol Keywords

Channel Keyword(s)	Description
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.
EXPN Keywords	**Specifies how the channel handles EXPN keywords**
expnallow	Allows EXPN even if it has been disabled at the SMTP server level with the DISABLE_EXPAND SMTP channel option.
expndisable	Disables EXPN unconditionally.
expndefault	Allows EXPN if the SMTP server is set to allow it. (Default)
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 with headers that contain illegal 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**
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 smtp_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.

EXPN Support

Keywords: expnallow, expndisable, expndefault

expnallow allows EXPN even if it has been disabled at the SMTP server level with the DISABLE_EXPAND SMTP channel option. expndisable disables EXPN unconditionally. expndefault allows EXPN if the SMTP server is set to allow it (default). Expansion can be disabled on a per-list basis, but if it is disabled at the server level, the per-list settings are irrelevant.

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 with headers that contain illegal 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:

Table 12-5 lists the TCP/IP connection and DNS lookup keywords described in this section.

Table 12-5 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.
`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**

Table 12-5 TCP/IP Connection and DNS Lookup Keywords

Channel Keyword(s)	Description
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.
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.

Table 12-5 TCP / IP Connection and DNS Lookup Keywords

Channel Keyword(s)	Description
`tlsswitchchannel`	Switches to another channel upon successful TLS negotiation.
`saslswitchchannel`	Switches to another channel when SASL authentication is successful.
Target Host Choice and Storage of Message Copies	**Specifies a target host system and how message copies are stored.**
`daemon`	Connects to a specific host system regardless of the envelope address.
`single`	Specifies that a separate copy of the message should be created for each destination address on the channel.
`single_sys`	Creates a single copy of the message for each destination system used.

TCP/IP Port Number and Interface Address

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 341, and tlsswitchchannel on 342

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 mail hub 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 19, “Configuring Security and Access Control”.”

The maysaslserver, mustsaslserver, nosasl, nosaslserver, switchchannel, 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 12-6.

Table 12-6 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.

Note that for TLS to work, the following conditions must be in place:

- Protections/ownerships of the certificates have to be set so that the `mailsrv` account can access the files.
- The directory where the certificates are stored need to have protections/ownerships set such that the `mailsrv` account can access the files within that directory.

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:

- “Expansion of Multiple Addresses” on page 352
- “Enable Service Conversions” on page 353

For conceptual information on message processing and delivery, refer to “The Job Controller” on page 187 and “Job Controller File” on page 242.

Table 12-7 summarizes the keywords described in this section.

Table 12-7 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.
Channel Directionality	**Specifies type of by which program a channel is served**
`bidirectional`	Channel is served by a master and slave programs.
`master`	Channel is served by a master program (`master`).
`slave`	Channel is served by a slave program (`slave`).
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.

Table 12-7 Message Processing and Delivery Keywords

Keyword	Definition
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.
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.
Transaction Limits	**Specifies connection transaction limits**
`transactionlimit`	Limits the number of messages allowed per connection.
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 265.)

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 *Sun Java System Messaging Server Administration Reference*.

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 processed before lower-priority messages. (See “Message Priority Based on Size” on page 351.)

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 242, “The Job Controller” on page 187 and “Service Job Limits” on page 348.)

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 242, “Processing Pools for Channel Execution Jobs” on page 348 and “The Job Controller” on page 187.)

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 351.

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.

Setting Connection Transaction Limits

Keywords: `transactionlimit`

`transactionlimit` limits the number of messages allowed per connection. This can be used to thwart attackers in the following way:

An attacker can connect via SMTP and send many `RCPT TO` commands in an attempt to guess legitimate email addresses. Such an attack can be thwarted by limiting the number of invalid `RCPT TO`s allowed in a transaction. The attacker may respond by using multiple transactions, but with `transactionlimit` you can limit

the number of transaction allowed in an SMTP session. The attacker can use multiple sessions, but now his cost is getting prohibitive. Connection throttling can be used to limit the number of sessions in various ways making the cost really prohibitive in most cases.

This is not without cost our side, however. Some SMTP clients react badly to recipient limits, transaction limits, or both. Exceptions need to be made for these clients. But TCP channel options apply to the SMTP server unconditionally. The solution is to use channel keywords and `switchchannel` to route problematic agents to channels with larger limits.

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)" on page 339 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 12-8 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 252 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. Header option file format is described in the MTA chapter of the *Sun Java System Messaging Server Administration Reference*.

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 occasional 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, *instance_root*/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.

Defragmentation Channel Retention Time

Messages are retained in the defragment channel queue only for a limited time. When one half of the time before the first nondelivery notice is sent has elapsed, the various parts of a message will be sent on without being reassembled. This choice of time value eliminates the possibility of a nondelivery notification being sent about a message in the defragment channel queue.

The notices channel keyword controls the amount of time that can elapse before nondelivery notifications are sent, and hence also controls the amount of time messages are retained before being sent on in pieces. Set the notices keyword value to twice the amount of time you wish to retain messages for possible defragmentation. For example, a notices value of 4 would cause retention of message fragments for two days:

```
defragment notices 4
DEFRAGMENT-DAEMON
```

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.

Retargeting Messages Exceeding Limit on Size or Recipients

Keywords: alternatechannel, alternateblocklimit, alternatelinelimit, alternaterecipientlimit

The MTA provides the ability to retarget messages that exceed a specified limit on the number of recipients, message size, or message lines to an alternate destination channel. This is implemented as a set of the following channel keywords alternatechannel, alternateblocklimit, alternatelinelimit, and alternaterecipientlimit that can be placed on any destination channel. The alternatechannel keyword takes a single argument specifying the name of the alternate channel to use. The other keywords each accept an integer argument specifying a corresponding threshold. A message that exceeds any of these thresholds will be enqueued to the alternate channel instead of the original destination channel.

In the following channel block example, large messages over 5,000 blocks, that would have gone out the tcp_local channel to the Internet, instead go out the tcp_big channel:

```
tcp_local smtp ... rest of keywords ...   alternatechannel tcp_big
alternateblocklimit 5
tcp-daemon


tcp_big smtp ...rest of keywords...
tcp-big-daemon
```

Here are some examples of how the alternate* channel keywords can be used:

- If you want to deliver large messages at a delayed or an off-hours time, you can control when the alternatechannel (for example, tcp_big) runs.

One method is to use the `imsimta qm` utility's `STOP channel_name` and `START channel_name` commands, executing these commands periodically via your own custom periodic job that is run by the Job Controller or via a `cron` job.

- When you want the Job Controller to process large messages or messages with many recipients in their own pool, you might also use the `alternatechannel`.

  You can separate small messages or messages with few recipients from the large messages or messages with many recipients, since the latter might take longer for remote SMTP servers to process and accept; you might not want the larger messages to delay delivery of the smaller messages.

  Note that the Job Controller's regular scheduling of messages and assigning of messages to threads and processes are acceptable in most configurations.

- When you want to set special TCP/IP channel time-out values for large messages or for messages with many recipients, you can use the `alternatechannel`.

  In particular, setting special TCP/IP channel time-out values can be helpful if you want to send messages to remote hosts that take exceptionally long to receive large messages or messages with many recipients.

  Note that the default automatic time-out adjustment should be sufficient for most configurations. At most, you might want to adjust the values from the defaults and not use a special channel. In particular, see the channel options `STATUS_DATA_RECV_PER_ADDR_TIME` and `STATUS_DATA_RECV_PER_BLOCK_TIME` in the Messaging Server Reference Manual.

- When you want special MIME message fragmentation for especially large messages, you can use the `alternatechannel` and the `alternateblocklimit` channel keywords along with the `maxblocks` channel keyword.

  Typically, you would put the desired `maxblocks` size on your regular outbound TCP/IP channels, when you want to fragment messages over a specified size. The `maxblocks` channel keyword is normally both the threshold at which to perform fragmentation and the size to make the fragments.

  But, if you want to have a larger threshold trigger and make smaller actual fragments, you can use the `alternatechannel` and `alternateblocklimit` on the outbound TCP/IP channel. You can then use the `maxblock` size on your alternate channel to fragment messages over a particular size.

- You might use the alternatechannel in conjunction with special filtering. For instance, a message with many recipients might need more careful scrutiny of its content in case it is spam. You might want to do different filtering based on the outgoing channel (See the destinationfilter channel keyword in the *Sun Java System Messaging Server Administration Reference*.)

  If you are performing relatively resource-intensive scanning (such as virus filtering) via the conversion channel, very large messages might have a resource issue. You might want to use an alternate conversion channel. Or, you might want to do special conversion procedures within the regular conversion channel, based on the outgoing channel.

- You can use the alternatechannel when you want large outgoing messages to go out their own channel, so that they stand out when you analyze the mail.log* file or in counters displays.

  Furthermore, if you are trying to do careful analysis of delivery statistics, it is useful to process large messages in their own channel. This is because large messages or messages with many recipients that are sent to remote SMTP hosts are likely to take longer to finish processing, thus creating different delivery statistics for larger messages than for typical messages.

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:

- "Spreading a Channel Message Queue Across Multiple Subdirectories" on page 380

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 20, "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:

- "Specifying Mailbox Filter File Location" on page 382

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 13

Using Pre-defined Channels

When you first install Messaging Server, several channels are already defined (see Table 13-1). This chapter describes how to use pre-defined channel definitions in the MTA.

If you have not already read Chapter 10, "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 11, "Configuring Rewrite Rules"."

This chapter contains the following sections:

The `defaults` Channel is described in "Configuring Channel Defaults" on page 322.

Table 13-1 Predefined Channels

Channel	Definition
`defaults`	Used to specify which keywords are defaults for various channels. See "Configuring Channel Defaults" on page 322.
`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.)

Table 13-1 Predefined Channels

Channel	Definition
`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.
`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.
`sms`	Provides support for one-way email to an SMS gateway.

`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. The pipe channel performs delivery of messages using per-user, site-supplied programs.

To facilitate program delivery, you must first register programs as able to be invoked by the pipe channel. Do this by using the `imsimta program` utility. This utility gives a unique name to each command that you register as able to be invoked by the pipe channel. End users can then specify the method 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 *msg_svr_base*`/data/site-programs`. Make sure also 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, *msg_svr_base*/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 13-2 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 13-2 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 hold messages.

When messages are to be held, they are directed to the hold channel, in the *msg_svr_base*/queue/hold directory, using the same mechanism used to direct messages to the reprocess channel. In this way, the envelope To: addresses are unchanged. The messages are written to the hold channel queue, in the *msg-server*/queue/hold directory, as *ZZxxx*.HELD files. This prevents them from being seen by the job controller, and thus they are "held." Use the imsimta qm dir -held

command to view a list of .HELD files. These messages can be selected and released using the `imsimta qm -release` command. Releasing them changes their name to *ZZxxx*`.00` and informs the job controller. The messages are then processed by the master program associated with the hold channel, `reprocess.exe`. Thus the message (and the To: addresses) are processed using the normal rewriting machinery.

For more information on the `imsimta qm` command, see the *Sun Java System Messaging Server Administration Reference*.

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. Also, body parts are specified and delineated by MIME headers.) 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 182). For supplemental information on virus scanning using the `conversion` channel, refer to the Messaging Server Technical Notes at the bottom of the Messaging Server Documentation website at http://docs.sun.com/db/coll/S1_MsgTechNotes.

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.

This section consists of the following sections:

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"
```

NOTE MIME header lines are not the same as general, non-MIME header lines such as `To:`, `Subject:` and `From:`. Basically, for Conversion channel discussion, MIME header lines start with the string `Content-`.

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 t*he 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: *msg_svr_base*/conversions). The conversions file consists of line-separated entries that 1) qualify which types of body parts will be processed, and 2) 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. If value matches 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 13-1 conversions 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 13-6.) 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 13-1, 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 13-1) 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 13-1,

```
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 396) 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.

NOTE Whenever the `conversions` file is modified, you must recompile the configuration (see the `imsimta refresh` command in the *Sun Java System Messaging Server Administration Reference*).

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 13-3) 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 13-4 on page 400).

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=NAME; parameter-copy-0=*;
  dparameter-symbol-0=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=NAME specifies that the first Content-type parameter value (Draft1.doc in our example) be stored in an environment variable called 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=FILENAME specifies that the first Content-disposition parameter value (Draft1.doc in our example) be stored in an environment variable called 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 ORIGINAL_HEADERS.

`override-header-file=1` specifies that MIME headers are read from the file specified by environmental variable `OUTPUT_HEADERS`, overriding the original MIME header lines 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 MIME header lines changed during the conversion process. The conversion channel would then read the MIME header lines from this file when it re-assembles the body part. Note that only MIME header lines can be modified. Other general, non-MIME header lines cannot be cannot be altered by the conversion channel.

`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 399.

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

Table 13-3 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 MIME header lines (not `option=value` lines) followed by a blank line as its final line. Note also that only MIME header lines can be modified. Other general, non-MIME header lines cannot be cannot be altered by the conversion channel.
`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 399."

To Use Conversion Channel Output Options

Conversion channel output options (Table 13-4) 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 *msg_svr_base*/include/deprecated/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 402.)

A complete list of the output options is shown below.

Table 13-4 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 *msg_svr_base*/include/deprecated/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 ORIGINAL_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 ORIGINAL_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 a 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 *msg_svr_base*`/include/deprecated/pmdf_err.h`. The ones commonly used by the conversion channel are:

Table 13-5 Special Directives Commonly Used By the Conversion Channel

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'"
```

Automatic Arabic Character Set Detection

A new *auto_ef* program was added to automatically detect Arabic character sets.

You can call the auto_ef program from the conversion channel to automatically detect and label most unlabeled or incorrectly labeled text messages in Arabic character sets. These unlabeled or mislabeled messages are usually sent from Yahoo or Hotmail in Arabic.

Without the correct character set labeling, many mail clients cannot display the messages correctly.

If a message has MIME content-type headers, the auto_ef program examines and processes only those with text/plain content type. If the message is not labeled with a MIME content-type header, then auto_ef adds a text/plain content-type unconditionally.

To activate or enable this program, you must:

1. Edit your mappings file in the *msg_svr_base*/config directory to enable a conversion channel for the source and destination channel of your choosing. To enable a conversion channel for all mail coming in from the Internet to your local users, add a section to your mappings file similar to the following:

   ```
   CONVERSIONS

       IN-CHAN=tcp*;OUT-CHAN=ims-ms;CONVERT   YES
   ```

   Note that the IN and OUT channels depend on your configuration. If you are deploying on a relay MTA, you must modify the channels to fit your configuration. For example,

   ```
   IN-CHAN=tcp*;OUT-CHAN=tcp*;CONVERT YES
   ```

   Or, you could turn it on for all channels as follows:

   ```
   IN-CHAN=*;OUT-CHAN=*;CONVERT YES
   ```

2. Create a conversions file in the *msg_svr_base*/config directory that is owned and readable by the Messaging Server user, and that contains the following:

   ```
   !
   in-channel=*; out-channel=*;
     in-type=text; in-subtype=*;
     parameter-copy-0=*; dparameter-copy-0=*;
     original-header-file=1; override-header-file=1;
     command="msg_svr_base/lib/arabicdetect.sh"
   !
   ```

3. Compile your MTA configuration with the following command:

   msg_svr_base/sbin/imsimta cnbuild

4. Restart with the command:

   msg_svr_base/sbin/imsimta restart

Table 13-6 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.
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.

Table 13-6 Conversion Parameters *(Continued)*

Parameter	Description
TAG	Input tag, as set by a mail list CONVERSION_TAG parameter.
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.
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. Use / to specify paths, not \. Example: command="D:/tmp/mybat.bat"
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.)

Table 13-6 Conversion Parameters *(Continued)*

Parameter	Description
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. Use / to specify paths, not \. Example: command="D:/tmp/mybat.bat"
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 ORIGINAL_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 13-7 lists the keywords.

Table 13-7 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. This keyword only applies to message parts that are already encoded. Messages with Content-transfer-encoding: 7BIT or 8bit do not require any special encoding and therefore this BASE64 option will have no effect on them.

Table 13-7 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 14

Integrating Anti-spam and Anti-virus Programs

This chapter describes how to integrate and configure the Brightmail and SpamAssassin anti-spam / anti-virus programs using the software hooks provided by Messaging Server.

NOTE In this chapter, references to anti-spam features also mean, when applicable, anti-virus features. Brightmail offers both anti-spam and anti-virus features. SpamAssassin only offers anti-spam.

From the perspective of Messaging Server, most anti-spam solutions do much the same thing. Messaging Server sends copies of messages to the anti-spam software which analyzes a message and returns a verdict of spam, not spam. (SpamAssassin also returns a verdict along with a *spam score,* which is a numerical rating of the probability of a message being spam.) Messaging Server reads the verdict and takes some type of action using a Sieve script. This chapter describes the hooks provided by Messaging Server to use these commercial anti-spam programs as well as configuration examples.

This chapter is divided in to the following sections:

Deploying and Configuring Third Party Anti-spam Programs

In terms of deployment and configuration of third party anti-spam programs, there are only a few administrator decisions required. These decisions are as follows:

- "Determining How Many Anti-Spam Servers to Deploy." See your Brightmail representative or the SpamAssassin documentation for guidelines on determining the hardware requirements for filtering mail at your site.
- "Specifying the Messages to Be Filtered" on page 418
- "Specifying Actions to Perform on Spam Messages" on page 424

NOTE Previous versions of Messaging Server only supported the Brightmail filtering technology and hence keywords and options had names such as `sourcebrightmail` or `Brightmail_config_file`. These keywords and options have been changed to more generic to names such as `sourcespamfilter` or `spamfilter_config_file` as applicable. The previous Brightmail names are retained for compatibility.

Specifying the Messages to Be Filtered

Messaging Server provides a number ways to specify which messages should be filtered. The system can be configured to filter messages by user, domain, or channel. This section contains the following sections:

- "To Specify User-level Filtering" on page 419
- "To Specify Domain-level Filtering" on page 420
- "To Specify Channel-level Filtering" on page 421

NOTE In this discussion the expression *optin* means to specify that a user, domain or channel receives mail filtering.

To Specify User-level Filtering

It is possible to specify which users receive spam filtering. An example of this type of usage would be if anti-spam or anti-virus filtering was offered as a premium service to ISP customers. The general steps for specifying user-level filtering is as follows:

1. Specify the LDAP attribute that activates spam filter processing on specified users with the `LDAP_OPTIN` in the `option.dat` file:

   ```
   LDAP_OPTIN=mailAntiUBEService
   ```

2. Set `mailAntiUBEService` in the user entries to receive spam filtering.

   The values for `mailAntiUBEService` will depend on the server. For Brightmail, the valid values are `spam` (filter for spam) and `virus` (filter for viruses). For SpamAssassin, it can be any string, but `spam` is recommended for clarity. When used as a multi-valued attribute (Brightmail), each value requires a separate attribute value entry. For example:

   ```
   mailAntiUBEService: spam
   mailAntiUBEService: virus
   ```

Example

This example assumes Brightmail is being used. It also assumes that `LDAP_OPTIN` was set to `mailAntiUBEService` in the `option.dat` file. The user, Otis Fanning, has the `mailAntiUBEService` attribute set to `spam` and `virus` in his user entry. His mail will be filtered for spam and viruses. Code Example 14-3 shows the Brightmail enabled user entry for Otis Fanning.

Code Example 14-1 Example LDAP User Entry for Brightmail

```
dn: uid=fanning,ou=people,o=sesta.com,o=ISP
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: inetUser
objectClass: ipUser
objectClass: inetMailUser
objectClass: inetLocalMailRecipient
objectClass: nsManagedPerson
objectClass: userPresenceProfile
cn: Otis Fanning
sn: fanning
initials: OTF
givenName: Otis
pabURI: ldap://ldap.siroe.com:389/ou=fanning,ou=people,o=sesta.com,o=isp,o=pab
mail: Otis.Fanning@sesta.com
mailAlternateAddress: ofanning@sesta.com
mailDeliveryOption: mailbox
```

Code Example 14-1 Example LDAP User Entry for Brightmail

```
mailHost: manatee.siroe.com
uid: fanning
dataSource: iMS 5.0 @(#)ims50users.sh 1.5a 02/3/00
userPassword: password
inetUserStatus: active
mailUserStatus: active
mailQuota: -1
mailMsgQuota: 100
mailAntiUBEService: virus
mailAntiUBEService: spam
```

If SpamAssassin was used, the entry would be the same, but without **`mailAntiUBEService: virus`.** See the section in this chapter describing your third-party software for more examples and details.

To Specify Domain-level Filtering

It is possible to specify which domains receive spam filtering. An example of using this feature would be if anti-spam or anti-virus filtering was offered as a premium service to ISP domain customers. The general steps for specifying domain-level filtering is as follows:

1. Specify the LDAP attribute that activates spam filter processing on specified domains with the `LDAP_OPTIN` in the `option.dat` file.

   ```
   LDAP_DOMAIN_ATTR_OPTIN=mailAntiUBEService
   ```

2. Set `mailAntiUBEService` in the domain entries to receive spam filtering.

   Valid values for `mailAntiUBEService` depend on the server. For Brightmail, the valid values are `spam` (filter for spam) and `virus` (filter for viruses). For SpamAssassin, it can be any string, but `spam` is recommended for clarity. When used as a multi-valued attribute (Brightmail), each value requires a separate attribute value entry. For example:

   ```
   mailAntiUBEService: spam
   mailAntiUBEService: virus
   ```

Domain-level Filtering Example

This example assumes Brightmail is being used. It also assumes that `LDAP_DOMAIN_ATTR_OPTIN` was set to `mailAntiUBEService` in the `option.dat` file. The `mailAntiUBEService` attribute is set to `spam` and `virus` in the `sesta.com` domain entry in the DC tree for Sun LDAP Schema 1. For Sun LDAP Schema 2 you would also set `mailAntiUBEService` in the domain entries to receive spam filtering.

All mail sent to sesta.com is filtered for spam and viruses by Brightmail. Code Example 14-4 shows the domain entry.

Code Example 14-2 Example LDAP Domain Entry for Brightmail

```
dn: dc=sesta,dc=com,o=internet
objectClass: domain
objectClass: inetDomain
objectClass: mailDomain
objectClass: nsManagedDomain
objectClass: icsCalendarDomain
description: DC node for sesta.com hosted domain
dc: sesta
inetDomainBaseDN: o=sesta.com,o=isp
inetDomainStatus: active
mailDomainStatus: active
mailDomainAllowedServiceAccess: +imap, pop3, http:*
mailRoutingHosts: manatee.siroe.com
preferredMailHost: manatee.siroe.com
mailDomainDiskQuota: 100000000
mailDomainMsgQuota: -1
mailClientAttachmentQuota: 5
mailAntiUBEService: spam
mailAntiUBEService: virus
```

If SpamAssassin was used, the entry would be the same, but without **`mailAntiUBEService: virus.`** See the section in this chapter describing your particular third-party software for more examples and details.

To Specify Channel-level Filtering

Specifying spam filtering by user or domain is an obvious criteria, but specifying by channel is less obvious. The reason to specify filtering by source or destination channel is to provide greater flexibility and granularity for spam filtering.

Messaging Server allows you to specify filtering by source or destination channel. The mechanism for doing this are the channel keywords described in Table 14-1. The following example demonstrates how to set up channel-level filtering.

1. Add a rewrite rule in the `imta.cnf` file for all inbound SMTP servers that sends messages to a specific backend message store host. Example:

   ```
   msg_store1.siroe.com   $U@msg_store1.siroe.com
   ```

2. Add a channel corresponding to the rewrite rule with the `destinationspamfilteroptin` keyword. Example:

```
tcp_msg_store1 subdirs 20 backoff “pt5m” “pt10” “pt30” “pt1h” \
“pt2h” “pt4h” maxjobs 1 pool IMS_POOL fileinto $U+$S@$D \
destinationspamfilteroptin spam
msg_store1.siroe.com
```

Table 14-1 MTA Channel Keywords for Spam Filters

Channel Keyword	Description
`destinationspamfilteroptin`	Specifies that all messages destined to this channel are filtered even if those services are not specified by the user or domain with the `LDAP_OPTIN` LDAP attribute. The filter parameters follow the keyword. The available parameters depend on the filtering program. For example, Brightmail parameters are `spam` or `virus` or `spam,virus`. The SpamAssassin parameter is `spam`.. In this example, all mail destined for the message store is scanned for spam and virus by Brightmail: `ims-ms destinationspamfilteroptin spam,virus. . .`
`sourcespamfilteroptin`	Specifies that all messages originating from this channel are filtered even if those services are not specified by the user or domain with the `LDAP_OPTIN` LDAP attribute.The system-wide default parameters follow the keyword, and the available parameters depend on the filtering program. For example, for Brightmail parameters are `spam` or `virus` or `spam,virus`. For SpamAssassin, the parameter is `spam`. If `switchchannel` is in effect, this keyword is placed on the switched-to channel.

Channel-level Filtering Examples

Example 1. Use Brightmail to filter all mail for spam and viruses from an MTA relay to a backend message store called `siroemail`. A rewrite rule is setup to send these messages through an MTA channel called `tcp_siroemail`:

1. Add a rewrite rule in the `imta.cnf` file that sends messages to a backend message store host. Example:

```
msg_store1.siroe.com   $U@msg_store1.siroe.com
```

2. Add a channel corresponding to that rewrite rule with the `destinationspamfilteroptin` keyword. Example:

```
tcp_msg_store1 subdirs 20 backoff “pt5m” “pt10” “pt30” “pt1h” \
“pt2h” “pt4h” maxjobs 1 pool IMS_POOL fileinto $U+$S@$D \
destinationspamfilteroptin spam, virus
msg_store1.siroe.com
```

Example 2. Filter all incoming mail passing through your MTA:

```
tcp_local smtp mx single_sys remotehost inner switchchannel \
identnonelimited subdirs 20 maxjobs 7 pool SMTP_POOL \
maytlsserver maysaslserver saslswitchchannel tcp_auth \
sourcespamfilteroptin spam
tcp-daemon
```

Example 3. Filter all outgoing (to the internet) mail passing through your MTA:

```
tcp_local smtp mx single_sys remotehost inner switchchannel \
identnonelimited subdirs 20 maxjobs 7 pool SMTP_POOL \
maytlsserver maysaslserver saslswitchchannel tcp_auth \
destinationspamfilteroptin spam
tcp-daemon
```

Example 4. Filter all incoming and outgoing mail passing through your MTA:

```
tcp_local smtp mx single_sys remotehost inner switchchannel \
identnonelimited subdirs 20 maxjobs 7 pool SMTP_POOL \
maytlsserver maysaslserver saslswitchchannel tcp_auth \
sourcespamfilteroptin spam destinationspamfilteroptin spam
tcp-daemon
```

Example 5. Filter mail of users or domains with the `mailAntiUBEService` attribute destined to the local message store:

```
tcp_local smtp mx single_sys remotehost inner switchchannel \
identnonelimited subdirs 20 maxjobs 7 pool SMTP_POOL \
maytlsserver maysaslserver saslswitchchannel tcp_auth \
destinationspamfilteroptin spam
tcp-daemon
```

Example 6. Filter all mail destined to the local message store in a two-tiered system without using per-user optin:

```
ims-ms smtp mx single_sys remotehost inner switchchannel \
identnonelimited subdirs 20 maxjobs 7 pool SMTP_POOL \
maytlsserver maysaslserver saslswitchchannel tcp_auth \
destinationspamfilteroptin spam
tcp-daemon
```

Example 7. Use Brightmail to filter all incoming and outgoing mail for spam and viruses:

```
tcp_local smtp mx single_sys remotehost inner switchchannel \
identnonelimited subdirs 20 maxjobs 7 pool SMTP_POOL \
maytlsserver maysaslserver saslswitchchannel tcp_auth \
destinationspamfilteroptin spam,virus sourcespamfilteroptin \
spam,virus
tcp-daemon
```

Specifying Actions to Perform on Spam Messages

As discussed earlier, anti-spam programs analyze messages and return a verdict of spam or not spam to Messaging Server. Messaging Server then takes action on the message. Anti-spam programs typically return a string or a null value to the MTA to indicate that message is spam. Some programs also return a spam score—a number rating the probability of the message being spam or not. This score can be used as part of the action sequence.

Actions are specified using the Sieve mail filtering language. Possible Sieve actions are to discard the message, file it into a folder, add a header, add a tag to the subject line, and so on. Complex Sieve scripts with if-then-else statements are also possible. Refer to RFC 3028 for the complete Sieve syntax.

Sieve scripts are specified with the MTA spam filter options described in Table 14-2. The primary spam filter action options are `Spamfilter_null_action`, which specifies the Sieve rule to execute when a null value is returned as the spam verdict value, and `Spamfilter_null_action`, which specifies the Sieve rule to execute when a string is returned as the spam verdict. Both values must be stored in `option.dat`.

Example 1: This option line files spam messages with a null verdict value to the file `SPAM_CAN`.

```
spamfilter_null_action=data:,require "fileinto"; fileinto "SPAM_CAN”;
```

The same action can be performed on a spam message that returns a string:

```
spamfilter_string_action=data:,require "fileinto"; fileinto "SPAM_CAN”;
```

Example 2: This option line files spam messages with a string verdict into a file named after that verdict string returned to the MTA (this is what `$U` does). That is, if the verdict string returned is `spam`, the message is stored in a file called `spam`.

```
spamfilter_null_action=data:,require "fileinto"; fileinto "$U”;
```

Example 3: This line discards spam messages with a string verdict value.

```
spamfilter_string_action=data:,discard
```

Example 4. The line below adds the header `Spam-test: FAIL` to each message determined to be spam a string verdict value.

```
spamfilter_string_action=data:,require ["addheader"];addheader "Spam-test: FAIL”;
```

Example 5. The line below adds the string `[PROBABLE SPAM]` to the subject line of the spam messages.

```
spamfilter_string_action=data:,addtag “[PROBABLE SPAM]”;
```

Example 6. This line assumes a string verdict value and files a spam message in the mailbox `testspam` if the header contains `resent-from` and `User-1`. If it does not have that header, it simply files the message into `spam`.

```
spamfilter_string_action=data:,require "fileinto";\
  if header :contains ["resent-from"] ["User-1"] {\
  fileinto "testspam";\
  } else {\
  fileinto "spam";};
```

Because verdict strings are configurable with most spam filter software, you can specify different actions depending on the string returned. This can be done with the matched pairs `spamfilter_verdict_`*n* and `spamfilter_action_`*n* options.

Example 7. These matched pair options discard spam messages with the returned verdict string `remove`. Refer to the specific spam filter sections for instructions on how to specify the spam verdict string.

```
spamfilter_verdict_0=kill
spamfilter_action_0=data:,discard
```

Table 14-2 MTA Spam Filter Options (`options.dat`)

MTA Options for Spam Assassin	Description
`Spamfilter_config_file`	Full file path and name of the SpamAssassin configuration file. Default: none
`Spamfilter_library`	Full file path and name of the SpamAssassin shared library. Default: none
`Spamfilter_optional`	Controls whether certain failures reported by the filtering library are treated as a temporary processing failure or ignored. The default value of `0` specifies that spam filtering problems will cause a temporary processing failure. Changing the value to `1` causes spam filter processing to be skipped in the event of some, but possibly not all, filtering library failures. (In particular, if the system gets stuck without a return in the library code, some portion of the MTA may also get stuck.) Default: `0`
`LDAP_optin`	The name of the LDAP attribute used to activate SpamAssassin on a per-user basis. This should1 be an attribute in the `inetMailUser` objectclass. If you do not have predefined attribute, use `mailAntiUBEService`. The attribute itself (example: `mailAntiUBEService`) can take multiple values and is case-sensitive. For SpamAssassin, its value should be `spam` in lowercase. Default: none
`LDAP_domain_attr_optin`	The name of the LDAP attribute used to activate SpamAssassin on a per-domain basis. It applies to the destination domain. It is just like `LDAP_optin`, described above, except it should be in the objectclass `mailDomain`. Default: none
`Spamfilter_null_optin`	Specifies a string, which if found as a value of the attribute `mailantiUBEservice`, causes the MTA to act as if the attribute wasn't there. That is, it disables filtering for that entry. See "Specifying Which Messages Will Be Filtered by SpamAssassin" on page 441 for usage details. Default: The empty string. Empty optin attributes are ignored by default. (This is a change from iPlanet Messaging Server 5.2, where empty optin attributes triggered filtering with an empty optin list. The 5.2 behavior can be restored by setting `SPAMFILTER_NULL_OPTIN` to a string that never occurs in practice.)
`Spamfilter_null_action`	Sieve rule specifying what to do with the message when the SpamAssassin verdict returns as null. Sieve expressions can be stored externally using a file URL. For example: `file:///var/opt/SUNWmsgsr/config/null_action.sieve`. Also, do not reject spam using the Sieve reject action, as it tends to deliver a nondelivery notification to the innocent party whose server has been used to send the spam. that has been used to send the spam. Default: `data:,discard;`

Table 14-2 MTA Spam Filter Options (`options.dat`)

MTA Options for Spam Assassin	Description
`Spamfilter_string_action`	Sieve rule specifying what to do with the message if the verdict is a string. Sieve expressions can be stored externally using a file URL. For example: `file:///var/opt/SUNWmsgsr/config/null_action.sieve`. Also, do not reject spam using the Sieve reject action, as it tends to deliver a nondelivery notification to the innocent party whose server has been used to send the spam. that has been used to send the spam. Default: `data:,require "fileinto"; fileinto "$U;` `where $U` is the string that `verdict` returned.

`spamfilter_verdict_`*n*	`spamfilter_verdict_`*n* and `spamfilter_action_`*n* are matched pairs, where *n* is a number from 0 to 9. These options allow you to specify Sieve filters for arbitrary verdict strings. This is done by setting `spamfilter_verdict_`*n* and `spamfilter_action_`*n* to the verdict string and sieve filter, respectively, where n is an integer from 0 to 9. For example, a site could have the "reject" verdict cause a sieve reject action by specifying:
	`spamfilter_verdict_`*0*`=reject`
	`spamfilter_action_`*0*`=data:,require "reject"; reject "Rejected by spam filter";`
	The default values for all of the `spamfilter_verdict_`*0* options and the corresponding action options are empty strings.
	Default: none

`spamfilter_action_`*n*	See `spamfilter_verdict_n`. Default: none
`spamfilter_final`	Some filtering libraries have the ability to perform a set of actions based on recipient addresses. `spamfilter_final` specifies the sort of recipient address passed to the filtering library. `0` results in a so-called intermediate address being used; `1` sends the final form of the recipient address. Default: `0`

Table 14-2 MTA Spam Filter Options (options.dat)

MTA Options for Spam Assassin	Description
optin_user_carryover	Forwarding is a challenge for spam filter processing. Consider a user entry that specifies the forward delivery option and specifies the forwarding address of another user. Additionally, the user entry opts in to some specific sort of filtering. Should the filtering be applied to the forwarded message or not? On the one hand, the correct filtering choice for one particular user may not be the correct choice for another. On the other hand, eliminating a filtering operation might be used as means of violating a site's security policy. No single answer is correct in all cases so OPTIN_USER_CARRYOVER, controls how the spam filtering optin list is carried from one user/alias entry to another when forwarding occurs. This is a bit-encoded value. The various bit values have the following meanings: bit 0 (value 1). Each LDAP user entry override any previously active user/domain optins unconditionally. bit 1 (value 2). If a user's domain has an optin attribute, it overrides any previous user/domain/alias optins that were active. bit 2 (value 4). If a user has an optin attribute, it overrides any previous user/domain/alias optins that were active. bit 3 (value 8). An optin specified by an [optin] non-positional parameter overrides any previous user/domain/alias optins that were active. Default: 0 (optins will accumulate if one user has a delivery option that forwards to another. The default insures that site security policies will be effective when forwarding; other settings may not.)

Using Brightmail

Brightmail Inc. is a company that provides an anti-spam and anti-virus software solution for email servers. The Brightmail solution consists of the Brightmail server along with realtime anti-spam and anti-virus rule updates downloaded to email servers.

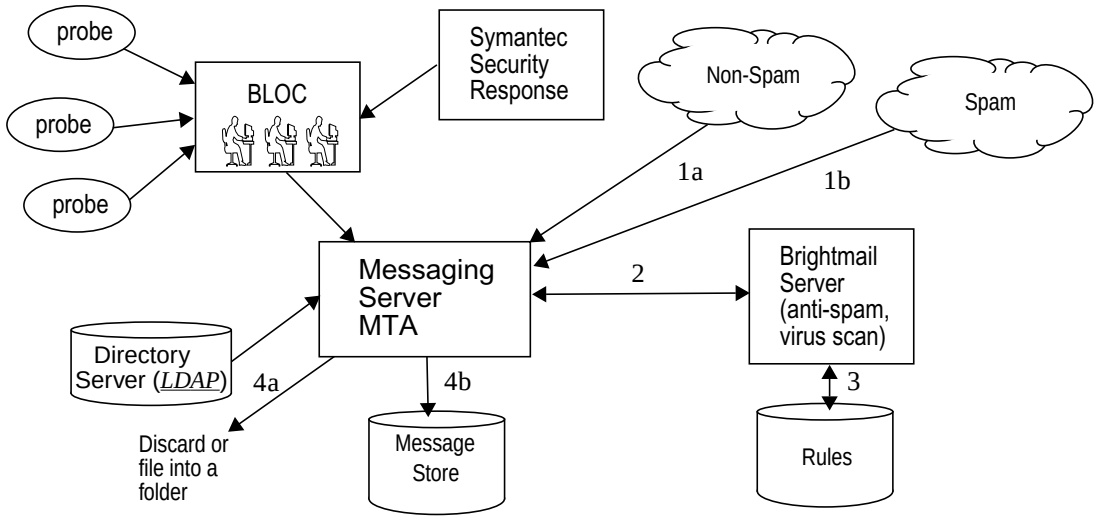
How Brightmail Works

The Brightmail server is deployed at a customer site. Brightmail has email probes set around the internet for detection of new spam. Brightmail technicians create custom rules to block this spam in realtime. These rules are downloaded to Brightmail servers, also in realtime. The Brightmail database is updated and Brightmail server runs this database filter against the email for the specified users or domains.

Brightmail Architecture

Figure 14-1 depicts the Brightmail architecture.

Figure 14-1 Brightmail and Messaging Server Architecture



When the Brightmail Logistics and Operations Center (BLOC) receives spam from email probes, operators immediately create appropriate anti-spam rules, which are downloaded to Brightmail customer machines. Similarly, the Symantec Security Response realtime virus rules are also sent from Brightmail. These rules are used by customer's Brightmail servers to catch spam and viruses.

The MTA uses the Brightmail SDK to communicate with the Brightmail Server. The MTA dispatches messages based on the response back from Brightmail. After the mail (1a) or (1b) is received by the MTA, the MTA sends the message to the Brightmail server (2). The Brightmail server uses its rules and data to determine if the message is a spam or virus (3), and returns a verdict back to the MTA. Based on the verdict, the MTA either (4a) discards the message or files the message into a folder, or (4b) delivers it normally to the destination.

Since the Brightmail SDK is third party software, we do not include it in our installation kit. The Brightmail SDK and server software must be obtained from Brightmail Inc. The MTA has configuration settings to tell it whether and where to load the Brightmail SDK to enable Brightmail integration.

Once the SDK is loaded, Brightmail message processing is determined by several factors and levels of granularity (the term used by Brightmail to specify active processing is *optin*). This is specified by the following criteria:

- Whether the source or destination channel is enabled for Brightmail (`imta.cnf`)
- Whether there is a channel default for the services opted in (`imta.cnf`)
- Whether there is a per domain optin (LDAP)
- Whether there is per-user optin (LDAP)

For any particular message recipient, the optins and defaults above are combined, which means, if the channel default is already specified for both spam and virus, then there is no reason to bother with per-user optin. That is, if the system administrator decides to do spam and virus filtering for everyone, then there is no reason to expose to the user the ability to optin for spam or virus. There is no way to opt out of processing, that is, you can not say you do not want the service if a user is already optin via a system or domain optin. This also means that if you are optin for a service, and you have forwarded your mail to another address, that address would get the mail after the filtering has been performed on your behalf.

There are only two services offered, virus or spam detection. Brightmail also provides “content-filtering” service, but this functionality is provided using Sieve, so there is no added value to have Brightmail do the Sieve filtering.

When a message is determined to contain a virus, the Brightmail server can be configured to clean the virus and resubmit the cleaned message back to the MTA. (Due to some undesirable side effects caused by loss of information about the original message in a resubmitted cleaned message, we recommend you do not configure Brightmail to resubmit the cleaned message back to the MTA.) When the message is spam, the verdict back from the Brightmail along with the configuration in Brightmail allows the MTA to determine what happens to the message. The message can be discarded, filed into a folder, tagged as spam or virus on the subject line, passed to a Sieve rule, delivered normally in the `INBOX`, and so on.

The Brightmail servers can be located on the same system as the MTA, or it can be on a separate system. In fact, you can have a farm of Brightmail servers serving one or more MTAs. The Brightmail SDK uses the Brightmail configuration file to determine which Brightmail servers to use.

Brightmail Requirements and Performance Considerations

- Brightmail servers must run on the Solaris Operating System.

- If Brightmail implements both spam and virus checking, MTA message throughput can be reduced by as much 50%. To keep up with MTA throughput, you may need two Brightmail servers for each MTA.
- While SpamAssassin has the ability to perform different sorts of filtering on a per-user basis, it is unable to apply two different sets of filtering criteria to the same message at the same time. Thus, SpamAssassin only allows system-wide filtering. Customized filtering for individual users is not possible.

Deploying Brightmail

This section describes how to deploy Brightmail for the following configurations:

- "To Activate Brightmail Processing for Users on a Destination or Source Channel" on page 431
- "To Activate Brightmail Processing for Selected Users" on page 432
- "To Activate Brightmail Processing for Selected Domains on a System" on page 434

Brightmail filtering is enabled in Messaging Server using the channel or an LDAP attribute. The method of filtering on the system is additive. That is, it is the combination of both keywords and the attribute.

To Activate Brightmail Processing for Users on a Destination or Source Channel

1. Install and Configure Brightmail server.

   To install Brightmail on your system, see your Brightmail representative.

2. Set the Brightmail Library and Configuration File Parameters by adding the following two MTA options to the `options.dat` file:

   `spamfilter_Library=`*path_and_filename_of_libbmiclient.so*
   `spamfilter_config_file=`*path_and_filename_of_brightmail_config_file*

3. Specify the desired Brightmail options in the MTA options file (Table 14-2 on page 426) and Brightmail configuration file (Table 14-3 on page 437).

4. Specify the channels and email direction (source or destination) on which Brightmail processing will occur.

   Set the keyword `sourcespamfilteroptin` or `destinationspamfilteroptin` on a channel block (see "To Specify Channel-level Filtering" on page 421 and "The MTA Configuration File" on page 222)

   `sourcespamfilteroptin` specifies that every message coming from the channel be processed by Brightmail software.

   `destinationspamfilteroptin` specifies that every message going to the channel be processed by Brightmail software.

   Valid values for these attributes are as follows:

   `spam` - filter for spam
   `virus` - filter for viruses
   `spam,virus` - filter for spam and viruses

Examples

Example 1. Filter mail coming from an incoming MTA relay and going to a specific backend message store called `siroe` for spam and viruses. The messages will pass through the MTA channel called `tcp_siroemail`:

```
tcp_siroemail smtp mx single_sys remotehost inner switchchannel \
identnonelimited subdirs 20 maxjobs 7 pool SMTP_POOL \
maytlsserver maysaslserver saslswitchchannel tcp_auth \
destinationspamfilteroptin spam,virus
tcp_siroemail-daemon
```

Example 2. Filter all incoming mail going through your MTA. from the `tcp_local` channel will be filtered by the Brightmail for spam:

```
tcp_local smtp mx single_sys remotehost inner switchchannel \
identnonelimited subdirs 20 maxjobs 7 pool SMTP_POOL \
maytlsserver maysaslserver saslswitchchannel tcp_auth \
sourcespamfilteroptin spam
tcp-daemon
```

To Activate Brightmail Processing for Selected Users

1. Install and configure Brightmail software.

   To install Brightmail on your system, see your Brightmail representative.

2. Set the Brightmail library and configuration file parameters.

   Use the following two MTA options in the `options.dat` file:

   `spamfilter_Library=`*`path_and_filename_of_libbmiclient.so`*
   `spamfilter_config_file=`*`path_and_filename_of_brightmail_config_file`*

3. Specify the desired Brightmail options in the MTA options file (Table 14-2 on page 426) and Brightmail configuration file (Table 14-3 on page 437).

4. Specify the LDAP attribute that will be used to activate Brightmail processing on specified users.

   Set `LDAP_OPTIN=mailAntiUBEService` in the `option.dat` file. It is possible to specify an LDAP attribute other than `mailAntiUBEService`, but we recommend using this name.

5. Set LDAP attribute `mailAntiUBEService` in the user entries to receive Brightmail processing.

   Valid values for `mailAntiUBEService` are `spam` (filter for spam) and `virus` (filter for viruses).

Example

Assume that `LDAP_OPTIN` was set to `mailAntiUBEService` in the `option.dat` file. If the user, Otis Fanning, has the `mailAntiUBEService` attribute set to `spam` and `virus` in his user entry, then his mail will be filtered by Brightmail for spam and viruses. Code Example 14-3 shows the Brightmail enabled user entry for Otis Fanning.

Code Example 14-3 Example LDAP User Entry for Brightmail

```
dn: uid=fanning,ou=people,o=sesta.com,o=ISP
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: inetUser
objectClass: ipUser
objectClass: inetMailUser
objectClass: inetLocalMailRecipient
objectClass: nsManagedPerson
objectClass: userPresenceProfile
cn: Otis Fanning
sn: fanning
initials: OTF
givenName: Otis
pabURI: ldap://ldap.siroe.com:389/ou=fanning,ou=people,o=sesta.com,o=isp,o=pab
mail: Otis.Fanning@sesta.com
mailAlternateAddress: ofanning@sesta.com
mailDeliveryOption: mailbox
mailHost: manatee.siroe.com
```

Code Example 14-3 Example LDAP User Entry for Brightmail

```
uid: fanning
dataSource: iMS 5.0 @(#)ims50users.sh 1.5a 02/3/00
userPassword: password
inetUserStatus: active
mailUserStatus: active
mailQuota: -1
mailMsgQuota: 100
mailAntiUBEService: virus
mailAntiUBEService: spam
```

To Activate Brightmail Processing for Selected Domains on a System

1. Install and configure Brightmail software.

   To install Brightmail on your system, see with your Brightmail representative.

2. Set the Brightmail library and configuration file parameters.

   Set the following two MTA options in the `options.dat` file:

   `spamfilter_Library=`*path_and_filename_of_libbmiclient.so*
   `spamfilter_config_file=`*path_and_filename_of_brightmail_config_file*

3. Specify the desired Brightmail options in the MTA options file (Table 14-2 on page 426) and Brightmail configuration file (Table 14-3 on page 437).

4. Specify the LDAP attribute that will be used to activate Brightmail processing on specified domains.

   Set `LDAP_DOMAIN_ATTR_OPTIN=mailAntiUBEService` in the `option.dat` file. It is possible to specify a different LDAP attribute name, but we recommend using this name so that the Messaging Server schema remains consistent.

5. Set the LDAP attribute `mailAntiUBEService` in the domain entries (in the DC tree) whose email will receive Brightmail processing.

   Valid values for `mailAntiUBEService` are `spam` (filter for spam) and `virus` (filter for viruses).

Example

Assume that `LDAP_DOMAIN_ATTR_OPTIN` was set to `mailAntiUBEService` in the `option.dat` file. The `mailAntiUBEService` attribute is set to `spam` and `virus` in the example.com domain entry in the DC tree. Code Example 14-4 shows the Brightmail-enabled domain entry.

Code Example 14-4 Example LDAP Domain Entry for Brightmail

```
dn: dc=sesta,dc=com,o=internet
objectClass: domain
objectClass: inetDomain
objectClass: mailDomain
objectClass: nsManagedDomain
objectClass: icsCalendarDomain
description: DC node for sesta.com hosted domain
dc: sesta
inetDomainBaseDN: o=sesta.com,o=isp
inetDomainStatus: active
mailDomainStatus: active
mailDomainAllowedServiceAccess: +imap, pop3, http:*
mailRoutingHosts: manatee.siroe.com
preferredMailHost: manatee.siroe.com
mailDomainDiskQuota: 100000000
mailDomainMsgQuota: -1
mailClientAttachmentQuota: 5
mailAntiUBEService: spam
mailAntiUBEService: virus
```

Example Brightmail Deployment Scenarios

There are several common deployment Brightmail scenarios that are discussed in this section. In each case you will do the standard Brightmail configuration procedures like install Brightmail and specify the library and configuration files, (see previous sections). This section describes the following deployment scenarios:

- “Brightmail Processing on Local Incoming Messages” on page 435
- “Brightmail Processing on Incoming Messages from the Internet” on page 436
- “Brightmail Processing on Outgoing Messages to the Internet” on page 436
- “Brightmail Processing on Incoming Messages to a Specific Backend Message Store Host” on page 436
- “Adding a Header to Spam Messages” on page 437

Brightmail Processing on Local Incoming Messages

You may wish to configure your system so that all mail delivered locally is screened for spam and viruses. To set up Brightmail processing of all incoming messages to the local message store (that is, to the ims-ms channel in `imta.cnf`), add the `destinationspamfilteroptin` keyword to the `ims-ms` channel definition. Example:

```
ims-ms defragment subdirs 20 backoff “pt5m” “pt10” “pt30” “pt1h” \
“pt2h” “pt4h” maxjobs 1 pool IMS_POOL fileinto $U+$S@$D filter \
ssrd:$A ims-ms-daemon destinationspamfilteroptin spam,virus
ims-ms-daemon
```

Brightmail Processing on Incoming Messages from the Internet

You may wish to configure your system so that all mail coming from the internet is screened for spam. To set up Brightmail processing of all incoming messages from the internet, add the `sourcespamfilteroptin` keyword to the `tcp-local` channel definition. Example:

```
tcp_local smtp mx single_sys remotehost inner switchchannel \
identnonelimited subdirs 20 maxjobs 7 pool SMTP_POOL maytlsserver \
maysaslserver saslswitchchannel tcp_auth sourcespamfilteroptin spam
tcp-daemon
```

NOTE Brightmail allows you to either discard spam messages, or save them in a designated spam folder. If the ability to designate a spam folder is not available for the receiving system, then the address syntax for the spam folder will be meaningless to that system.

Brightmail Processing on Outgoing Messages to the Internet

You may wish to configure your system so that all mail going to the internet is screened for spam. To set up Brightmail processing of all outgoing messages to the internet, add the `destinationspamfilteroptin` keyword to the `tcp-local` channel definition on the outgoing MTA. Example:

```
tcp_local smtp mx single_sys remotehost inner switchchannel \
identnonelimited subdirs 20 maxjobs 7 pool SMTP_POOL maytlsserver \
maysaslserver saslswitchchannel tcp_auth \
destinationspamfilteroptin spam
tcp-daemon
```

Brightmail Processing on Incoming Messages to a Specific Backend Message Store Host

To configure your system so that all mail coming into a specific backend message store host is screened for virus and spam, do the following:

1. Add a rewrite rule in the `imta.cnf` file of all inbound SMTP servers that will send messages to the backend message store host. Example:

   ```
   msg_store1.siroe.com   $U@msg_store1.siroe.com
   ```

2. Add a channel corresponding to that rewrite rule with the `destinationspamfilteroptin` keyword. Example:

```
tcp_msg_store1 subdirs 20 backoff “pt5m” “pt10” “pt30” “pt1h” \
“pt2h” “pt4h” maxjobs 1 pool IMS_POOL fileinto $U+$S@$D \
destinationspamfilteroptin spam,virus
msg_store1.siroe.com
```

Adding a Header to Spam Messages

You can add an arbitrary header to a spam message. To add the header *spam-result: Brightmail says this is spam* to a message caught by Brightmail software, add the following to `option.dat`:

```
spamfilter_string_action=data:,require ["addheader" "spamtest"];addheader
"spam-result: Brightmail says this is spam";
```

Only the text after `"spamtest"];` are customizable.

Brightmail Configuration Options

Selected Brightmail configuration file options are shown in Table 14-3. The latest and complete listing of Brightmail configuration file options can be obtained from Brightmail.

Table 14-3 Selected Brightmail Configuration File Options

Brightmail Option (Not Case-sensitive)	Description (value of the attributes are case-sensitive)
`blSWPrecedence`	A given message can have multiple verdicts. This specifies the precedence order. So if a message is processed for virus first, then for spam if you specified this option as `virus-spam` the verdicts are separated by hyphens (-). This is the recommended setting when using Brightmail with Sun Java System Messaging Server.
`blSWClientDestinationDefault`	Specifies how to deliver normal messages, that is, not a spam or virus, and thus have no verdict. Usually you want to deliver this message normally, so you would specify `inbox` as the value. There is no default.
`blSWLocalDomain`	This attribute specifies what domain(s) are considered to be local. There can be multiple lines of this attribute specifying several domains which are all considered local. Local versus foreign domain is used to specify two different handling for a verdict. See below `blSWClientDestinationLocal` and `blSWClientDestinationForeign`. For example, you can specify `blSWLocalDomain=siroe.com`

Table 14-3 Selected Brightmail Configuration File Options

Brightmail Option (Not Case-sensitive)	Description (value of the attributes are case-sensitive)

`blSWClientDestinationLocal`	This specifies the verdict and action pair for the local domain. You would normally have two lines for this, one for spam and one for virus. The value is of the form `verdict|action`, For example,
	`blSWClientDestinationLocal=spam|spambox`
	`blSWClientDestinationLocal=virus|`
	The default Brightmail interpretation for the "null" action, meaning nothing to the right of the |, is to discard the message. So the example above discards the message if it has a verdict of `virus`. And if the verdict is `spam`, the above example files the message into the folder called `spambox`. If the message is not spam or virus, then the verdicts do not match, and the mail is delivered normally based on what's set in the `blSWClientDestinationDefault` setting above.
	When using a separate Brightmail server or servers from the MTA, you can customize the actions taken by each MTA by using the `Brightmail_verdict_n`/`Brightmail_action_n`/`Brightmail_null_action`/`Brightmail_string_action` MTA options to override the actions and verdicts returned by the Brightmail server. In this example, you can use different `Brightmail_null_action` on the MTA to override the Virus action (which would be to discard it) or to use `Brightmail_verdict_0=spambox`, and `Brightmail_action_0=data:,require "fileinto";fileinto "Junk";` to file into a folder called `Junk` instead of `spambox`.

`blSWClientDesintationForeign`	Same format and interpretation as `blSWClientDestinationLocal` above, except this applies to users in the domain which are NOT local.
`blSWUseClientOptin`	Always set this to TRUE when used with Sun Java System Messaging Server.
`blswcServerAddress`	Is of the form `ip:port[,ip:port,...]` to specify one or more Brightmail server's IP address and port numbers

Using SpamAssassin

This section consists of the following subsections:

• "SpamAssassin Options" on page 454

SpamAssassin Overview

Messaging Server supports the use of SpamAssassin, a freeware mail filter used to identify spam. SpamAssassin consists of a library written in Perl and a set of applications and utilities that can be used to integrate SpamAssassin into messaging systems.

SpamAssassin calculates a score for every message. Scores are calculated by performing a series of tests on message header and body information. Each test either succeeds or fails, and a verdict of true (spam) or false (not spam) is rendered. Scores are real numbers that can be positive or negative. Scores that exceed a specified threshold, typically 5.0, are considered to be spam. An example of a SpamAssassin result string would be as follows:

```
True ; 18.3 / 5.0
```

`True` indicates the message is spam. `18.3` is the SpamAssassin score. `5.0` is the threshold.

SpamAssassin is highly configurable. Tests may be added or removed at any time, and the scores of existing tests can be adjusted. This is all done through various configuration files. Further information on SpamAssassin can be found on the SpamAssassin web site.

The same mechanism used for calling out to the Brightmail spam and virus scanning library can be used to connect to the SpamAssassin `spamd` server. The module provided in Sun Java System Messaging Server is called `libspamass.so`.

NOTE In previous versions of Messaging Server, option names and channel keywords contained the word `brightmail`, for example, `sourcebrightmail` or `brightmail_library`. The word `brightmail` is now replaced with the more general `spamfilter`, although the old names are still valid for backwards compatibility.

SpamAssassin/Messaging Server Theory of Operations

`spamd` is the daemon version of SpamAssassin and can be invoked from the MTA. `spamd` listens on a socket for requests and spawns a child process to test the message. The child process dies after processing the message and sending back a result. In theory, the forking should be an efficient process, because the code itself is shared among the children processes.

The client portion, `spamc` from the SpamAssassin installation, is not used. Instead, its function is done by a shared library called `libspamass.so`, which is part of the Messaging Server. `libspamass.so` is loaded the same way that the Brightmail SDK is loaded.

From the MTA's point of view, you can almost transparently switch between SpamAssassin and Brightmail for spam filtering. It's not completely transparent, because they do not have same functions. For example, Brightmail can also filter for viruses, but SpamAssassin is only used to filter for spam. The result, or *verdict*, returned by the two software packages are also different. SpamAssassin provides a score, while Brightmail provides just the verdict name, so the configuration would also have some differences.

When using SpamAssassin integrated with the MTA, only a score and verdict is returned from SpamAssassin. The message itself is not modified by SpamAssassin. That is, options such as adding headers and modifying subject lines must be done by Sieve scripts.

SpamAssassin Requirements and Usage Considerations

- SpamAssassin is free. Go to http://www.spamassassin.org for software and documentation.
- SpamAssassin can be tuned and configured to provide very accurate detection of spam. The tuning of SpamAssassin is up to you and the SpamAssassin community. Messaging Server does not provide or enhance what SpamAssassin can do.
- While no specific numbers are available, SpamAssassin seems to reduce throughput more than Brightmail.
- SpamAssassin integrated with the MTA can be enabled for a user, for a domain, or for a channel.
- SpamAssassin can be configured to use other online databases such as Vipul's Razor or Distributed checksum clearinghouse (DCC).
- Messaging Server does not supply an Secure Socket Layer (SSL) version of `libspamass.so`, however, it is possible to build SpamAssassin to use openSSL.
- Perl 5.6 or later is required.

Where Should You Run SpamAssassin?

SpamAssassin can run on a separate system of its own, on the same system as the Messaging Server in a single system deployment, or on the same system as the MTA in a two-tier deployment. If Local Mail Transfer Protocol (LMTP) is used between the MTA and the message store, the filtering must be invoked from the MTA. It cannot be invoked from the message store. When SMTP is used between the MTA and the message store, it can be invoked from either one, and it can run on either system or a separate third system.

If you want to use a farm of servers running SpamAssassin, you would have to use a load balancer in front of them. The MTA is configured with only one address for the SpamAssassin server.

Configuring SpamAssassin

This section describes how to configure SpamAssassin on Messaging Server. It assumes that Perl 5.6 or later and SpamAssassin are installed. Configuring SpamAssassin involves two things:

1. "Specifying Which Messages Will Be Filtered by SpamAssassin" on page 441.

   Messages can be filtered for an entire system, for a channel, for a user, or for a domain.

2. "Specifying What to Do with Spam Messages" on page 443.

   You may want to return spam messages, delete them, file them in a separate folder, or tag them with a new header or `Subject` line.

Specifying Which Messages Will Be Filtered by SpamAssassin

The first part of configuring SpamAssassin is to specify which messages to filter. Filtering can be specified as follows:

- "To Filter All Messages to the Local Message Store" on page 442
- "To Filter All Messages Coming from the Internet" on page 442
- "To Filter All Messages Going Out to the Internet" on page 442
- "To Filter Messages to a Specific Message Store" on page 442
- "To Filter to a Specific User or Domain" on page 443

To Filter All Messages to the Local Message Store

You may wish to configure your system so that all mail delivered locally be filtered by SpamAssassin. To filter incoming messages to the local message store add `destinationspamfilteroptin spam` to the `ims-ms` channel. For example:

```
ims-ms defragment subdirs 20 backoff “pt5m” “pt10” “pt30” “pt1h” \
“pt2h” “pt4h” maxjobs 1 pool IMS_POOL fileinto $U+$S@$D filter \
ssrd:$A ims-ms-daemon destinationspamfilteroptin spam
ims-ms-daemon
```

To Filter All Messages Coming from the Internet

To filter all messages coming into your system from the internet, add the `sourcespamfilteroptin` keyword to the `tcp-local` channel. For example:

```
tcp_local smtp mx single_sys remotehost inner switchchannel \
identnonelimited subdirs 20 maxjobs 7 pool SMTP_POOL maytlsserver \
maysaslserver saslswitchchannel tcp_auth sourcespamfilteroptin spam
tcp-daemon
```

To Filter All Messages Going Out to the Internet

To filter all messages from your system going out to the internet, add the `destinationspamfilteroptin` keyword to the `tcp_local` channel definition on the outgoing MTA. For example:

```
tcp_local smtp mx single_sys remotehost inner switchchannel \
identnonelimited subdirs 20 maxjobs 7 pool SMTP_POOL maytlsserver \
maysaslserver saslswitchchannel tcp_auth \
destinationspamfilteroptin spam
tcp-daemon
```

To Filter Messages to a Specific Message Store

To configure your system so that all mail coming into a backend message store host is filtered for spam, do the following:

1. Add a rewrite rule in the `imta.cnf` file for all inbound SMTP servers that will send messages to a specific backend message store host. Example:

   ```
   msg_store1.siroe.com   $U@msg_store1.siroe.com
   ```

2. Add a channel corresponding to that rewrite rule with the `destinationspamfilteroptin` keyword. Example:

   ```
   tcp_msg_store1 subdirs 20 backoff “pt5m” “pt10” “pt30” “pt1h” \
   “pt2h” “pt4h” maxjobs 1 pool IMS_POOL fileinto $U+$S@$D \
   destinationspamfilteroptin spam
   msg_store1.siroe.com
   ```

To Filter to a Specific User or Domain

To filter messages for a user or domain, add the following attribute-value pair to the user and domain LDAP entry:

```
MailAntiUBEService: spam
```

Also specify the appropriate `LDAP_optin` value in the `option.dat` file. For user-level filtering, add the following:

```
LDAP_optin=mailantiUBEservice
```

For domain-level filtering add the following:

```
LDAP_domain_attr_optin=mailAntiUBEService
```

When activating SpamAssassin in this manner, there is no need to put `destinationspamfilter` on the channel definition. The attribute name is case-insensitive.

NOTE `MailAntiUBEService` requires any string value to enable filtering. The value `spam` is used for consistency and clarity. Some deployments may require that all entries have the same set of attributes and a value for each attribute. In this type of deployment, all entries have `spam` filtering enabled. You can, however, set the MTA option `SPAMFILTER_NULL_OPTIN` to a string which when set as the value of `MailAntiUBEService`, will disable filtering.

Specifying What to Do with Spam Messages

The second part of configuring SpamAssassin is to decide what to do with spam messages. Spam can be discarded, rejected, filed, or tagged with a header. This is controlled by a Sieve script specified by the MTA options `spamfilter_null_action` and `spamfilter_string_action`. In addition, the SpamAssassin options `mode`, `verdict`, and `field` can be used to specify folder names, header tags and so on. This is described in detail in the following sections.

NOTE Sieve is a computer language that can be used to create filters for electronic mail. The examples shown in this section can be used to create your own custom Sieve scripts. Refer to the Sieve specifications that can be found on-line for complete syntax information. A good start is http://www.cyrusoft.com/sieve/

SpamAssassin Configuration Examples

This section describes some common SpamAssassin configuration examples:

NOTE These examples use a number of options and keywords. Refer to "MTA Channel Keywords for Spam Filters" on page 422 and "MTA Spam Filter Options (options.dat)" on page 426 for details.

To File Spam to a Separate Folder

This example tests messages arriving at the local message store and files spam into a folder called `spam`. Note that the first three steps can be done in any order.

1. Specify the messages to be filtered.

   To filter all messages coming into the local message store, change the `imta.cnf` file by adding the `destinationspamfilteroptin spam` keywords on the `ims-ms` channel:

```
!
! ims-ms
ims-ms defragment subdirs 20 notices 1 7 14 21 28 backoff "pt5m" "pt10m"
"pt30m" "pt1h"  "pt2h" "pt4h" maxjobs 4 pool IMS_POOL fileinto
$U+$S@$D destinationspamfilteroptin spam
ims-ms-daemon
```

2. Create the SpamAssassin configuration file.

   The name and location of this file is specified in Step 3. A good name is `spamassassin.opt`. This file contains the following lines:

```
host=127.0.0.1
port=2000
mode=0
verdict=spam
debug=1
```

   `host` and `port` specify the name of the system where `spamd` is running and the port on which `spamd` listens for incoming requests. `mode=0` specifies that a string, specified by `verdict` (in this case the word `spam`), will be returned if the message is perceived as spam. `debug=1` turns on debugging in the SpamAssassin library.

3. Add the following lines to the `option.dat` file:

```
! for Spamassassin
spamfilter_config_file=/opt/SUNWmsgsr/config/spamassassin.opt
spamfilter_library=/opt/SUNWmsgsr/lib/libspamass.so
spamfilter_optional=1
spamfilter_string_action=data:,require "fileinto"; fileinto "$U;
```

   `spamfilter_config_files` specifies the SpamAssassin configuration file.

   `spamfilter_library` specifies the SpamAssassin shared library.

   `spamfilter_optional=1` specifies that the MTA continue operation if there is a failure by `spamd`.

   `spamfilter_string_action` specifies the Sieve action to take for a spam messages.

   Note that in this example, `spamfilter_string_action` is not necessary because the default value already is `data:,require "fileinto"; fileinto "$U;`. This line specifies that spam messages are sent to a folder. The name of the folder is the spam verdict value returned by SpamAssassin. The value returned by SpamAssassin is specified by the `verdict` option in `spamassassin.opt`. (See Step 2.) In this case, the folder name is `spam`.

4. Recompile the configuration and restart the server. Only the MTA needs to be restarted. You do not need to execute `stop-msg`.

   ```
   # imsimta cnbuild
   # imsimta restart
   ```

5. Start the `spamd` daemon. This is normally done with a command of the general form:

   ```
   spamd -d
   ```

   `spamd` defaults to only accepting connections from the local system. If SpamAssassin and Messaging Server are running on different systems, this syntax is required:

   `spamd -d -i <listen_ip_address> -A <allowed_hosts>`

   where *listen_ip_address* is the address on which to listen and *allowed_hosts* is a list of authorized hosts or networks (using IP addresses) which can connect to this `spamd` instance.

NOTE 0.0.0.0 can be used with `-i <listen_ip_address>` to have `spamd` listen on all addresses. Listening on all addresses is preferable since it avoids having to change command scripts when changing a system's IP address.

To Discard Spam

To discard spam instead of filing it into a folder, follow Step 1 and Step 2 as described in "To File Spam to a Separate Folder" on page 444, but change the `Spamfilter_string_action` option in `option.dat` as follows:

```
! for Spamassassin
spamfilter_config_file=/opt/SUNWmsgsr/config/spamassassin.opt
spamfilter_library=/opt/SUNWmsgsr/lib/libspamass.so
spamfilter_optional=1
spamfilter_string_action=data:,discard;
```

After doing this, recompile the configuration and restart the server. (See Step 4 in "To File Spam to a Separate Folder" on page 444.)

To Add a Simple Header to Spam

This example adds the header `Spam-test: spam` to each message determined to be spam. Use the same instructions described in Step 1 and Step 2 in "To File Spam to a Separate Folder" on page 444, but add the `Spamfilter_string_action` in `option.dat` as follows:

```
! for Spamassassin
spamfilter_config_file=/opt/SUNWmsgsr/config/spamassassin.opt
spamfilter_library=/opt/SUNWmsgsr/lib/libspamass.so
spamfilter_optional=1
spamfilter_string_action=data:,require ["addheader"];addheader "Spam-test: $U";
```

The first three options listed above specify the SpamAssassin configuration file (`spamfilter_config_file`), the SpamAssassin shared library (`spamfilter_library`), and to continued MTA operation if there is a failure by the shared library (`spamfilter_optional=1`). The following line specifies that a header of the form `Spam-test: spam` be added to the message:

`spamfilter_string_action=data:,require ["addheader"];addheader "Spam-test: $U";`

`data:,require ["addheader"];addheader` is Sieve sytax for adding headers to messages.

`"Spam-test: $U"` is the header string. Where "`Spam-test: `" is a literal, and `$U` specifies the string value returned by SpamAssassin. (This is specified by the `verdict` option in `spammassassin.opt`, in this case, `spam`.)

After doing this, recompile the configuration and restart the server. (See Step 4 in "To File Spam to a Separate Folder" on page 444.)

To Add a Header Containing SpamAssassin Score to Spam Messages

This example adds the header `Spam-test:` *result string* to messages determined to be spam by SpamAssassin. An example header might be:

```
Spam-test: True ; 7.3 / 5.0
```

where `Spam-test;` is a literal and everything after that is the result string. `True` means that it is spam (`false` would be not spam). `7.3` is the SpamAssassin score. `5.0` is the threshold. This result is useful for setting up a Sieve filter that can file or discard mail above or between a certain score.

In addition, setting `USE_CHECK` to `0`will return the list of SpamAssassin tests that matched along with the verdict string. See `USE_CHECK` in Table 14-4 on page 454.

1. Specify the messages to be filtered. This is described in Step 1 in “To File Spam to a Separate Folder” on page 444
2. Create the SpamAssassin configuration file.

   The name and location of this file is specified in Step 3. A good name is `spamassassin.opt`. This file will contain the following lines:

```
host=127.0.0.1
port=2000
mode=1
field=
verdict=spam
debug=1
```

   `host` and `port` specify the name of the system where `spamd` is running and the port on which `spamd` listens for incoming requests. `mode=1` specifies that the SpamAssassin result string is returned if the message is found to be spam. `field=` specifies a string prefix for the SpamAssasin result string. In this example, a prefix is not desired because we are specifying it in the Sieve script. `debug=1` turns on the debugging in the SpamAssassin library.

3. Add the following lines to the `option.dat` file:

```
!for Spamassassin
spamfilter_config_file=/opt/SUNWmsgsr/config/spamassassin.opt
spamfilter_library=/opt/SUNWmsgsr/lib/libspamass.so
spamfilter_optional=1
spamfilter_string_action=data:,require ["addheader"];addheader "Spam-test: $U";
```

   As in previous examples, the first three options specify the SpamAssassin configuration file, shared library, and to continue MTA operation if there is a failure by the shared library. The following line

   `spamfilter_string_action=data:,require ["addheader"];addheader "Spam-test: $U";`

   specifies that a header be added to spam messages. The header will have the literal prefix `Spam-text:` followed by the string returned by SpamAssassin. Because we specified `mode=1` in Step 2, the SpamAssassin result string is returned. For example: `True; 7.3/5.0`

4. Recompile the configuration and restart the server. Only the MTA needs to be restarted, so you don't need to execute `stop-msg`.

```
# imsimta cnbuild
# imsimta restart
```

5. Start the `spamd` daemon. See Step 5 in "To File Spam to a Separate Folder" on page 444.

To Add the SpamAssassin Result String to the Subject Line

By adding the SpamAssassin result string to the Subject line, users can determine whether they wish to read a message with a SpamAssassin score. For example:

```
Subject: [SPAM True ; 99.3 / 5.0] Free Money At Home with Prescription Xanirex!
```

Note that setting `USE_CHECK` to `0` will return the list of SpamAssassin tests that matched along with the verdict string (see `USE_CHECK` in Table 14-4 on page 454). This list can be very long, so we recommend setting `USE_CHECK` to `1`.

1. Specify the messages to be filtered. See Step 1 in "To File Spam to a Separate Folder" on page 444.

2. Create the SpamAssassin configuration file.

   The name and location of this file is specified in Step 3. A good name is `spamassassin.opt`. This file contains the following lines:

```
host=127.0.0.1
port=2000
mode=1
verdict=spam
debug=1
```

   `host` and `port` specify the name of the system where `spamd` is running and the port on which `spamd` listens for incoming requests. `mode=1` specifies that the SpamAssassin result string is returned if the message is found to be spam. `debug=1` turns on the debugging in the SpamAssassin library.

3. Add the following lines to the `option.dat` file:

```
!for Spamassassin
spamfilter_config_file=/opt/SUNWmsgsr/config/spamassassin.opt
spamfilter_library=/opt/SUNWmsgsr/lib/libspamass.so
spamfilter_optional=1
spamfilter_string_action=data:,addtag “[SPAM detected: $U]”;
```

As in previous examples, the first three options specify the SpamAssassin configuration file, shared library, and to continue MTA operation if there is a failure by the shared library. The following line

`spamfilter_string_action=data:,addtag “[SPAM detected $U]”;`

specifies that a tag be added to the `Subject:` line. It has the literal prefix `SPAM detected` followed by the `field` string (default: `Spam-Test`) followed by “[*result string*]” returned by SpamAssassin. Because you specified `mode=1` in Step 2, the SpamAssassin result string is returned. Thus, a subject line looks something like this:

```
Subject: [SPAM detected Spam-Test: True ; 11.3 / 5.0] Make Money at
Home!
```

You can also use `addheader` and `addtag` together:

```
spamfilter_string_action=data:,require ["addheader"];addtag "[SPAM
detected $U]";addheader "Spamscore: $U";
```

to get a message like this:

```
Subject: [SPAM detected Spam-Test: True ; 12.3 / 5.0] Vigara Now!
Spamscore: Spam-Test: True ; 12.3 / 5.0
```

Set `field=` in `spamassassin.opt` to remove the default value of Spam-Test. A cleaner message is returned:

```
Subject: [SPAM True ; 91.3 / 5.0] Viagra again!
Spamscore: True ; 91.3 / 5.0
```

4. Recompile the configuration and restart the server. Only the MTA needs to be restarted, so you don’t need to do `stop-msg`.

```
# imsimta cnbuild
# imsimta restart
```

5. Start the spamd daemon. See Step 5 in “To File Spam to a Separate Folder” on page 444.

Support for Sieve Extensions spamtest and spamadjust

Messaging Server provides support for the spamtest and spamadjust which can be used by SpamAssassin. spamtest is described in ftp://ftp.isi.edu/in-notes/rfc3685.txt. spamadjust is a non-standard action. These extensions give administrators the ability to set a different threshold values as well as the ability to set up white lists that override SpamAssassin verdicts. The two can even be combined to have different thresholds depending on who sent a particular message.

spamtest can be used to compare SpamAssassin scores to specific values using the Sieve [RELATIONAL] extension with the "i;ascii-numeric" comparator. The SpamAssassin score is typically a real number, but spamtest forces the score into an integer value between 0 an 10 by first rounding the score to the nearest integer. Values below 0 are forced up to 0 while values above 10 are forced down to 10. Finally, the text string maintained by Messaging Server is appended to produce the test string the spamtest test sees.

spamadjust is used to adjust the current spam score. This action takes a single string argument which is scanned for a real numeric value. This value is used to adjust the current spam score. The entire string is also appended to the current score text string as well. In the example shown below, the string would be “undisclosed recipients.” Multiple spamadjust actions are allowed; each one is added to the current score. Again, the score value always starts at 0. Signed numeric values are allowed, making it possible to lower the current score as well as raise it. There is no require clause for spamadjust; the spamtest extension should be listed instead.

For example, a possible use of the spamadjust with a SpamAssassin MODE setting of 2 might be:

```
SPAMFILTER_STRING_ACTION=data:,require ["spamtest"];spamadjust "$U";
```

A system-level Sieve filter could then modify the SpamAssassin score by checking for a particular type of header and, if found, adding 5 to the SpamAssassin score:

```
spamfilter_string_action=require "spamtest";
if header :contains ["to", "cc", "bcc", "resent-to", "resent-cc", "resent-bcc"]
                    ["<undisclosed recipients>", "undisclosed.recipients"]
{spamadjust "+5 undisclosed recipients";}
```

Finally, a user-level Sieve script could test the resulting value, discard messages certain to be spam, file messages likely to be spam, and allow messages from addresses in the local domain to pass through:

```
spamfilter_string_action=require ["spamtest", "relational",  \
"comparator-i;ascii-numeric", "fileinto"];                   \
if anyof (address :matches "from" ["*@siroe.com",            \
                                   "*@*.siroe.com"])         \
    {keep;}                                                  \
elsif spamtest :value "ge" :comparator "i;ascii-numeric" "8" \
    {discard;}                                               \
elsif spamtest :value "ge" :comparator "i;ascii-numeric" "5" \
    {fileinfo "spam-likely";}                                \
else                                                         \
   {keep;}                                                   \
```

Testing SpamAssassin

To test SpamAssassin, first set `debug=1` in the `spamassassion.opt` file. You do not have to turn on the channel-specific `master_debug` or `slave_debug` in the `imta.cnf`. Then send a test message to a test user. The *msg_svr_base*`/data/tcp_local_slave.log*` file should have lines similar to these:

```
15:15:45.44: SpamAssassin callout debugging enabled; config
/opt/SUNWmsgsr/config/spamassassin.opt
     15:15:45.44: IP address 127.0.0.1 specified
     15:15:45.44: Port 2000 selected
     15:15:45.44: Mode 0 selected
     15:15:45.44: Field "Spam-Test: " selected
     15:15:45.44: Verdict "spam" selected
     15:15:45.44: Using CHECK rather than SYMBOLS
     15:15:45.44: Initializing SpamAssassin message context
     ...
     15:15:51.42: Creating socket to connect to SpamAssassin
     15:15:51.42: Binding SpamAssassin socket
     15:15:51.42: Connecting to SpamAssassin
     15:15:51.42: Sending SpamAssassin announcement
     15:15:51.42: Sending SpamAssassin the message
     15:15:51.42: Performing SpamAssassin half close
     15:15:51.42: Reading SpamAssassin status
     15:15:51.67: Status line: SPAMD/1.1 0 EX_OK
     15:15:51.67: Reading SpamAssassin result
     15:15:51.67: Result line: Spam: False ; 1.3 / 5.0
     15:15:51.67: Verdict line: Spam-Test: False ; 1.3 / 5.0
     15:15:51.67: Closing connection to SpamAssassin
     15:15:51.73: Freeing SpamAssassin message context
```

If your log file does not contain lines similar to these, or if spamd is not running, the following error message will be returned in your SMTP dialog after the last "." is sent to the SMTP server.

```
452 4.4.5 Error writing message temporaries - Temporary scan failure:
End message status = -1
```

In addition, if spamfilter_optional=1 (highly recommended) is set in options.dat, the message is accepted, but not filtered. It is as if spam filtering was not enabled, and the following lines appear in tcp_local_slave.log*:

```
15:35:15.69: Creating socket to connect to SpamAssassin
15:35:15.69: Binding SpamAssassin socket
15:35:15.69: Connecting to SpamAssassin
15:35:15.69: Error connecting socket: Connection refused
15:35:15.72: Freeing SpamAssassin message context
```

The call out to SpamAssassin happens after the SMTP server received the entire message (that is, after the last "." is sent to the SMTP server), but before the SMTP server acknowledges to the sender that it accepted the message.

Another test is to send a sample spam message using sample-spam.txt from, for example, the Mail-SpamAssassin-2.60 directory. This message has the following special text string in it:

```
XJS*C4JDBQADN1.NSBN3*2IDNEN*GTUBE-STANDARD-ANTI-UBE-TEST-EMAIL*C.34X
```

The corresponding tcp_local_slave.log* contains something like this:

```
16:00:08.15: Creating socket to connect to SpamAssassin
16:00:08.15: Binding SpamAssassin socket
16:00:08.15: Connecting to SpamAssassin
16:00:08.15: Sending SpamAssassin announcement
16:00:08.15: Sending SpamAssassin the message
16:00:08.15: Performing SpamAssassin half close
16:00:08.15: Reading SpamAssassin status
16:00:08.43: Status line: SPAMD/1.1 0 EX_OK
16:00:08.43: Reading SpamAssassin result
16:00:08.43: Result line: Spam: True ; 1002.9 / 5.0
16:00:08.43: Verdict line: Spam-Test: True ; 1002.9 / 5.0
16:00:08.43: Closing connection to SpamAssassin
16:00:08.43: Mode 0 verdict of spam
16:00:08.43: Mode 0 verdict of spam
16:00:08.47: Freeing SpamAssassin message context
```

A corresponding entry in the `mail.log_current` file would look as follows. Note the `+spam` part of the destination address, which means the message is filed in the folder called `spam`:

```
15-Dec-2003 15:32:17.44 tcp_intranet ims-ms E 1 morchia@siroe.com rfc822;morchia
morchia+spam@ims-ms-daemon
15-Dec-2003 15:32:18.53 ims-ms D 1 morchia@siroe.com rfc822;morchia
morchia+spam@ims-ms-daemon
```

SpamAssassin Options

This section contains the SpamAssassin option table.

Table 14-4 SpamAssassin Options (`spamassassin.opt`)

Options	Description	Default
`debug`	0 or 1. Specifies whether to turn on debugging in the `libspamass.so`. Debugging of `spamd` itself is controlled by the command line invoking `spamd`.	0

`field`	A string specifying the prefix for the SpamAssasin result string. SpamAssassin results string look like this:
	`Spam-Test: False ; 0.0 / 5.0`
	`Spam-Test: True ; 27.7 / 5.0`
	The FIELD option provides the means for changing the "`Spam-Test: `" part of the result. Note that the ": " will also be removed if an empty `field` value is specified.
	If `USE_CHECK` is set to 0, then the result string will look similar to this:
	`Spam-test: False ; 0.3 / 4.5 ; HTML_MESSAGE,NO_REAL_NAME`
	`Spam-test: True ; 8.8 / 4.5 ; NIGERIAN_BODY, NO_REAL_NAME,PLING_PLING,RCVD_IN_SBL,SUBJ_ALL_CAPS`
		"Spam-test"

`host`	The name of the system where `spamd` is running	localhost
`port`	Port number where `spamd` listens for incoming requests.	783

Table 14-4 SpamAssassin Options (`spamassassin.opt`)

Options	Description	Default
`mode`	Controls what SpamAssassin verdict information is returned after processing a message. Three modes are available: 0 - Returns a *verdict string* (which is specified by the `verdict` option described below), if the message is spam. Returns a *SpamAssassin default verdict string* if it is not spam. (A default verdict always means to take no action and deliver as normal.) Returns a *null verdict* if the `verdict` option (defined below) is empty or unspecified. (The MTA option `spamfilter_null_action` specifies what to do if a null verdict is returned.) 1 - Returns the SpamAssassin *result string* if the message is found to be spam. A SpamAssassin result string looks something like this: `True; 6.5 / 7.3` 2- Same as mode 1 except that the SpamAssassin result string is returned regardless of whether the message is spam.	0
`USE_CHECK`	Set to 1, the `spamd CHECK` command is used to return the SpamAssassin score. Setting to 0 enables use of the `SYMBOLS` command which returns a score and a list of the SpamAssassin tests that matched. The system may hang or have other problems with this option in pre-2.55 versions of SpamAssassin. See `field` above.	
`verdict`	Specifies the verdict string used for MODE 0.	“”

Chapter 15

LMTP Delivery

The Sun™ ONE Messaging Server's MTA can use LMTP (Local Mail Transfer Protocol, defined in RFC 2033) for delivery to the message store in situations where a multi-tier messaging server deployment is used. In these scenarios, where you are using inbound relays and back end message stores, the relays become responsible for address expansion and delivery methods such as autoreply and forwarding and also for mailing list expansion. Delivery to the back end stores historically has been over SMTP which requires the back end system to look up the recipient addresses in the LDAP directory again, thereby engaging the full machinery of the MTA. For speed and efficiency, the MTA can use LMTP rather than SMTP to deliver messages to the back end store. The Sun Java System Messaging Server's LMTP server is not intended as a general purpose LMTP server, but rather as a private protocol between the relays and the back end message stores. For simplicity of discussion, examples involving two-tier deployments are used.

NOTE By design, LMTP is intended for use in multi-tier deployments. It is not possible to use LMTP with single-system deployments.

This chapter consists of the following sections:

LMTP Delivery Features

The MTA's LMTP server is more efficient for delivering to the back end message store because it:

- Reduces the load on the back end stores.

  Because relays are horizontally scalable and back end stores are not, it is good practice to push as much processing to the relays as possible

- Reduces the load on the LDAP servers.

  The LDAP infrastructure is often a limiting factor in large messaging deployments.

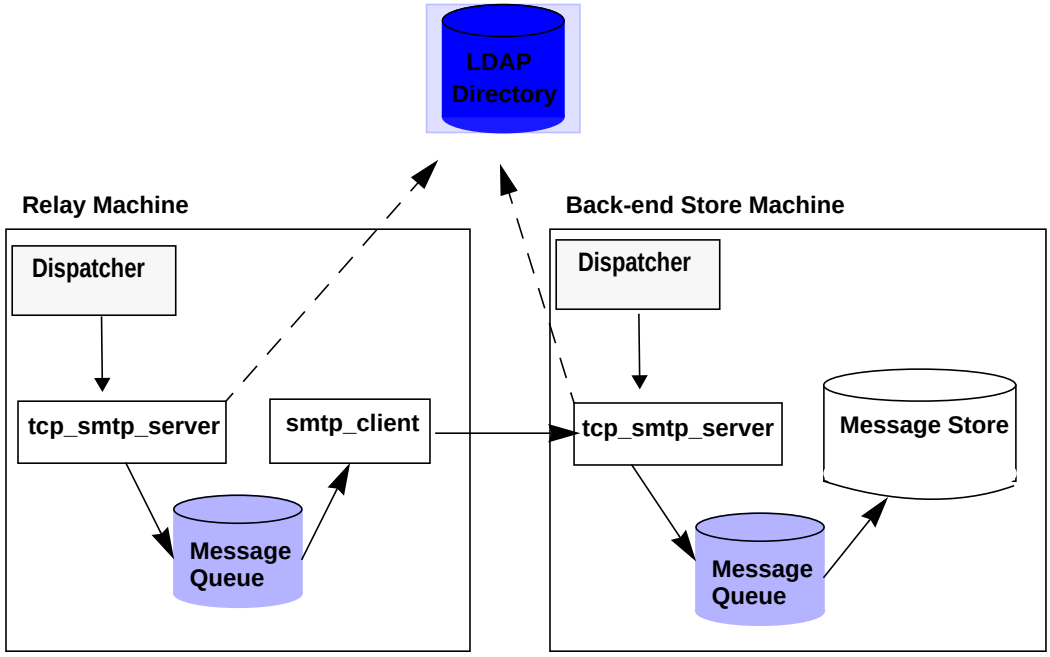
- Reduces the number of message queues.

  Having queues on both the relay and the back end store makes finding a lost message that much harder for the administrator of a messaging deployment.

Messaging Processing in a Two-Tier Deployment Without LMTP

Figure 15-1 presents in pictorial form the following discussion of message processing in a two-tier deployment scenario without LMTP.

Figure 15-1 Two Tier Deployment Without LMTP



Without LMTP, in a two level deployment with relays in front of the store systems, the processing of an inbound message begins with a connection on the SMTP port picked up by the dispatcher on the relay machine and handed off to a `tcp_smtp_server` process. This process does a number of things with the inbound message including:

- Looking up the user in the directory
- Determining if the user is within a domain hosted by this email deployment
- Determining if the user is a valid user in the domain
- Rewriting the envelope address as `@mailhost:user@domain`
- Enqueuing the message for delivery to the mailhost

The `smtp_client` process then picks up the mail message from the queue and sends it to the mailhost. On the mailhost, some very similar processing takes place. A connection on the SMTP port is picked up by the dispatcher and handed off to a `tcp_smtp_server` process. This process does a number of things to the message, including:

- Looking up the user in the directory
- Determining if the user is within a domain hosted by this email deployment
- Determining if the user is a valid user in the domain
- Rewriting the envelope address to direct the message to the `ims_ms` channel
- Enqueuing the message for delivery to the store

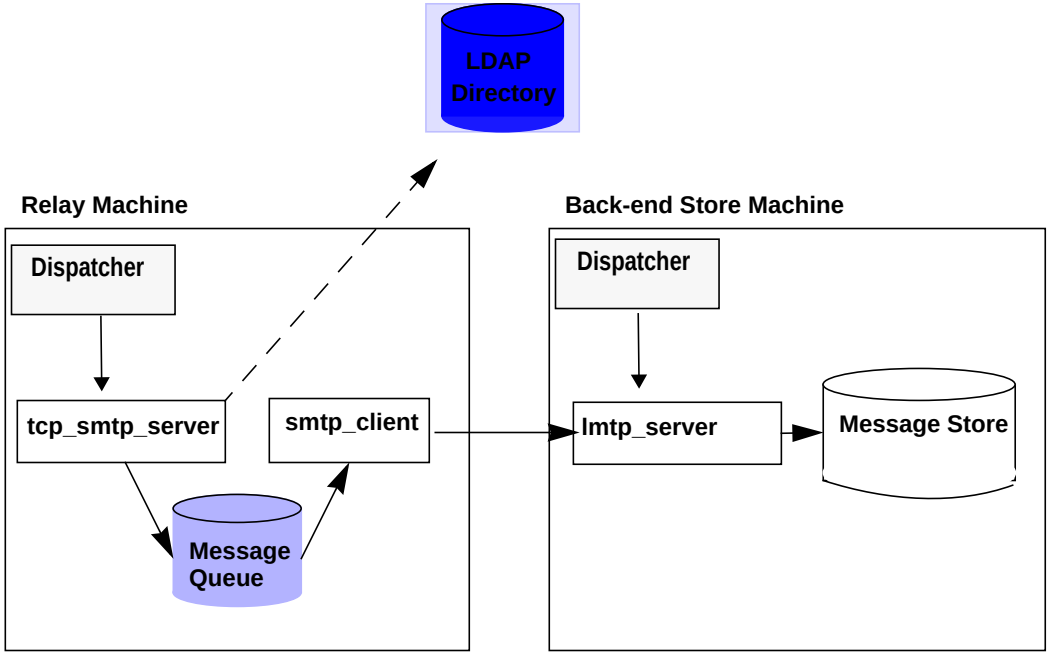
Then the `ims_ms` process picks up the mail message and attempts to deliver it to the store.

In this scenario, the enqueuing processing is performed twice, and the MTAs each perform an LDAP lookup.

Messaging Processing in a Two-Tier Deployment With LMTP

Figure 15-2 presents in pictorial form the following discussion of message processing in a two-tier deployment scenario with LMTP.

Figure 15-2 Two Tier Deployment With LMTP



With LMTP in place, a connection on the SMTP port of the relay machine is picked up by the dispatcher and handed off to a `tcp_smtp_server` process. This process does a number of things with the inbound message including:

- Looking up the user in the directory
- Determining if the user is within a domain hosted by this email deployment
- Determining if the user is a valid user in the domain
- Determining which back end message store machine hosts the mailbox for the user
- Rewriting the address as `@mailhost:uid@domain.LMTP` or `@mailhost:uid@domain.LMTPNATIVE`
- Enqueuing the message for delivery to the mailhost

Addresses of the format `user@domain.LMTP` and `user@domain.LMTPNATIVE` are routed to the message store system via a `tcp_lmtp` channel or a `tcp_lmtpnative` channel respectively. These channels communicate with the back end message store using LMTP instead of SMTP. On the store machine, a connection to the

LMTP port is received by the dispatcher and handed off to the `lmtp_server` process. The LMTP server then inserts the message into the user's mailbox or into the UNIX native mailbox. If message delivery is successful, the message is dequeued on the relay machine. If unsuccessful, the message remains on the relay machine. Note that the LMTP process on the message store does not engage any MTA machinery for processing addresses or messages.

LMTP Overview

For the most part, the MTA itself can be basically absent from the back end server. The only necessary MTA components are:

- the dispatcher
- `libimta`
- the LMTP server
- the `imta.cnf` file
- the `mappings` file
- the `imta.tailor` file

While the dispatcher requires MTA configuration files, these files can be very short. The dispatcher must run on the back end server so that it can start the LMTP servers which run under it. Because the dispatcher and the LMTP server use various functions of `libimta`, this needs to be present on the back end server as well.

The LMTP server does not perform any of the usual MTA enqueuing or dequeuing functions, header processing, or address translations. The relay system performs all the manipulation of the content of the messages and addresses which then presents to the LMTP server the message in exactly the form to be delivered to the message store and with the delivery address already in the form required by the store. Additional recipient information that is usually available as a message is delivered to the store, such as the user’s quota, is presented along with the recipient address as LMTP parameters. Should a delivery attempt fail, the message is left enqueued in the LMTP queue on the relay system.

Configuring LMTP Delivery

Configuring the LMTP delivery mechanism requires configuration on both the relay machines and on the back end stores. On the relays, the `DELIVERY_OPTIONS` MTA option (in `option.dat`) has to be changed so that messages being delivered to the stores are passed to the LMTP channel. The back end store must be configured with the dispatcher, but does not need the job controller. The dispatcher must be configured to run the LMTP server.

In a typical multi-tier deployment, users are provisioned on different backend message store machines. One or more of these backend machines may not have LMTP turned on and therefore the front-end relays need to be aware of which store machines are LMTP aware. This is achieved by using the General Database facility to explicitly name those message stores which are configured to accept LMTP delivery.

To Configure the Inbound MTA Relays with LMTP

To configure inbound MTA relays to use LMTP, do the following:

1. Activate text databases by adding the following line to `option.dat`.

   ```
   USE_TEXT_DATABASES=1
   ```

   In this step, the use of a flat text file for the General Database is enabled in the MTA. Note that if you already use the general database, you may want to skip this step.

2. Create or modify the General Database text file.

   ```
   # cd /opt/SUNWmsgsr/config/
   # vi general.txt
   LMTP_CS|msg-store.siroe.com           lmtpcs-daemon
   LMTP_CS|name-1-lmtp-store.siroe.com   lmtpcs-daemon
   LMTP_CS|name-2-lmtp-store.siroe.com   lmtpcs-daemon
   ..
   ..
   LMTP_CN|Zmar.Talek@siroe.com          lmtpcs-daemon
   ..
   LMTP_CN|Fred.Bloggs@siroe.com         lmtpcs-daemon
   # chown mailsrv general.txt
   ```

As you can see, there two types of entries, one for handling user specific deliveries to the `lmtpnative` and one for handling store wide settings for delivery via the `tcp_lmtpcs` channel.

3. Add an LMTP rewrite rule to `imta.cnf file`:

```
# cd /opt/SUNWmsgsr/config/
# cp imta.cnf imta.cnf.orig
# vi imta.cnf
!
! pipe
.pipe-daemon $U%$H.pipe-daemon@pipe-daemon
!
! tcp_local
! Rules for top level internet domains
<IMTA_TABLE:internet.rules
!
! Do mapping lookup for internal IP addresses
[] $E$R${INTERNAL_IP,$L}$U%[$L]@tcp_intranet-daemon
!
! Do general.txt lookup for lmtp hosts
.domain-name.com $S$U%$H$D@$(LMTP_CN|$U@$H$D)
.domain-name.com $S$U%$H$D@$(LMTP_CS|$H$D)
!
! tcp_intranet
! Do mapping lookup for internal IP addresses
[] $E$R${INTERNAL_IP,$L}$U%[$L]@tcp_intranet-daemon
.domain-name.com $U%$H.domain-name.com@tcp_intranet-daemon
```

In this step, a pair of rewrite rules do a tagged probe of the General Database to see if the source route portion of the address matches any entries for doing LMTP delivery. In the `general.txt` file, which was created in step 2, there are tagged entries to designate delivery to the backend message store via the appropriate channel. Here, the `$S` in the rewrite rule means that it will only apply when the address contains a source route. If there is a match to an entry in the General Database, the rewrite rule succeeds and the message is sent to the source route backend host via the `tcp_lmtpX` channel which does delivery via LMTP.

If a match is not found, the rewrite process will continue until a match is found in some other rewrite rule. In most cases, if a match is not found via a probe of the General Database, then message is routed via the `tcp_intranet` channel which does delivery via SMTP.

4. Add new channel blocks to `imta.cnf`

   You must also include channel definitions for the `lmtp` and `lmtpn` channels in the channel definition section of the `imta.cnf` file. For example:

```
! tcp_lmtpcs (LMTP client - store)
tcp_lmtpcs defragment lmtp port 225 nomx single_sys subdirs 20
maxjobs 7 pool SMTP_POOL dequeue_removeroute
lmtpcs-daemon

!
! tcp_lmtpcn (LMTP client - native)
tcp_lmtpcn defragment lmtp port 226 nomx single_sys subdirs 20
maxjobs 7 pool SMTP_POOL dequeue_removeroute
lmtpcn-daemon
```

5. Commit your configuration changes.

```
# cd /opt/SUNWmsgsr/bin
# ./imsimta refresh
Compiled configuration done
Killing Dispatcher : 23021
Dispatcher startup requested
Job Controller shutdown requested
Job Controller startup requested
```

NOTE Be sure to use the `lmtp` channel keyword on LMTP channels. Do not use both `smtp` and `lmtp` channel keywords on LMTP channels. Note also that by default, the LMTP channel definitions are commented out. You must uncomment them if you want LMTP to work.

Configuring the Back End Stores with LMTP and No MTA

The back end stores need no MTA if they are receiving messages over LMTP. This means that they have no job controller, and none of the address rewriting machinery associated with the MTA. They do however still require a dispatcher and a simple MTA configuration. In particular they need a `dispatcher.cnf` file and a `mappings` file which comprise the only significant part of the MTA configuration.

The dispatcher.cnf file must contain the following:

```
! rfc 2033 LMTP server - store
!
[SERVICE=LMTPSS]
PORT=225
IMAGE=IMTA_BIN:tcp_lmtp_server
LOGFILE=IMTA_LOG:tcp_lmtpss_server.log
PARAMETER=CHANNEL=tcp_lmtpss
STACKSIZE=2048000
! Uncomment the following line and set INTERFACE_ADDRESS to an appropriate
! host IP (dotted quad) if the dispatcher needs to listen on a specific
! interface (e.g. in a HA environment).
!INTERFACE_ADDRESS=
!
! rfc 2033 LMTP server - native
!
[SERVICE=LMTPSN]
PORT=226
IMAGE=IMTA_BIN:tcp_lmtpn_server
LOGFILE=IMTA_LOG:tcp_lmtpsn_server.log
PARAMETER=CHANNEL=tcp_lmtpsn
STACKSIZE=2048000
! Uncomment the following line and set INTERFACE_ADDRESS to an appropriate
! host IP (dotted quad) if the dispatcher needs to listen on a specific
! interface (e.g. in a HA environment).
!INTERFACE_ADDRESS=
!
```

Note that by default, the LMTP services in the dispatcher.cnf file are commented out. You must uncomment them to get LMTP to work.

The normal dispatcher options of MAX_CONNS, MAX_PROCS, MAX_LIFE_CONNS, and MAX_LIFE_TIME can also be set, but need to be set appropriately for your hardware.

The PORT_ACCESS mapping is important. The LMTP implementation for the back end servers is intended as a private protocol between Sun Java System Messaging Server relays and back end stores. You must use the PORT_ACCESS mapping to make sure that only such relays can connect to these services. Your mapping file should look like this:

```
PORT_ACCESS

  TCP|*|225|1.2.3.4|* $Y
  TCP|*|226|1.2.3.4|* $Y
  TCP|*|225|1.2.3.5|* $Y
  TCP|*|226|1.2.3.5|* $Y
  TCP|*|*|*|*   $N500$ Do$ not$ connect$ to$ this$ machine
```

You should replace the sample IP addresses specified in the PORT_ACCESS mapping table here with the IP addresses of your relays on the network that connect to the back end stores.

There has to be an imta.cnf file, but it is there merely to make the configuration complete. A minimal imta.cnf file consists of the following channel definitions:

```
! tcp_lmtpss (LMTP server - store)
tcp_lmtpss lmtp subdirs 20
tcp_lmtpss-daemon

!
! tcp_lmtpsn (LMTP server - native)
tcp_lmtpsn lmtp subdirs 20
tcp_lmtpsn-daemon
```

Note that by default, the LMTP channel definitions are commented out. You must uncomment them if you want LMTP to work.

Configuring Relays for Sending Messages Via LMTP to Back End Systems with Message Stores and Full MTAs

There are situations where you might want the back end stores to have the full capabilities of the MTA but still to have the load savings of using LMTP. For example, you might want program delivery on the back end store. In this case the relays should be configured as described above for relays using LMTP to deliver to the back end store.

Configuring LMTP on Back End Message Store Systems Having Full MTAs

The only changes from the configuration of a back end store messaging system to one with LMTP direct delivery to the store are that the following lines need to be added to the end of the dispatcher.cnf file:

```
! rfc 2033 LMTP server - store
!
[SERVICE=LMTPSS]
PORT=225
IMAGE=IMTA_BIN:tcp_lmtp_server
LOGFILE=IMTA_LOG:tcp_lmtpss_server.log
PARAMETER=CHANNEL=tcp_lmtpss
STACKSIZE=2048000
! Uncomment the following line and set INTERFACE_ADDRESS to an appropriate
! host IP (dotted quad) if the dispatcher needs to listen on a specific
! interface (e.g. in a HA environment).
!INTERFACE_ADDRESS=
!
! rfc 2033 LMTP server - native
!
[SERVICE=LMTPSN]
PORT=226
IMAGE=IMTA_BIN:tcp_lmtpn_server
LOGFILE=IMTA_LOG:tcp_lmtpsn_server.log
PARAMETER=CHANNEL=tcp_lmtpsn
STACKSIZE=2048000
! Uncomment the following line and set INTERFACE_ADDRESS to an appropriate
! host IP (dotted quad) if the dispatcher needs to listen on a specific
! interface (e.g. in a HA environment).
!INTERFACE_ADDRESS=
!
```

Note that by default, the LMTP services in the dispatcher.cnf file are commented out. You must uncomment them to get LMTP to work. Also, the LMTP port numbers are just examples, and can be anything you choose.

This is the same as the whole dispatcher.cnf file described above for when the back end store is configured only for LMTP. The mappings file also requires the PORT_ACCESS mappings as described for LMTP only back end stores.

LMTP Protocol as Implemented

This section provides a sample LMTP dialogue with an explanation of what is seen in that dialogue. The LMTP client on the relay uses standard LMTP protocol to talk to the LMTP server on the back end store. However the protocol is used in specific ways. For example:

```
---> LHLO
<--- 250 OK
```

No action is taken on the `LHLO` message. The reply is always `250 OK`.

```
---> MAIL FROM: address size=messageSizeInBytes
<--- 250 OK
```

No checks or conversions are made on the originator address. The `size=` parameter gives a size in bytes for the message that is to be delivered. This is the size of the message exactly as it appears in the protocol. It is not necessarily the exact size of the message, but the actual message size will not exceed this size. The LMTP server allocates a memory buffer of this size to receive the message.

```
---> RCPT TO: uid+folder@domain xquota=size,number xdflg=xxx
<--- 250 OK
```

No checks are made on the recipient addresses at the time they are received, but a list of recipients is built for later use. Note that the `@domain` part of the address is omitted for `uids` in the primary domain, and that the `+folder` part is optional. This is the same address format used by the message store channel in the MTA.

The xquota= parameter gives the user's message quotas which consist of the maximum total size and the maximum number of messages. The MTA provides this information which it retrieves while performing an LDAP lookup on the user to do the address translation. This information is used to keep the quota information in the message store synchronized with the directory. Getting the quota information does not result in an additional performance hit.

The xdflg= parameter specifies a number which is interpreted as a bit field. These bits control how the message is delivered. For example, the bit whose value is 2, if set, guarantees delivery of the message even if the user is over quota.

This interaction may be repeated many times, once for each recipient.

```
--->DATA
---> <the message text>
--->.
```

The LMTP client then sends the entire message, dot-stuffed, just as SMTP does. The message finishes with a dot (.) alone on a line. If the message size is exceeded the LMTP server sends:

```
<--- 500 message too big
```

and ends the connection.

Assuming that the message is received correctly, the LMTP server then sends back to the LMTP client the status for each recipient given in the RCPT TO: lines. For instance, if the message is delivered successfully, the response is:

```
<--- 250 2.5.0 address OK
```

where address is exactly as it appeared on the RCPT TO: line.

The conversation can either repeat with another MAIL FROM: line or end with the following interaction:

```
---> quit
<--- 221 OK
```

Table 15-1 shows the possible status codes for each recipient. This three-column table shows the short code in the first column, its long-code equivalent in the second column and the status text in the third column. 2.x.x status codes are success codes, 4.x.x codes are retryable errors, 5.x.x codes are non retryable errors.

Table 15-1 LMTP Status Codes for Recipients

Short Code	Long Code	Status Text
250	2.5.0	OK
420	4.2.0	Mailbox Locked
422	4.2.2	Quota Exceeded
420	4.2.0	Mailbox Bad Formats
420	4.2.0	Mailbox not supported
430	4.3.0	IMAP IOERROR
522	5.2.2	Persistent Quota Exceeded
523	5.2.3	Message too large
511	5.1.1	mailbox nonexistent
560	5.6.0	message contains null
560	5.6.0	message contains nl
560	5.6.0	message has bad header
560	5.6.0	message has no blank line

Otherwise, there are changes to the delivery options for mailbox, native (and, therefore, UNIX), and file. The object of these rules is to generate addresses that will cause the messages to be sent through the appropriate LMTP channel to the back end servers. The addresses generated are source routed addresses of the form:

```
@sourceroute:localpart@domain
```

Chapter 16

Automatic Message Reply

For automatically generated responses to email (autoreply), specifically vacation messages, the MTA uses Message Disposition Notifications (*MDNs*) and the Sieve scripting language. MDNs are email messages sent by the MTA to a sender and / or postmaster reporting on a message's delivery disposition. MDNs are also known as read receipts, acknowledgements, receipt notifications, or delivery receipts. The Sieve is a simple scripting language used to create mail filters.

This section describes the vacation autoreply mechanism. In most cases, modification to the default configuration is unnecessary, but there is a case where you may wish to configure your system such that vacation processing is done on MTA relay machines rather than on back-end message stores.

This chapter consists of the following sections:

Vacation Autoreply Overview

Vacation Sieve scripts are generated automatically from the various LDAP vacation attributes (see "Vacation Autoreply Attributes" on page 477). They can also be specified explicitly for additional flexibility. The underlying mechanism for tracking vacations is a set of files, one per intended recipient, that keep track of when replies were sent to the various senders.

By default, the MTA evaluates vacation on the back-end store systems. However, since MTA relays do not do as much work as back-end stores, for performance reasons, you can have the MTA evaluate vacation on the mail relay machines instead of on the back-end store. Use of this feature, however, can result in vacation responses being sent out more often than intended because different relays handle different messages. If you do not want vacation messages to be sent out more often than you intend, you may share the tracking of files between the relays. If this is also unacceptable to you, you can always have vacation evaluated on the back-end store systems.

Configuring Autoreply

Delivery addresses are generated through a set of patterns. The patterns used depend upon the values defined for the `mailDeliveryOption` attribute. A delivery address is generated for each valid `mailDeliveryOption`. The patterns are defined by the MTA option `DELIVERY_OPTIONS`, which are defined in the `option.dat` file. The default autoreply rule in `DELIVERY_OPTIONS` in the `option.dat` file is:

```
*^!autoreply=$M+$D@bitbucket
```

The MTA notes the "^" on the autoreply `DELIVERY_OPTION` MTA option. This causes the MTA to check the vacation dates. If the current date is within the vacation dates, processing continues and the MTA notes the "!" on the autoreply `DELIVERY_OPTION`. The MTA then creates a vacation Sieve script based on the various autoreply LDAP attributes on the user's entry. The autoreply rule can have the prefix characters '!', '#', '^', and '*'.

You could have the '!' flag on the mailbox delivery option. This would enable the generation of the vacation script unconditionally. It makes sense, however, to have the autoreply machinery enabled by a separate delivery option so that it can be further gated by the '^' flag. Checking the dates at this stage is more efficient than engaging the Sieve logic.

Table 16-1 shows the prefix characters used for the autoreply rule in the first column and their definitions in the second column.

Table 16-1 Prefix Characters for the Autoreply Rule in DELIVERY_OPTIONS

Prefix Character	Definition
!	Enable the generation of the autoreply Sieve script.
#	Allows the processing to take place on relays.
^	Option is only evaluated if the vacation dates indicate that it should be evaluated.

Table 16-1 Prefix Characters for the Autoreply Rule in DELIVERY_OPTIONS

Prefix Character	Definition
*	Rule is only applicable to users.

The autoreply rule itself specifies an address destined for the bitbucket channel. The mail is considered delivered by this method once the autoreply is generated, but the MTA machinery requires a delivery address. Anything delivered to the bitbucket channel is discarded.

Configuring Autoreply on the Back-end Store System

The default autoreply rule in `DELIVERY_OPTIONS` causes the autoreply to take place on the mail server that serves the user. If you want vacation messages to be evaluated on the back-end store system, you do not have to configure anything. This is the default behavior.

Configuring Autoreply on the Relay

If you want to evaluate vacation on the relay rather than on the back-end store system to enhance performance, edit the `option.dat` file and prepend the character # to the autoreply rule in `DELIVERY_OPTIONS`. For example:

1. Use an editor to open the `option.dat` file.
2. Add or change the `DELIVERY_OPTIONS` option so the autoreply rule now looks like:

   ```
   #*^!autoreply=$M+$D@bitbucket
   ```

   The default `DELIVERY_OPTIONS` option looks like:

   ```
   DELIVERY_OPTIONS=*mailbox=$M%$\$2I$_+$2S@ims-ms-daemon, \
    &members=*, \
    *native=$M@native-daemon, \
    /hold=@hold-daemon:$A, \
    *unix=$M@native-daemon, \
    &file=+$F@native-daemon, \
   ```

```
&@members_offline=* \
,program=$M%$P@pipe-daemon, \
#forward=**, \
*^!autoreply=$M+$D@bitbucket \
```

This allows the processing to take place on relays. If you have the MTA perform autoreplies on the relays, then either each relay can keep track independently of whether a particular correspondent has sent an away message recently, or this information can be shared between the relays. The former case is simpler, especially if having away messages sent out too many times does not matter. If you want strict application of the away message frequency rules, then the information must be shared between relays. To share the information among the relays, the files should be NFS mounted

The location of these files is controlled by the option `VACATION_TEMPLATE`. This option (in `option.dat`) should be set to /<path>/%A where <path> is the path to a directory that is shared between the various relay machines. The template needs to be a `file:URL` and you use $U to substitute the name of the user. The default setting is:

```
VACATION_TEMPLATE=file:///opt/SUNWmsgsr/data/vacation/$3I/$1A/$2A/$U.vac
```

Vacation Autoreply Theory of Operation

The vacation action, when invoked, works as follows:

1. The Sun Java System Messaging Server checks to make sure that the vacation action was performed by a user level rather than a system level Sieve script. An error results if vacation is used in a system level script.
2. The “no vacation notice” internal MTA flag is checked. If it is set, processing terminates and no vacation notice is sent.
3. The return address for the message is now checked. If it is blank processing terminates and no vacation notice is sent.
4. The MTA checks to see if the user's address or any of the additional addresses specified in the `:addresses` tagged argument appear in a `To:`, `Cc:`, `Resent-to:`, or `Resent-cc:` header field for the current message. Processing terminates and no vacation notice is sent if none of the addresses is found in any of the header fields.

5. The Messaging Server constructs a hash of the :subject argument and the reason string. This string, along with the return address of the current message, is checked against a per-user record of previous vacation responses. Processing terminates and no response is sent if a response has already been sent within the time allowed by the :days argument.
6. The Messaging Server constructs a vacation notice from the :subject argument, reason string, and :mime argument. Two basic forms for this response message are possible:
   - A message disposition notification of the form specified in RFC 2298, with the first part containing the reason text.
   - A single part text reply. (This form is only used to support the "reply" autoreply mode attribute setting.)

Note that mailautoreplymode is automatically set to reply when vacation messages are configured through Messenger Express.

The "no vacation notice" MTA flag is clear by default. It can be set by a system level Sieve script through the use of the nonstandard novacation action. The novacation Sieve action is only allowed in a system level Sieve script. It will generate an error if it is used in a user level script. You can use this action to implement site-wide restrictions on vacation replies such as blocking replies to addresses containing the substring "MAILER-DAEMON".

Per-user per-response information is stored in a set of flat text files, one per local user. The location and naming scheme for these files is specified by the setting of the VACATION_TEMPLATE MTA option. This option should be set to a file: URL.

Maintenance of these files is automatic and controlled by the VACATION_CLEANUP integer MTA option setting. Each time one of these files is opened, the value of the current time in seconds modulo this value is computed. If the result is zero the file is scanned and all expired entries are removed. The default value for the option is 200, which means that there is 1-in-200 chance that a cleanup pass will be performed.

The machinery used to read and write these flat text files is designed in such a way that it should be able to operate correctly over NFS. This allows multiple MTAs to share a single set of files on a common file system.

Vacation Autoreply Attributes

The set of user directory attributes that the vacation action uses are:

- vacationStartDate

  Vacation start date and time. The value is in the format YYYYMMDDHHMMSSZ. This value is normalized to GMT. An autoreply should only be generated if the current time is after the time specified by this attribute. No start date is enforced if this attribute is missing. The MTA can be instructed to look at a different attribute for this information by setting the LDAP_START_DATE MTA option to a different attribute name.

  This attribute will be read and checked by the code that generated the Sieve script. Vacation processing will be aborted if the current date is before the vacation start date. This attribute cannot be handled by the script itself because at present Sieve lacks date/time testing and comparison facilities.

- vacationEndDate

  Vacation end date and time. The value is in the format YYYYMMDDHHMMSSZ. This value is normalized to GMT. An autoreply should only be generated if the current time is before the time specified by this attribute. No end date is enforced if this attribute is missing. The MTA can be instructed to look at a different attribute for this information by setting the LDAP_END_DATE MTA option to a different attribute name.

  This attribute will be read and checked by the code that generated the Sieve script. Vacation processing will be aborted if the current date is after the vacation end date. This attribute cannot be handled in the script itself because at present Sieve lacks date/time testing and comparison facilities.

- mailAutoReplyMode

  Specifies autoreply mode for the user mail account. Valid values of this attribute are:

  - echo - Create a multipart that echoes the original message text in addition to the added mailAutoReplyText or mailAutoReplyTextInternal text.
  - reply - Send a single part reply as specified by either mailAutoReplyText or mailAutoReplyTextInternal to the original sender.

  These modes will appear in the Sieve script as nonstandard :echo and :reply arguments to the vacation action. echo will produce a “processed” message disposition notification (MDN) that contains the original message as returned content. reply will produce a pure reply containing only the reply text. An illegal value will not manifest as any argument to the vacation action and this will produce an MDN containing only the headers of the original message. Note also that selecting an autoreply mode of echo causes an automatic reply to be sent to every message regardless of how recently a previous reply was sent.

The MTA can be instructed to use a different attribute for this information by setting the LDAP_AUTOREPLY_MODE MTA option to a different attribute name.

- mailAutoReplySubject

  Specifies the contents of the subject field to use in the autoreply response. This must be a UTF-8 string. This value gets passed as the :subject argument to the vacation action. The MTA can be instructed to use a different attribute for this information by setting the LDAP_AUTOREPLY_SUBJECT MTA option to a different attribute name.

  Note that because Sieve currently lacks the ability to perform certain substitutions at present use of $SUBJECT to insert the original message into the header cannot be implemented.

- mailAutoReplyText

  Autoreply text sent to all senders except users in the recipient's domain. If not specified, external users receive no vacation message. The MTA can be instructed to use a different attribute for this information by setting the LDAP_AUTOREPLY_TEXT MTA option to a different attribute name.

- mailAutoReplyTextInternal

  Auto-reply text sent to senders from the recipients domain. If not specified, then internal uses get the mail autoreply text message. The MTA can be instructed to use a different attribute for this information by setting the LDAP_AUTOREPLY_TEXT_INT MTA option to a different attribute name.

  The MTA will pass either the mailAutoReplyText or mailAutoReplyTextInternal attribute value as the reason string to the vacation action.

- mailAutoReplyTimeOut

  Duration, in hours, for successive autoreply responses to any given mail sender. Used only when mailAutoReplyMode=reply. If value is 0 then a response is sent back every time a message is received. This value will be converted to the nonstandard :hours argument to the vacation action. (Normally the Sieve vacation action only supports the :days argument for this purpose and does not allow a value of 0.)

  If this attribute doesn't appear on a user entry, a default time-out will be obtained from the AUTOREPLY_TIMEOUT_DEFAULT MTA option. The MTA can be instructed to use a different attribute for this information by setting the LDAP_AUTOREPLY_TIMEOUT MTA option.

Chapter 17

Mail Filtering and Access Control

This chapter discusses how to filter mail based on its source (sender, IP address and so on) or header strings. Two mail filtering mechanisms are used, controlling access to the MTA using mapping tables and Sieve server-side rules (SSR).

Limiting access to the MTA using the mapping tables allows messages to be filtered based on From: and To: addresses, IP address, port numbers and source or destination channels. Mapping tables allow SMTP relaying to be enabled or disabled. Sieve is a mail filtering script that allows messages to be filtered based on strings found in headers. (It doesn't work on the message body.)

If envelope-level controls are desired, use mapping tables to filter mail. If header-based controls are desired, use Sieve server-side rules.

This chapter is divided into two parts:

PART 1. MAPPING TABLES. Allows the administrator to control access to MTA services by configuring certain mapping tables. The administrator can control who can or cannot send mail, receive mail through Messaging Server.

PART 1. MAPPING TABLES. Allows users and administrators to filter messages and specify actions on those filtered messages based on a string found in the message header. Uses the Sieve filter language and can filter at the channel, MTA or user level.

PART 1. MAPPING TABLES

Part 1 contains the following sections:

Controlling Access with Mapping Tables

You can control access to your mail services by configuring certain mapping tables. These mapping tables (Table 17-1) 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 "Mappings File" on page 225.

NOTE Whenever the `mappings` file is modified, you must recompile the configuration (see the `imsimta refresh` command in the *Sun Java System Messaging Server Administration Reference*).

Table 17-1 lists the mapping tables described in this section.

Table 17-1 Access Control Mapping Tables

Mapping Table	Description
`SEND_ACCESS` (See page 483.)	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 483.)	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 485.)	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 485.)	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`.

Table 17-1 Access Control Mapping Tables

Mapping Table	Description
FROM_ACCESS (See page 486.)	Used to filter mail based on envelope From addresses. Use this table if the To address is irrelevant.
PORT_ACCESS (See page 489.)	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 505.

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 the example below 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`).

Code Example 17-1 SEND_ACCESS Mapping Table

```
SEND_ACCESS

  *|postmaster@sesta.com|*|*    $Y
  *|*|*|postmaster@sesta.com    $Y
  l|*@sesta.com|tcp_*|*         $NInternet$ postings$ are$ not$  \
    permitted
```

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.

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 the example below.

Code Example 17-2 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 505.

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.

Code Example 17-3 FROM_ACCESS Mapping Table

```
FROM_ACCESS

  *|SMTP|*|tcp_auth|*|        $Y
  *|SMTP|*|tcp_auth|*|*       $Y$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_auth|*|     $Y
  *|SMTP|*|tcp_auth|*|*    $Y$K$3
```

However, the real purpose of `FROM_ACCESS` is to permit more complex and subtle alterations, as shown in the example below. 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.

```
FROM_ACCESS

! If no authenticated address is available, do nothing
  *|SMTP|*|tcp_auth|*|              $Y
! If authenticated address matches envelope From:, do nothing
  *|SMTP|*|tcp_auth|*|$2*           $Y
! If authenticated address matches envelope From: sans
! subaddress, do nothing
  *|SMTP|*|tcp_auth|*+*@*|$2*@$4*    $Y
! Fall though to...
! ...authenticated address present, but didn't match, so force
! Sender: header
  *|SMTP|*|tcp_auth|*|*             $Y$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.

NOTE The MMP does not make use of the PORT_ACCESS mapping table. If you wish to reject SMTP connections from certain IP addresses and you are using the MMP, you must use the TCPAccess option. See "To Configure Mail Access with MMP" on page 165. If you want to control SMTP connections using mapping tables, use the INTERNAL_IP mapping table (see "Allowing SMTP Relaying for External Sites" on page 495).

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 17-2 lists the available flags.

Table 17-2 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.

Table 17-2 PORT_ACCESS Mapping Flags

Flag	Description
$> string	Send string to syslog (UNIX) or to the event log (NT) if access is rejected.
$N string	Reject access with the optional error text string
$F string	Synonym for $N string; that is, reject access with the optional error text string
$T text	If bit 1 of the LOG_CONNECTION MTA option is set and the $N flag is set so that the connection is rejected, then $T causes a "T" entry to be written to the connection log; the optional text (which must appear subsequent to two vertical bar characters) may be included in the connection log entry.

+To use multiple flags with arguments, separate the arguments with the vertical bar character, |, placing the arguments in the order listed in this table.

For example, the following mapping will only accept SMTP connections (to port 25, the normal SMTP port) from a single network, except for a particular host singled out for rejection without explanatory text:

```
PORT_ACCESS

  TCP|*|25|192.123.10.70|*  $N500
  TCP|*|25|192.123.10.*|*   $Y
  TCP|*|25|*|*              $N500$ Bzzzt$ thank$ you$ for$  \
     playing.
```

Note that you will need to restart the Dispatcher after making any changes to the PORT_ACCESS mapping table so that the Dispatcher will see the changes. (If you are using a compiled MTA configuration, you will first need to recompile your configuration to get the change incorporated into the compiled configuration.)

The PORT_ACCESS mapping table is specifically intended for performing IP-based rejections. For more general control at the email address level, the SEND_ACCESS or MAIL_ACCESS mapping table, might be more appropriate.

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:

`$[`*msg_svr_base*`/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|*|* \
$C$[msg_svr_base/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.

`$[`*msg_svr_base*`/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. The example below shows a sample `SEND_ACCESS` mapping table and the resulting 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 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 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 *msg_svr_base*`/config/mappings`

For instance, on an 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 `imsimta test -match` utility can be useful for checking whether an IP address matches a particular $(.../...) test condition. The `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 `imsimta restart` command (if you are not running with a compiled configuration) or the `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 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” on page 499
- “Prevent Mail Relay” on page 500

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 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.

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|*|*@*|*|* \
$[msg_svr_base/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...
```

In this case, messages are rejected at the SMTP level, that is, the messages are rejected during the SMTP dialogue and thus never sent to the MTA. 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|*|* \
$C$[msg_svr_base/lib/dns_verify.so, \
dns_verify_domain_port,$1,rbl.maps.vix.com.]EXTERNAL$E


ORIG_MAIL_ACCESS

  TCP|*|25|*|*|SMTP|*|tcp_local|*@*|*|* \
$C$[msg_svr_base/lib/dns_verify.so, \
dns_verify_domain,$1,rbl.maps.vix.com.]$E
```

Support for DNS-based Databases

The dns_verify program 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 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|*|*  \
$C$[<msg_svr_base/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 `ORIG_SEND_ACCESS` mapping table shown below.

```
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 the example below, which shows sample source text file for a general database and a sample ORIG_SEND_ACCESS mapping table. To compile this source file into database format run the imsimta crdb command:

% imsimta crdb *input-file-spec output-database-spec*

for detailed information about the imsimta crdb utility, refer to the *Sun Java System Messaging Server Administration Reference.*

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 17-3 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 17-2).

Table 17-3 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 17-3 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

Mailbox filters are a set of specified actions to apply to a mail message depending upon a string found in the message header. 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 and are sometimes called SIEVE filters.

This part contains the following sections:

Sieve Filtering Overview

A SIEVE filter consists of one or more conditional actions to apply to a mail message depending upon a string found in the message header. 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. Users can create per-user filters for their own mailboxes using Messenger Express. Specific instructions for this are described in the Messenger Express on-line help.

The server applies filters in the following priority:

1. User-level 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. Per-user filters are set

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 Messenger Express 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 User-level Filters

Per-user mail filters apply to messages destined for a particular user’s mailbox. Per-user mail filters can only be created using Messenger Express.

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:

   `../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 17-4 on page 509.

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 17-4 `filter` Channel Keyword *URL-pattern* Substitution Tags (Case-insensitive)

Tag	Meaning
*	Perform group expansion.
**	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

Table 17-4 `filter` Channel Keyword *URL-pattern* Substitution Tags (Case-insensitive)

Tag	Meaning
$M	Insert the UID, stripped of any hosted domain
$P	Insert the method name (`mailProgramDeliveryInfo` attribute)
$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:

   `../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-level Filters

The following information will help you if you are having problems with the user filters on your system.

The MTA's SSR database is automatically updated with information about the users' filters. Short filters are stored in the database. For long filters, the database stores an LDAP dn.

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
  ```

- 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 18

Managing the Message Store

This chapter describes the message store and its 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 message store partitions (see "Configuring Message Store Partitions" on page 553).

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, Messaging Server provides a set of command-line utilities in addition to the Sun Java System Console interface. Table 18-1 describes these command-line utilities. For information about using these utilities, see "Performing Message Store Maintenance Procedures" on page 556 and the *Messaging Server Reference Manual*.

Table 18-1 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.
`imsconnutil`	Monitors user access of the message store.
`imexpire`	Automatically removes messages from the message store based on administrator-specified criteria like age.
`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.

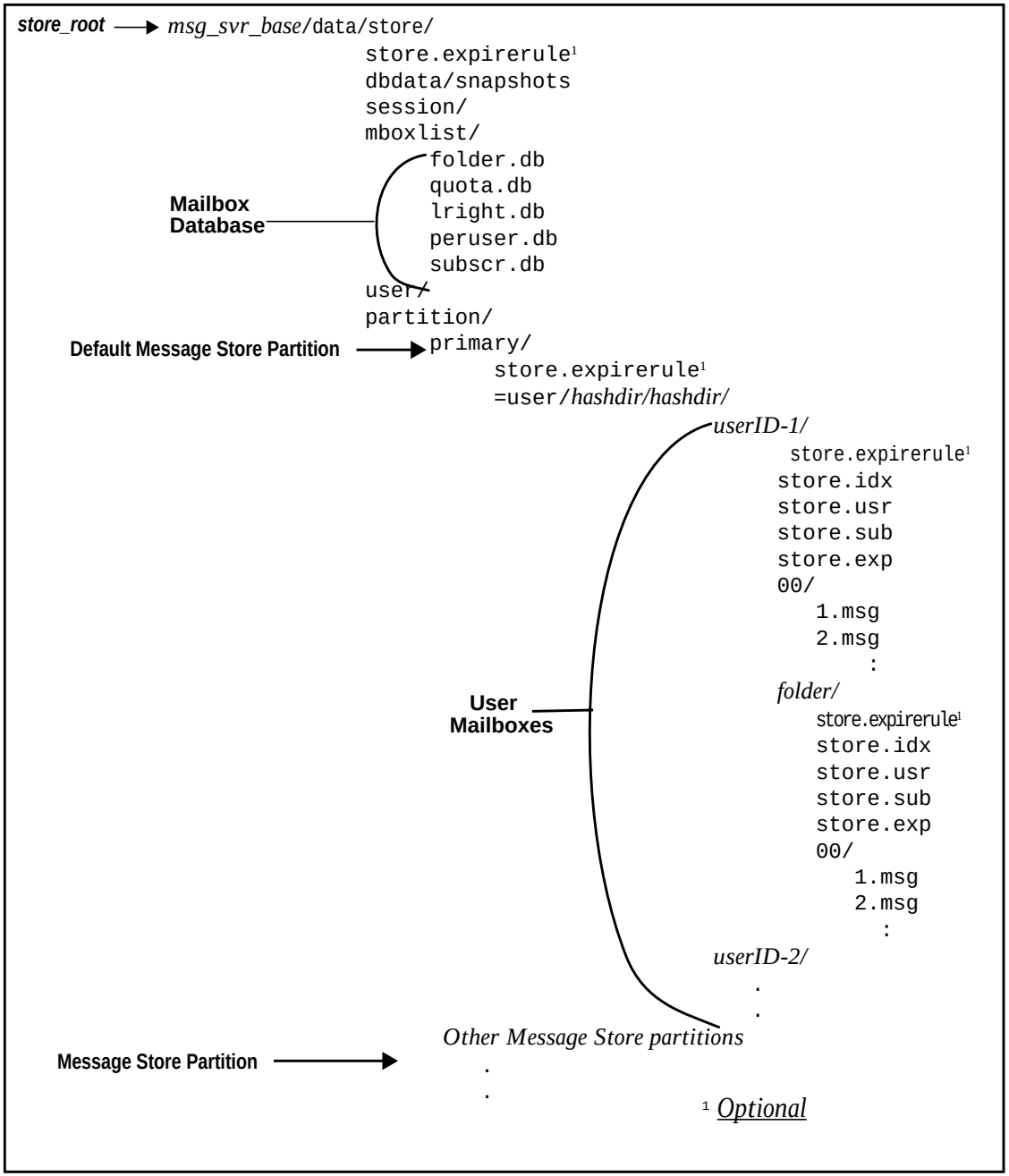
Table 18-1 Message Store Command-line Utilities

Utility	Description
`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.
`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.
`imquotacheck`	Calculates the total mailbox size for each user in the message store and compares the size with their assigned quota. Localized versions of `imquotacheck` notification incorrectly convert the % and the $ signs. To correct the encoding, replace every $ with \24 and replace every % with \25 in the message file.
`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 18-1 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 18-2.

Figure 18-1 Message Store Directory Layout



The message store consists of a number of mailbox databases and the user mailboxes. The mailbox databases consists of information about users, mailboxes, partitions, quotas and other message store related data. The user mailboxes contain the user's messages and folders. Mailboxes are stored in a *message store partition*, an area on a *disk* partition specifically devoted to storing the message store. See "Configuring Message Store Partitions" on page 553 for details. Message store partitions are not the same as disk partitions, though for ease of maintenance, we recommend having one disk partition for each message store partition.

Mailboxes such as INBOX are located in the *store_root*. For example, a sample directory path might be:

store_root`/partition/primary/=user/53/53/=mack1`

The table below describe the message store directory.

Table 18-2 Message Store Directory Description

Location	Content/Description
msg_svr_base	Default: `/opt/SUNWmsgsr` The directory on the Messaging Server machine that holds the server program, configuration, maintenance, and information files.
store_root	*msg_svr_base*`/data/store` Top-level directory of the message store. Contains the `mboxlist`, `user`, and `partition` subdirectories.
`./store.expirerule`	Contains the automatic message removal rules (expire rules). This optional file can be at different locations. See "To Set the Automatic Message Removal (Expire and Purge) Feature" on page 541.
store_root`/dbdata/snapshots`	Message store database backup snapshots.
store_root`/mboxlist/`	Contains mailbox database, database (Berkeley DB) that stores information about the mailboxes and quota information. `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 `quota.db` contains information about quotas and quota usage. There is one entry in `quota.db` per user. `lright.db` - an index for the folders by acl lookup rights. `peruser.db` contains information about per-user flags. The flags indicate whether a particular user has seen or deleted a message. `subscr.db` contains information about user subscriptions.
store_root`/session/`	Contains active message store process information.
store_root`/user/`	Not used.

Table 18-2 Message Store Directory Description

Location	Content/Description
store_root/partition/	Contains the message store partitions. A default primary partition is created. Place any other partitions you define in this directory.
store_root/partition/primary/=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*. This is the user's INBOX. For the default domain, *userid* is *uid*. For hosted domains, *userid* is *uid@domain*. Incoming messages are delivered to this mail folder.
.../*userid*/*folder*	A user-defined folder on the messaging server.
.../*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.sub	Contains information about user subscriptions.
.../*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*/*nn*/ or .../*userid/folder/nn*/	*nn* is a hash directory that contains messages in the format *message_id*.msg; *nn* can be a number from 00 to 99. *message_id* is also a number. Example: messages 1 through 99 are stored in the .../00 directory. The first message is 1.msg, the second is 2.msg, thrid 3.msg, and so on. 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 Message Store Removes Messages

Messages are removed from the message store in three stages:

1. **Delete**. A client sets a message flag to *delete*. At this point, the message is marked for removal, but client can still restore the message by removing the delete flag. If there is a second client, the deleted flag may not be recognized immediately by that second client. You can set the `configutil` parameter `local.imap.immediateflagupdate` to enable immediate flag update.

2. **Expunge**. Messages are removed from the mailbox. Technically, they are removed from the message store index file, `store.idx`. The message itself is still on disk, but once messages are expunged, clients can no longer restore them.

   Expire is a special case of expunge. Messages that conform to a set of administrator-defined removal criteria such as message size, age and so forth, are expunged. See "To Set the Automatic Message Removal (Expire and Purge) Feature" on page 541

3. **Purge**. The `stored` utility purges from the disk any messages that have been expunged at 11PM everyday by default. This can be configured with `local.schedule.purge`, which controls the message purge schedule, and `store.cleanup`, which controls the purge grace period (period of time before which the message will not be purged).

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, some administrators may have these privileges.

You can perform administrator tasks as described in the following subsections:

- To Add an Administrator
- To Modify an Administrator Entry
- To Delete an Administrator Entry

To Add an Administrator

Console To add an administrator entry at the 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.

   The tab contains a list of existing administrator IDs.
4. Click the Add button beside the Administrator UID window.
5. 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 Sun Java System Directory Server.
6. Click OK to add the administrator ID to the list displayed in the Administrator tab.
7. 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. In addition, the administrator must be a member of the Service Administrator Group (in the LDAP user entry: `memberOf: cn=Service Administrators,ou=Groups,o=usergroup`).

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:

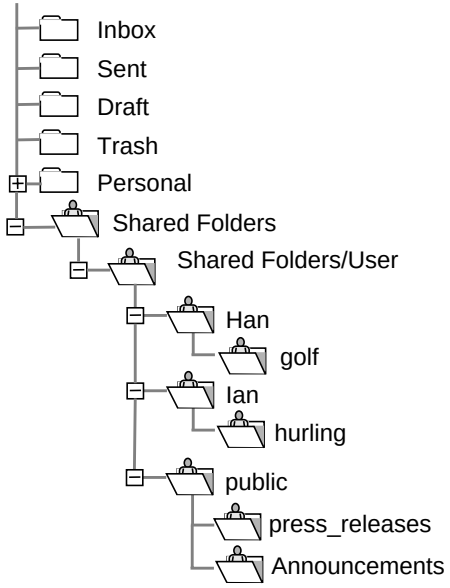
```
configutil -o store.admins -v "adminlist"
```

About Shared Folders

A *shared folder* is a folder that can be accessed and read by a group of users. In other words, access rights to shared folders are granted to multiple users. For example, a user can create a folder called `golf` and allow others to view the contents of that folder.

By default, Messaging Server creates a folder called `Shared Folders/Users` in all email accounts. Users create and access shared folders in this folder. An example of how shared folders would appear in a client is shown in Figure 18-2. This example will be described further in "To Set Up Distributed Shared Folders" on page 528.

Figure 18-2 Example of Ed's Client Shared Mail Folder List



Users can create private shared folders and provide access rights to those folders with their email client, if that client supports shared folders. These shared folders will appear in the `Shared Folders` of other users who are given access rights.

Shared folders are useful for starting, sharing, and archiving an ongoing conversation on a particular topic. For example, a group of software developers can create a shared folder for discussing development of a particular project. When a message is sent to the shared folder, everyone who is subscribed to the shared folder (subscribers can be added by individual address or by a group address) can open this mailbox and read the message.

There are two kinds of shared folders:

- **Private** – A private shared folder is a shared folder owned by a specific user. The owner of a folder grants access to others.
- **Public** – A public shared folder does not have an owner. Administrators create a public user account, which can be used to host public folders. The email address of a public folder looks like this:

  `public+`*foldername*`@`*domain*

For example, you might want a folder, such as `public+software_dev@siroe.com` for posting information about a special interest group inside the company. Interested employees would be granted access to this public folder.

Normally shared folders are only available to users on a particular message store. Messaging Server, however, allows you to create special shared folders that can be accessed across multiple message stores. These are called *distributed shared folders*. See "To Set Up Distributed Shared Folders" on page 528 for details.

Shared Folder Access Rights

Access rights are maintained in Access Control Lists (ACLs) stored in `folder.db`. Granting access rights is done by setting an ACL using either the IMAP `SETACL` command, the `-s` option with the `readership` command line utility (see "To Change a Public Folder's Access Control Rights" on page 526), or using the Messenger Express interface.

ACL Identifiers

Each ACL entry has an identifier which specifies the user or group of users for which the entry applies. Identifiers starting with a dash ("-") represent negative rights (those denied to the user or group).

`anyone` is a special identifier. The access rights for `anyone` apply to all users. Similarly, the access rights for `anyone@`*domain* apply to all users in the same domain.

Group identifiers start with `group=`.

ACL Rights Characters

Each ACL entry has a set of rights represented by a string of characters. The string of characters is defined by RFC 2086. To calculate the set of rights for a user, the server adds all the rights granted to this user and all of the groups this user belongs to, then it subtracts all of the rights denied the user and the groups this user belongs to.

The table that follows lists the characters recognized by Messaging Server and gives their names, a brief description of each, and shows the IMAP commands users with this permission are allowed to issue.

Table 18-3 ACL Rights Characters

Character	Description
l	lookup – User can see and subscribe to the shared folder. (IMAP commands allowed: LIST and LSUB)
r	read – Users can read the shared folder. (IMAP commands allowed: SELECT, CHECK, FETCH, PARTIAL, SEARCH, COPY from the folder)
s	seen – Directs the system to keep seen information across sessions. (Set IMAP STORE SEEN flag)
w	write – Users can mark as read, and delete messages. (Set IMAP STORE flags, other than SEEN and DELETED)
i	insert – Users can copy and move email from one folder to another. (IMAP commands allowed: APPEND, COPY into folder)
p	post – Users can send mail to the shared folder email address. (No IMAP command needed)
c	create – Users can create new sub-folders. (IMAP command allowed: CREATE)
d	delete – Users can delete entries from the shared folder. (IMAP commands allowed: EXPUNGE, set STORE DELETED flag)
a	administer – Users have administrative privileges. (IMAP command allowed: SETACL)

Group ACL

The identifier of an ACL entry can specify a group name. The access rights of this entry apply to all of the members of this group. The server determines group memberships by the aclGroupAddr attribute of the inetMailUser object class. A group is represented by a dynamic mailing list with a filter on the aclGroupAddr attribute. The following example shows an LDIF record that defines a group, including the aclGroupAddr attribute:

```
dn: cn=lee-staff,ou=Groups, o=sesta.com
cn: lee-staff
mailHost: mail.sesta.com
inetMailGroupStatus: active
mgrpErrorsTo: lee.jones@sesta.com
description: Dynamic Group of Lee’s staff
objectClass: top
objectClass: groupofuniquenames
objectClass: inetmailgroup
objectClass: inetmailgroupmanagement
objectClass: inetlocalmailrecipient
objectClass: groupofurls
```

```
mail: lee-staff@sesta.com
memberURL: ldap:///o=sesta.com??sub?
(&(aclGroupAddr=lee-staff@sesta.com)(objectclass=inetmailuser))
```

It is not necessary for a group to be created for the group email address to be used in the folder's ACL. In practice, it makes sense to create such a dynamic group and set the `aclGroupAddr` attribute on user entries when adding members to the group. Once such a group is created, static external members can be added by using their email addresses in the attribute `mgrpRfc822MailMember`. Members should not be added using the `uniqueMember` attribute, nor by creating additional values of the `memberURL` attribute. Doing so would cause a disconnect between what the MTA sees as a member of the mailing list and what the IMAP server sees as a group member.

When a user logs in to an IMAP server or logs in with an HTTP access service client such as Messenger Express, the server fetches the `aclGroupAddr` attribute (along with other message store related attributes) and caches the group names in memory. The server uses this information to determine the user's access rights whenever the client issues a command (such as `LIST` or `SELECT`) that requires access right verification.

Shared Folder Tasks

This section describes the shared folder administrator tasks:

To Create a Public Folder

Because public folders require access to the LDAP database as well as the `readership` command, they must be created by system administrators.

1. Add an LDAP user entry that will act as a container for all public folders, for example, one called `public`:

```
dn: cn=public,ou=people,o=sesta.com,o=ISP
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: inetUser
objectClass: ipUser
objectClass: inetMailUser
objectClass: inetLocalMailRecipient
objectClass: nsManagedPerson
objectClass: userPresenceProfile
cn: public
mail: public@sesta.com
mailDeliveryOption: mailbox
mailHost: manatee.siroe.com
uid: public
inetUserStatus: active
mailUserStatus: active
mailQuota: -1
mailMsgQuota: 100
```

2. Create folders within the public account by using the `mboxutil` command line utility. For example:

   ```
   mboxutil -c user/public/golftournament
   ```

3. Using the `readership` command line utility, set the appropriate ACL for this folder.

   In order to make this folder public, it must be assigned a group of users that can access it. This is done by setting an ACL using the `readership` command. For instructions on how to set the ACL, see "To Change a Public Folder's Access Control Rights" on page 526 that follows.

To Change a Public Folder's Access Control Rights

From time to time you might need to change the access control for a public folder, or you will need to set the access control for a public folder you have newly created.

To do this, use the `readership` command line utility. The command has the following form:

`readership -s` *foldername identifier rights_chars*

where *foldername* is the name of the public folder for which you are setting rights, *userid* is the person or group to whom you are assigning the rights, and *rights_chars* are the rights you are assigning (these are RFC 2086 compliant access rights characters). For the meaning of each character, see "ACL Rights Characters" on page 523. You can also change the access control for a public folder using the Messenger Express interface.

Examples

For example, if you wish everyone at the sesta domain to have lookup, read and email marking (but not posting) access to the public folder, `golftournament`, issue the command as follows:.

```
readership -s User/public/golftournament anyone@sesta lwr
```

To assign lookup, read, email marking and posting rights to a group, issue the command as follows:

```
readership -s User/public/golftournament group=golfinterest lwrp
```

If you want to assign administrator and posting rights for this folder to an individual, `jdoe`, issue the command as follows:

```
readership -s User/public/golftournament jdoe lwrpa
```

To deny an individual or group access to a public folder, prefix the `userid` with a dash. For example, to deny lookup, read and write rights to `jsmith`, issue the command as follows:

```
readership -s User/public/golftournament -jsmith lwr
```

To Enable or Disable Listing of Shared Folders

The server will or will not return shared folders when responding to a `LIST` command depending on the setting in the configuration option `local.store.sharedfolders`. Setting the option to `off` disables it. The setting is enabled by default (set to `on`).

`SELECT` and `LSUB` commands are not affected by this option. The `LSUB` command returns every subscribed folder, including shared folders. Users can `SELECT` the shared folders they own or are subscribed to.

To Set Up Distributed Shared Folders

Normally shared folders are only available to users on a particular message store. Messaging Server, however, allows you to create *distributed shared folders* that can be accessed across multiple message stores. That is, access rights to distributed shared folders can be granted to any users within the group of message stores. Note, however, that web mail clients (HTTP access clients like Messenger Express) do not support remote shared folders access. Users can list and subscribe to the folders, but they can't view or alter the contents.

Distributed shared folders require the following:

- The message store userids must be unique across the group of message stores.
- The directory data across the deployment must be identical.

The remote message stores (that is the message stores that do not hold the shared folder) must be configured as proxy servers by setting the configuration variables listed in Table 18-4 on page 528.

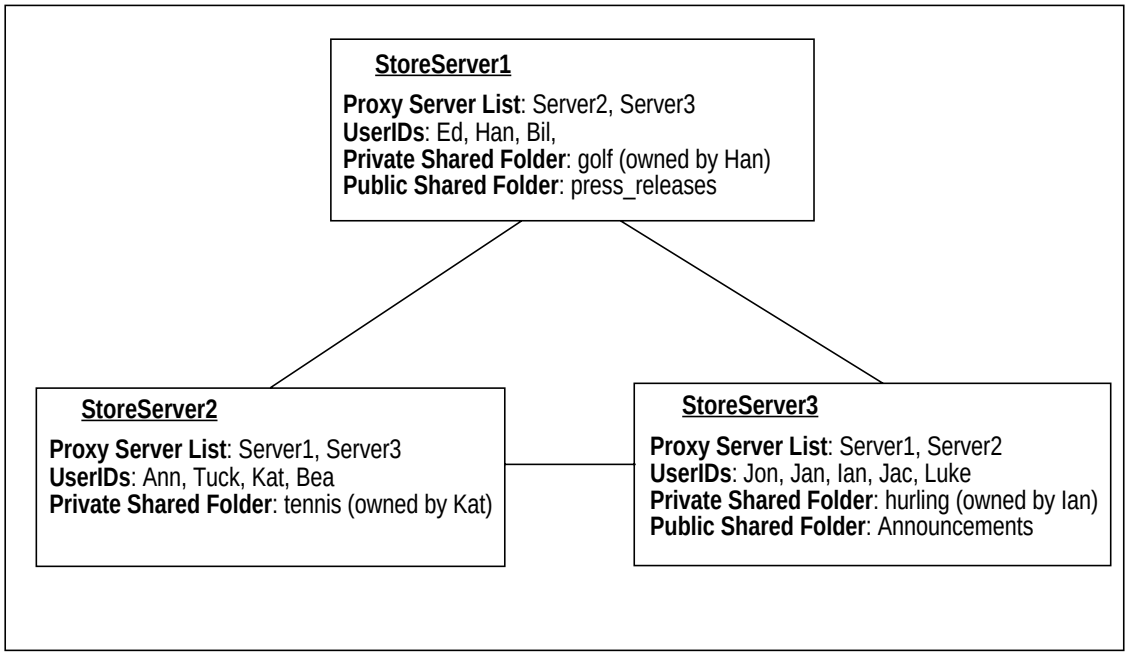
Table 18-4 Variables for Configuring Distributed Shared Folders

Name	Value	Data Format
`local.service.proxy.serverlist`	message store server list	space-separated strings
`local.service.proxy.admin`	default store admin login name	string
`local.service.proxy.adminpass`	default store admin password	string
`local.service.proxy.admin.`*`hostname`*	store admin login name for a specific host	string
`local.service.proxy.adminpass.`*`hostname`*	store admin password for a specific host	string

Setting Up Distributed Shared Folders—Example

Figure 18-3 shows a disturbed folder example of three message store servers called StoreServer1, StoreServer2, and StoreServer3.

Figure 18-3 Distributed Shared Folders—Example



These servers are connected to each other as peer proxy message stores by setting the variables shown in Table 18-4. Each server has a private shared folder—*golf* (owned by Han), *tennis* (owned by Kat), and *hurling* (owned by Luke). In addition there are two public shared folders called *press_releases* and *Announcements*. Users on any of the three servers can access any of these three shared folders. Figure 18-2 on page 522 shows Ed's shared folder list. Below is an example of the ACLs for each server in this configuration.

```
$ StoreServer1 :> readership -l
Ed: user/Han/golf
Ian: user/Han/golf
anyone: user/public/press_releases
```

```
$ StoreServer2 :> readership -l
Jan: user/Kat/tennis
Ann: user/Kat/tennis
anyone: user/public+Announcements user/public+press_releases
```

```
$ StoreServer3 :> readership -l
Tuck: user/Ian/hurling
Ed: user/Ian/hurling
Jac: user/Ian/hurling
anyone: user/public/Announcements
```

To Monitor and Maintain Shared Folder Data

The `readership` command line utility allows you to monitor and maintain shared folder data which is held in the `folder.db`, `peruser.db`, and `lright.db` files. `folder.db` has a record for each folder that holds a copy of the ACLs.The `peruser.db` has an entry per user and mailbox that lists the various flags settings and the last date the user accessed any folders. The `lright.db` has a list of all the users and the shared folders for which they have lookup rights.

The `readership` command line utility takes the following options:

Table 18-5 `readership` Options

Options	Description
`-d days`	Returns a report, per shared folder, of the number of users who have selected the folder within the specified days.
`-p months`	Removes data from the `peruser.db` for those users who have not selected their shared folders within the specified months.
`-l`	List the data in `lright.db`.
`-s` *folder_identifier_rights*	Set access rights for the specified folder. This updates the `lright.db` as well as the `folder.db`.

Using the various options, you can perform the following functions:

To Monitor Shared Folder Usage

To find out how many users are actively accessing shared folders, issue the command:

`readership -d` *days*

where *days* is the number of days to check. Note that this option returns the number of active users, not a list of the active users.

Example: To find out the number of users who have selected shared folders within the last 30 days, issue the following command:

```
readership -d 30
```

To List Users and Their Shared Folders

To list users and the shared folders to which they have access, issue the command:

```
readership -l
```

Example output:

```
$ readership -l
group=lee-staff@siroe.com: user/user2/lee-staff
richb: user/golf user/user10/Drafts user/user2/lee-staff user/user10/Trash
han1: user/public+hurling@siroe.com user/golf
gregk: user/public+hurling@siroe.com user/heaving user/tennis
```

To Remove Inactive Users

If you want to remove inactive users (those who have not accessed shared folders in a specified time period) issue the command:

`readership -p` *months*

where *months* is the number of months to check for.

Example: Remove users who have not accessed shared folders for the past six months:

```
readership -p 6
```

To Set Access Rights

You can assign access rights to a new public folder, or change access rights on a current public folder.

For an example of how to set access rights with this command, see "To Change a Public Folder's Access Control Rights" on page 526.

About Message Store Quotas

A message store quota is a way of setting a limit or *quota* for how much disk space or how many messages can be used by users or domains. This section contains information about the following:

- "User Quotas" on page 532
- "Domain Quotas" on page 533
- "Exceptions for Telephony Application Servers" on page 533

For further information, see "To Monitor Quota Limits" on page 560

User Quotas

You can specify user quotas by disk space or by number of messages. Disk space quotas specify, in bytes, the amount of disk space for 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 the total number of user messages. Message quotas allow you to limit the number of messages stored in a user's mailbox.

Quota information is stored in user LDAP attributes (Table 18-6) and `configutil` variables (Table 18-7). In addition to setting the quota itself, Messaging Server allows you to control the following features:

- **Quota notification** sends users a warning message when they have reached a *disk quota threshold*.
- **Quota enforcement** halts delivery of messages into the message store once the quota is exceeded, or it allows message delivery even if the quota is exceeded.

  If message delivery is halted, incoming messages remain in the MTA queue until one of the following occurs:

- The size or number of the user's messages no longer exceeds the quota, at which time the MTA delivers the messages.
- The undelivered message remains in the MTA queue longer than the specified grace period, at which time messages are returned to sender. (See "To Set a Grace Period" on page 540).

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.

- **Quota Default** sets default quotas for all users or different quotas for specific 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 looks at the default quota set for all users.

Domain Quotas

As for users, quotas can also be set for domains by either number of bytes or number of messages. This quota is for all the cumulative bytes or messages of all the users in a particular domain.

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 ensures the message is delivered to the store regardless of quota limits. For more information about configuring the TAS channel, see Chapter 12, "Configuring Channel Definitions".

Table 18-6 Message Store Quota Attributes (See *Sun Java System Communcations Services Schema Reference Manual* for latest and complete information.)

Attribute	Description
mailQuota	Bytes of disk space allowed for the user's mailbox. Special values: 0 - No space allowed for user's mailbox. -1 - No limit on space usage allowed. -2 - Use system default quota. (`configutil` parameter `store.defaultmailboxquota`)

Table 18-6 Message Store Quota Attributes (See *Sun Java System Communcations Services Schema Reference Manual* for latest and complete information.)

Attribute	Description
mailMsgQuota	Maximum number of messages permitted for a user. This is a cumulative count for all folders in the store. Special values: 0 - No message allowed for user's mailbox. -1 - No limit on number of messages allowed. -2 - Use system default quota. (`configutil` parameter `store.defaultmessage.quota`.)
mailUserStatus	Status of the mail user. Can be one of the following values: `active` - mail is processed as normal. Default is `active`. `inactive` - user's mail account is inactive. A transient failure is returned. `deleted` - Account marked deleted and ready for purge. Permanent failure returned. Access to mailbox blocked. `hold` - Mail sent to the hold queue and access to the mailbox is disallowed `overquota` - The MTA will not deliver mail to a mailbox with this status.
mailDomainDiskQuota	Bytes of disk space allowed for the cumulative count of all the mailboxes in a domain. A value of -1 means no limit on space usage. (Default) To enforce the domain disk quota run the command: `imquotacheck -f -d` *domain*
mailDomainMsgQuota	Maximum number of messages permitted for a domain, that is, the total count for all mailboxes in the store. A value of -1 means no limit. (Default). To enforce the domain message quota run the command: `imquotacheck -f -d` *domain*
mailDomainStatus	Status of the mail domain. Values and default are the same as mailUserStatus.

Table 18-7 Message Store `configutil` parameters (See *Sun Java System Messaging Server Administration Reference* for latest and complete information.)

Parameter	Description
store.quotaenforcement	Enable quota enforcement Default: On
store.quotanotification	Enable quota notification. Default: On
store.defaultmailboxquota	Store default quota by number of bytes. Default: -1 (unlimited)
store.defaultmessagequota	Store default quota by number of messages. Numeric. Default: -1 (unlimited)
store.quotaexceededmsg	Quota warning message. If none, notification is not sent. Default: None.
store.quotaexceededmsginterval	Interval, in days, for sending overquota notification. Default: 7
store.quotagraceperiod	Time, in hours, a mailbox has been overquota before messages to the mailbox will bounce back to the sender. Number of hours. Default: 120

Table 18-7 Message Store `configutil` parameters (See *Sun Java System Messaging Server Administration Reference* for latest and complete information.)

Parameter	Description
store.quotawarn	Quota warning threshold. Percentage of quota exceeded before clients are sent an over quota warning. Default: 90
local.store.quotaoverdraft	When ON, allow delivery of message that puts disk usage over quota. At that time, messages are deferred or bounced, the quota warning message is sent, and the quota grace period timer starts. (The default is that the quota warning messages are sent when the message store reaches the threshold.) Default: Off
local.store.overquotastatus	Enable quota enforcement before messages are enqueued in the MTA. This prevents the MTA queues from filling up. Default: off

Configuring Message Store Quotas

This section describes the following tasks:

To Specify a Default User Quota

To set a default quota that applies to users who do not have individual quotas set, perform the following steps:

Console To specify a default user quota at the Console:

1. Click the Configuration tab and select Message Store in the left pane.
2. Click the Quota tab.

3. 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 `Mbytes`.

4. To specify a message number quota, in the “Default user message quota” box, type a number.

5. Click Save.

6. Set the `Mbytes` attribute to `-1` in the user entries that use the default message store quota. See Table 18-6.

Command Line To specify a default user quota at the 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 and *number* indicates a number of bytes.

To specify a default user quota for total number of messages:

`configutil -o store.defaultmessagequota -v [ -1 |` *number* `]`

where `-1` indicates no quota and *number* indicates the number of messages.

Set the `mailQuota` attribute to `-1` in the user entries that use the default message store quota. See Table 18-6.

To Specify Individual User Quotas

Each user can have individualized quotas. To set user-specific quotas, set the `mailQuota` or `mailmsgquota` attributes in the user’s LDAP entry. (See Table 18-6.) To enforce the quota, set the `configutil store.quotaenforcement` to `on`.

To Specify Domain Quotas

You can set disk space quotas or message quotas for particular domains. These quotas are for the cumulative bytes or messages of all users in a particular domain. To set domain quotas, set the `mailDomainDiskQuota` or `mailDomainMsgQuota` attributes in the user’s LDAP entry (see Table 18-6) and run `imquotacheck -f`.

To Deploy Quota Notification

Quota notification is the process of sending users a warning message when they are getting close to their quota. Using this feature requires three procedures:

- “Enabling Quota Notification” on page 537
- “Defining a Quota Warning Message” on page 537
- “Specifying a Quota Threshold” on page 538

Enabling Quota Notification

Console To enable quota notification at the Console:

1. Click the Quota tab.
2. Check the “Enable quota notification” box. To disable quota notification, uncheck this box.
3. Define the quota warning messages. See “Defining a Quota Warning Message” on page 537.
4. Click Save.

Command Line To enable or disable 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, no quota warning message is sent to the user. See the next section for an example of quota warning message format.

Defining a Quota Warning Message

Define the message that will be sent to users who are close to exceeding 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. It must contain a header with at least a subject line, follow by $$, then the message body. '$' represents a new line. Example:

```
configutil -o store.quotaexceededmsg -v 'Subject: WARNING: User quota
exceeded$$User quota threshold exceeded - reduce space used.'
```

To define how often the warning message is sent:

```
configutil -o store.quotaexceededmsginterval -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

A quota threshold is a percentage of a quota that is exceeded before clients are sent a warning. 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 and 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 Enable Quota Enforcement

By default, users or domains can exceed their quotas with no effect except for receiving an over quota notication (if set). Quota enforcement locks the mailboxes from receiving further messages until the disk usage is reduced below the quota level.

Enabling Quota Enforcement at the User Level

Console To enable quota enforcement at the Console:

1. Click the Quota tab.
2. Check the “Enable quota enforcement” box. To disable quota enforcement, uncheck this box.
3. Click Save.

Command Line To enable or disable quota enforcement:

```
configutil -o store.quotaenforcement -v [ on | off]
```

Note that messages are added to the MTA queues until the grace period expires and all messages are sent back to the senders, or the disk usage falls below the quota and messages can be dequeued from the MTA and delivered to the message store. If you want to return messages taht are over qutoa before they get to the message queues, use the following command line:

```
configutil -o store.overquotastatus on
```

To reject the message that will push the message store over its quota:

```
configutil -o local.store.quotaoverdraft -v off
```

Setting the value to `on` will accept the message that pushes the message store over its quota. The default is off.

Enabling Quota Enforcement at the Domain Level

To enforce quotas for a particular domain, use the command:

`imquotacheck -f -d` *domain*

To enable for all domains exclude the `-d` option. When a domain exceeds its quota, the `maildomainstatus` attribute is set to `overquota`, which halts all delivery to this domain. If a domain is not `overquota`, the value is set to `active`.

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 bounces all messages including those in the queue. This time limit is controlled by the `quotagraceperiod` `configutil` parameter.
- The message has remained in the message queue longer than the maximum message queue time. This is controlled by the `notices` MTA channel keyword (see "To Set Notification Message Delivery Intervals" on page 265).

For example, if your grace period is set for two days, and you exceed quota for one day, new messages continue to be received and held in the message queue, and delivery attempts continue. After the second day, the messages bounce back to the sender.

NOTE The grace period is not how long the message is held in the message queue, it's how long the mailbox is over quota before all incoming messages, including those in the message queue, are bounced. The grace period starts when the user has reached the quota threshold. See "Specifying a Quota Threshold" on page 538 and been warned.

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 Set the Automatic Message Removal (Expire and Purge) Feature

The automatic message removal feature (also known as expire and purge) automatically removes messages from the message store based on a set of administrator-defined criteria.This feature can automatically remove old or overly large messages, seen / deleted messages, messages with certain Subject: lines and so on. This feature allows the following removal criteria:

- By folder (mailboxes), users, domains, the entire message store, or specific partitions
- Number of messages in the mailbox
- Total size of the mailbox
- Age, in days, that a message has been in the mailbox
- Size of message and grace period (days that an oversized message will remain in the message store before purging)
- Whether a message has been flagged as *seen* or *deleted*
- Header strings

This feature is performed by the `imexpire` utility, which expunges and purges messages. See "How the Message Store Removes Messages" on page 518 for details on the message removal process.

NOTE The server removes messages without warning, so it is important to inform users about automatic message removal policies. Unexpected message removal can be a source of consternation for users and administrators.

`imexpire` Theory of Operation

`imexpire` can be invoked from the command line or scheduled to run automatically by the `imsched` daemon. The administrator configures the global expiration rules (that is rules for the entire message store) with either Console or the `configutil` command line utility. Local expire rules (rules that apply to folders or users) can be configured by creating expire rule files (`store.expire`) in a message store partition, user or mailbox directories.

imexpire loads all of the expire rules at start up. By default, imexpire creates one thread per partition. Each thread goes through the list of user folders under its assigned partition and loads the local expire rule files as it goes. The expire function checks each folder against the expire rules applicable to this folder and expunges messages as needed. If there is a store.exp file that exists under the mailbox directory, and there are messages that have been expunged / expired for longer than the time specified by the store.cleanupage configuration parameter, the purge function will permanently remove the message files under the message hash directories and remove the UID records from the store.exp files.

To Deploy the Automatic Message Removal Feature

Automatic message removal can be deployed by command line or by using the Console GUI. The process requires three steps:

1. Define automatic message removal policy: Which messages will be automatically removed? What users, domains and partitions will have messages automatically removed? What size, message age, headers will define the removal criteria. See "To Define Automatic Message Removal Policy" on page 542.
2. Specify theimexpire rules to implement this policy. See "To Set Rules Implementing Automatic Message Removal Policy" on page 543.
3. Specify the imexpire scheduling. See "To Schedule Automatic Message Removal and Logging Level" on page 551.

To Define Automatic Message Removal Policy

Define your automatic message removal policy by specifying the criteria for removal. imexpire allows for removal using the following criteria:

Age of Message. Automatically remove messages older than *X* days. Attribute: messagedays.

Message Count. Automatically remove messages in a folder exceeding X messages. Attribute: messagecount.

Age of Oversized Message. Automatically remove messages that exceed X bytes after Y days grace period. Attributes: messagesize and messagesizedays.

***Seen* and *Deleted* Message Flag**. Automatically remove messages with the *Seen* or *Deleted* flag set. These criteria can be set to "and" or "or." If set to or, the message's Seen / Delete flag will cause automatic deletion regardless of other criteria. If set to and, the message's Seen / Delete flag must be set along with passing all other specified criteria. Attributes: seen and deleted.

Header Field of Message. Allows you specify a header and string as criteria for removing a message. For example, removing all messages with the header "Subject: Work from Home!"

Folder of Messages. Allows you to specify the folder on which to remove messages. Attribute: folderpattern

NOTE imexpire does not allow you to delete or preserve messages based on how long it has been since that message was read. For example, you cannot specify that messages that have not been read for 200 days will be removed.

Examples of Automatic Message Removal Policy

Example 1: Remove all messages 365 days old in a folder exceeding 1,000 messages.

Example 2: Remove messages in domain siroe.com that are older than 180 days.

Example 3: Remove all messages that have been marked as *deleted*.

Example 4: Remove messages in sesta.com that have been marked as *seen*, are older than 30 days, are larger than 100 kilobytes, from folders exceeding 1,000 messages, with the header X-spam.

To Set Rules Implementing Automatic Message Removal Policy

To implement the automatic message removal policy defined in the previous section, you must set the imexpire rules. Rules can be set in the following ways:

- By GUI (see Figure 18-4 on page 549)
- By setting the store.expirerule.*attribute* configutil parameters

- By putting them into a store.expirerule file. An example of two store.expirerule rules is shown below:

```
Rule1.folderpatter: user/.*/trash
Rule1.messagedays: 2
Rule2.folderpattern: user/.*
Rule2.messagedays: 14
```

In this example, Rule 1 specifies that all messages in the trash folder will be remove after two days. Rule 2 specifies that all messages in the message store will be removed after 14 days.

Expiration Rules Guidelines

This section sets the guidelines for the store.expirerule.*attribute* configutil parameters and the store.expirerule file rules.

- Rules are specified in a file called store.expirerule, or by using the configutil parameter store.expirerule.*rulename.attribute*.
- Multiple expiration criteria can be specified with the same rule. (See example above.)
- Rules can apply to the entire message store (global rules), a message store partition, a user, or a folder. Non-global rules can only be created with the store.expirerule rules.
  - Global rules are created by using the configutil parameter store.expirerule.*rulename.attribute* or by specifying rules in *msg_svr_base*/config/store.expirerule
  - Partition rules can be created by specifying rules in *store_root*/partition/*partition_name*/store.expirerule.
  - User rules can be created by specifying rules in *store_root*/partition/*partition_name*/*userid*/store.expirerule or by specifying the folderpattern rule to be user/*userid*/.*
  - Folder rules can be created by specifying rules in *store_root*/partition/*partition_name*/*userid/folder*/store.expirerule or by specifying the folderpattern rule to be user/*userid/folder*

NOTE User and folder rules can also be placed in the global expire file (*msg_svr_base*/config/store.expirerule) by specifying the folderpattern attribute.

- Multiple expire rules can be applied to a mailbox at the same time. An expire policy for a mailbox consists of global rules and local rules. Local rules apply to the mailbox under the same directory and all of its sub-folders.
- `imexpire` unifies all of the expiration rules applying to a mailbox, unless there is an exclusive rule specified for this mailbox (see Table 18-8). The resulting rule set represents the most restrictive expiration policy based on all applicable rules. For example, if rule X expires messages such that the maximum message life is 10 days, and rule Y specifies 5 days, the union will be 5 days.

Table 18-8 `imexpire` Attributes

Attribute	Description (Attribute Value)
`exclusive`	Specifies whether or not this is an exclusive rule. If specified as `exclusive`, only this rule applies to the specified mailbox(s) and all other rules are ignored. If more than one exclusive rule exists, the last exclusive rule loaded will be used. For example, if a global and a local exclusive rule are specified, the local rule will be used. If there is more than one global exclusive rule, the last global rule listed by `configutil` is used. (yes/no)
`folderpattern`	Specifies the folders affected by this rule. The format must start with a `user/`, which represents the directory *store_root*`/partition/*/` See Figure 18-4 on page 549 and Table 18-9 on page 547. (POSIX regular expression)
`messagecount`	Maximum number of messages in a folder. Oldest messages are expunged as additional messages are delivered. (integer)
`foldersize`	Maximum size of folder before the oldest messages are expunged when additional messages are delivered. (integer in bytes)
`messagedays`	Age of message in days before being expunged. (integer)
`messagesize`	Maximum size of message in bytes before it is marked to be expunged. (integer)
`messagesizedays`	Grace period. Days an over-sized message should remain in a folder. (integer)
message header field	Specifies a header field and string that marks a message for removal. Values are not case-sensitive and regular expressions are not recognized. Example: `Rule1.Subject: Get Rich Now!` For the header *Expires* and *Expiry-Date*, `imexpire` will remove the message if the date value specified with these header fields is older than the `messagedays` attribute. If multiple expiration header fields are specified, the earliest expiration date will be used. (string).
`regexp`	Enable UNIX regular expressions in rules creation. (1 or 0)
`seen`	`seen` is a message status flag set by the system when the user opens a message. If the attribute `seen` is set to `and`, then the message must be seen **and** other criteria must be met before the rule is fulfilled. If the attribute `seen` is set to `or`, then the message only needs to be seen **or** another criteria be met before the rule is fulfilled. (`and`/`or`).

Table 18-8 `imexpire` Attributes

Attribute	Description (Attribute Value)
`deleted`	`deleted` is a message status flag set by the system when the user deletes a message. If the attribute `deleted` is set to and, then the message must be deleted **and** another criteria must be met before the rule is fulfilled. If the attribute `deleted` is set to `or`, then the message only needs to be seen **or** another criteria be met before the rule is fulfilled. (**and/or**)

Setting `imexpire` Rules Textually

Automatic message removal rules can be set textually by using the `configutil` parameter `store.expirerule.`*rulename.attribute* or by specifying rules in a `store.expirerule` file.

The `store.expirerule` file contains one expire criteria per line. An expire criteria of the global rule configuration file (*msg_svr_base*`/data/store/store.expirerule`) has the following format:

rule_name`.`*attribute*`:` *value*

Code Example 18-1 shows a set of expiration rules in *msg_svr_base*`/config/store.expirerule`.

Rule 1 sets the global expiration policy (that is, policy that applies to all messages), as follows:

- Enable UNIX regular expressions in rules creation.
- Removes messages larger than 100,000 bytes after 3 days.
- Removes messages deleted by the user.
- Removes any message with the strings “Viagra Now!” or “XXX Porn!” in the Subject: header.
- Limits all folders to 1,000 messages. After 1,000 messages, the system removes the oldest messages on a folder to keep the total to 1,000.
- Removes all messages older than 365 days.

Rule 2 sets the automatic message removal policy for users at the hosted domain `siroe.com`. It limits mailbox sizes to 1 megabyte, removes messages that have been deleted, and removes messages older than 14 days.

Rule 3 sets the automatic message removal policy for messages in the `inbox` folder of user `f.dostoevski`. It removes messages with a subject line having the expression “On-line Casino.”

Code Example 18-1 Example `imexpire` Rules

```
Rule1.regexp: 1
Rule1.folderpattern: user/.*
Rule1.messagesize: 100000
Rule1.messagesizedays: 3
Rule1.deleted: or
Rule1.Subject: Viagra Now!
Rule1.Subject: XXX Porn!
Rule1.messagecount: 1000
Rule1.messagedays: 365
Rule2.regexp: 1
Rule2.folderpattern: user/.*@siroe.com/.*
Rule2.exclusive: yes
Rule2.deleted: or
Rule2.messagedays: 14
Rule2.messagecount: 1000
Rule3.folderpattern: user/f.dostoevski/inbox
Rule3.Subject: *On-line Casino*
```

Note that this same global expiration policy could be set with `configutil`:

```
% configutil store.expirerule.rule1.regexp 1
% configutil store.expirerule.rule1.messagesizedays 3
% configutil store.expirerule.rule1.deleted or
% configutil store.expirerule.rule1.Subject Viagra Now!
% configutil store.expirerule.rule1.Subject XXX Porn!
% configutil store.expirerule.rule1.messagecount 1000
% configutil store.expirerule.rule1.messagedays 365
% configutil store.expirerule.rule1.messagesize 100000
```

Setting `imexpire` Folder Patterns

Folder patterns can be specified using POSIX regular expressions. The format must start with a `user/`, which represents the directory *store_root*`/partition/*/` (Table 18-9 shows the folder pattern for various folders.)

Table 18-9 `imexpire` Folder Patterns

Folder Pattern	Scope
`user/`*userid*`/.*`	Apply rule to all messages in all folders of *userid*.
`user/`*userid*`/Sent`	Apply rule to messages of *userid* in folder `Sent`:
`user/.*`	Apply rule to entire message store.
`user/.*/trash`	Apply rule to the `trash` folder of all users.
`user/.*@siroe.com/.*`	Apply rule to folders in hosted domain siroe.com:

Table 18-9 `imexpire` Folder Patterns

Folder Pattern	Scope
`user/[^@]*/.*`	Apply rule to folders in default domain.
`user/`*partition_name*`/.*`	Apply rule to specific parition.

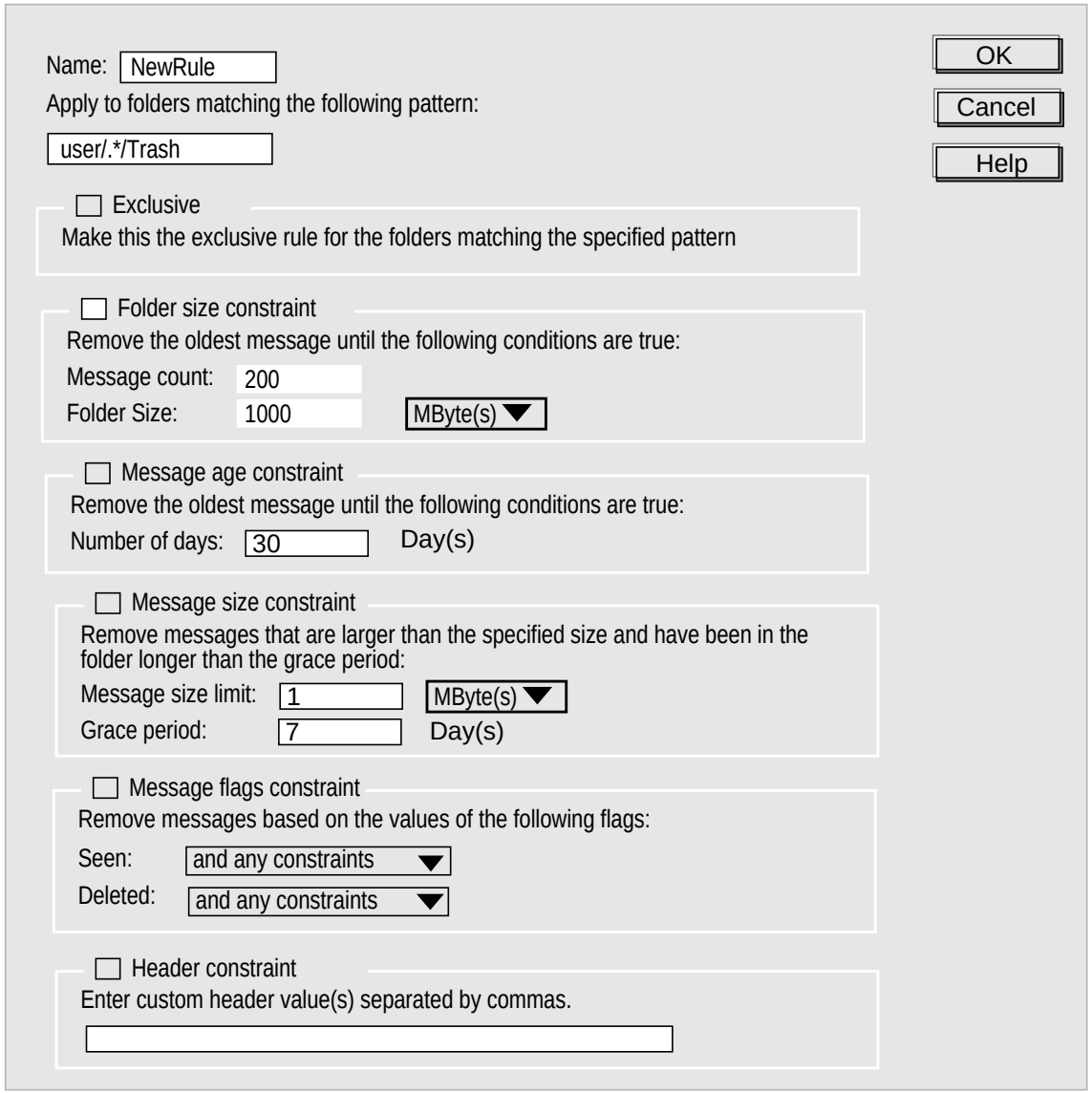
To Set Automatic Message Removal Rules with the Console

1. Bring up the automatic message removal GUI as follows:

   Main Console > Server Group > Messaging Server (Open) >Messaging Server Console > Configuration Tab > Message Store > Expire / Purge >Add

   A very rough drawing of the GUI is shown on Figure 18-4.

Figure 18-4 Automatic Message Removal (Expire / Purge) GUI—Rough Drawing



2. Enter a name for the new rule.
3. Enter the folders from which messages will be automatically removed.

   See "Setting imexpire Folder Patterns" on page 547 above.

4. If this rule is the exclusive rule for folders matching the specified criteria, then check the *Exclusive* box.

   If this box is checked, then this rule takes precedence over all other rules matching the specified pattern. See Table 18-8 on page 545 for details on the exclusive checkbox.

5. To create a rule based on folder size, do the following:

   - Check the Folder Size constraint checkbox. 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 the maximum folder size, in bytes, before the oldest messages are removed.

6. To create a rule based on message age, check the Message age constraint checkbox:

   In the Number of days field, specify the age, in days, of how long messages should remain in the folder.

7. To create a rule based on message size:

   - Check the Message size limit constraint checkbox. In the Message size limit field, enter the maximum size of a message allowed in the folder. In the "Grace period" field, enter the how long over-sized messages will remain in the folder before being removed.

8. To create a rule based on whether the Seen or Deleted message flags are set:

   - Check the Message flags constraint checkbox.
   - For the Seen: field, select "and" to specify that the message must be seen **and** another criteria must be met before the rule is fulfilled. Select "or" to specify that the message need only be seen **or** another criteria be met before the rule is fulfilled.
   - For the Deleted: field, select "and" to specify that the message must be deleted **and** another criteria must be met before the rule is fulfilled. Select "or" to specify that the message need only be deleted **or** another criteria be met before the rule is fulfilled.

9. To create a rule based on a header fields and their values:

   - Check the Header constraint checkbox.

- Enter a comma separated list of headers and values in the following format:

  header1: *value1*, *header2*: *value2*

  Example: `Subject: Work at Home!,From: virus@sesta.com`

For the header *Expires* and *Expiry-Date*, the system will remove the message if their date value is older than the Message age constraint. If multiple expiration header fields are specified, the earliest expiration date will be used. (string).

10. Click OK to add the new rule to the Automatic Message Removal list.

To Schedule Automatic Message Removal and Logging Level

Automatic message removal is activated by the `imsched` scheduling daemon. By default, `imsched` invokes `imexpire` at 23:00 every day, and messages are both expunged and purged. This schedule can be customized by setting the `configutil` parameters `local.schedule.expire`, `local.schedule.purge`, and `store.cleanupage` described in Table 18-10.

Expire and purge can take a long time to complete on a large message store, so you may wish to experiment and decide how often to run these processes. For example, if an expire / purge cycle takes 10 hours, you may not want the default schedule of running expire and purge once a day. Schedule expire and purge by using `local.schedule.purge` to specify a separate schedule for purge. If `local.schedule.purge` is not set, `imexpire` will perform purge after an expire.

Table 18-10 Expire and Purge `configutil` Log and Scheduling Parameters

Parameter	Description

`local.schedule.expire`	Interval for running `imexpire`. Uses UNIX `crontab` format:
	minute hour day-of-month month-of-year day-of-week
	The values are separated by a space or tab and can be 0-59, 0-23, 1-31, 1-12 or 0-6 (with 0=Sunday) respectively. Each time field can be either an asterisk (meaning all legal values), a list of comma-separated values, or a range of two values separated by a hyphen. Note that days can be specified by both day of the month and day of the week, however, it is not typical to use them both since the number of such occurrences are very small. If they are both specified, then both will be required. For example, setting the 17th day of the month and Tuesday will require both values to be true.
	Interval Examples:
	1) Run `imexpire` at 12:30am, 8:30am, and 4:30pm:
	`30 0,8,16 * * *`
	2) Run `imexpire` at weekday morning at 3:15 am:
	`15 3 * * 1-5`
	3) Run `imexpire` only on Mondays:
	`0 0 * * 1`
	Default: `0 23 * * *`

`local.schedule.purge`	Interval for running `purge`. Uses UNIX `crontab` format: *minute hour day-of-month month-of-year day-of-week*. Default: 0 0,4,8,12,16,20 * * * /opt/SUNWmsgsr/lib/purge -num=5 (Every four hours.)
`store.cleanupage`	Age (in hours) of expired or expunged message before `purge` will permanently remove it. Default: None
`local.store.expire.loglevel`	Specify a log level: 1 = log summary for the entire expire session. 2 = log one message per mailbox expired. 3 = log one message per message expired. Default: 1

`imexpire` Scheduling Using the Console

Bring up the automatic message removal GUI as follows:

Main Console > Server Group > Messaging Server (Open) >Messaging Server Console > Configuration Tab > Message Store > Expire / Purge

This Console page lists the expire rules on the top and the expire and purge schedule on the bottom. To schedule expire and purge, use the pull down menus in the Expire / Purge Schedule to set the month, day of month, day of week (with 0=Sunday), hours of day and minute of hours for both expire and purge.

NOTE The day value may be set by both day of the month and day of the week. Both are adhered to if both are set. If you set the 3rd day of the week (Wednesday) and the 17th day of the month, then a purge / expire will only occur on the 17th day of each month that falls on a Wednesday).

Setting `imexpire` Logging Levels

`imexpire` will log a summary to the default log file upon completion. If expire is invoked from the commmand line, the -v (verbose) and -d (debug) option can be used to instruct `imexpire` to log detail status / debug messages to stderr. If `imexpire` is invoked by `imsched`, the `configutil` parameter `local.store.expire.loglevel` can be set to 1,2 or 3 for different levels of logging. Loglevel 1 is the default, it will log a summary for the entire expire session. Loglevel 2 will log one message per mailbox expired. Loglevel 3 will log one message per message expired.

Configuring Message Store Partitions

Mailboxes are stored in message store partitions, an area on a disk partition specifically devoted to storing the message store. Message store partitions are not the same as disk partitions, though for ease of maintenance, it is recommended that you have one disk partition and one file system for each message store partition. Message store partitions are directories specifically designated as a message store.

User mailboxes are stored by default in the *store_root*`/partition/` directory (see Figure 18-1 on page 516). The `partition` directory is a logical directory that might contain a single partition or multiple partitions. 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:

store_root`/partition/mkting/`
store_root`/partition/eng/`
store_root`/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 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 554) 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 *Sun Java System Communcations Services Schema Reference Manual*.

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 Message Store Maintenance 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 20, "Logging and Log Analysis".

This section contains the following:

- "To Monitor Disk Space" on page 561
- "Using the stored Utility" on page 561

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. Note that you should not to kill the `mboxutil` process in the middle of execution. If it is killed with `SIGKILL` (`kill -9`), it may potentially require that every server gets restarted and a recovery be done.

`mboxutil` tasks include the following:

- List mailboxes
- Create 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 560.

Table 18-11 lists the `mboxutil` commands. For detailed syntax and usage requirements, see the *Messaging Server Reference Manual*.

Table 18-11 `mboxutil` Options

Option	Description
`-a`	Obsolete. Used to list all user quota information. Use. `imquotacheck`
`-c` *mailbox*	Creates the specified mailbox.
`-d` *mailbox*	Delete the specified mailbox.
`-f` *file*	Creates, deletes, or locks the mailbox or mailboxes listed in the specified data file.
`-k` *mailbox cmd*	Locks the specified mailbox at the folder level; runs the specified command; after command completes, unlocks the mailbox.

Table 18-11 `mboxutil` Options

Option	Description
`-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 POSIX regular expressions
`-q` *domain*	Obsolete. Use `imquotacheck -d` *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*	Obsolete. Used to list user information. Use `imquotacheck -u` *user*
`-x`	When used in conjunction with the -l option, shows the path and access control for a mailbox.

NOTE POSIX regular expressions can be used in the `mboxutil` command.

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

Monitor quota usage and limits by using `imqutoacheck`, which generates a report listing defined quotas and limits, and provides information on quota usage. Quotas and usage figures are reported in kilobytes. This utility can also compare mailbox size with a user's assigned quota. As an option, you can email a notification to users who have exceeded a set percentage of their assigned quota.

To list the usage of all users whose quota exceeds the least threshold in the rule file:

```
imquotacheck
```

List quota information for a the domain `siroe.com`:

```
imquotacheck -d siroe.com
```

To send a notification to all users in accordance to the default rule file:

```
imquotacheck -n
```

To send a notification to all users in accordance to a specified *rulefile*, *myrulefile*, and to a specified mail template file, *mytemplate.file* (for more information, refer to the *Sun Java System Messaging Server Administration Reference*):

`imquotacheck -n -r` *myrulefile* `-t` *mytemplate.file*

To list the usage of all users and (will ignore the rule file):

```
imquotacheck -i
```

To list per folder usages for users user1 (will ignore the rule file):

```
imquotacheck -u user1 -e
```

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 18-12.

Table 18-12 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 700.

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 710).
- Issuing of alarms if necessary.
- Database recovery as necessary (see "Message Store Startup and Recovery" on page 578).

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 18-13 lists some of the `stored` options. Some common usage examples follow the table. For detailed syntax and usage requirements, see the *Messaging Server Reference Manual*.

Table 18-13 `stored` Options

Option	Description
`-d`	OBSOLETE. Use `start-msg store` to start up stored which will run as daemon, performing system checks and activating alarms, deadlock detection, and database repair.
`-t`	Checks the status of stored. The return code of this command indicates the status.
`-v`	Verbose output.
`-v -v`	More verbose output.

To print the status, enter:

```
stored -t -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:

msg_svr_base/sbin/stop-msg store
msg_svr_base/sbin/start-msg store

If any server daemon crashes, you must stop all daemons and restart all daemons including stored.

Backing Up and Restoring the Message Store

Message store backup and restore is one of the most common and important administrative tasks. It consists of backing up all the messages and folders on a message store. You must implement a backup and restore policy for your message store to ensure that data is not lost if problems such as the following occur:

- System crashes
- Hardware failure
- Accidental deletion of messages or mailboxes
- Problems when reinstalling or upgrading a system
- Natural disasters (for example, earthquakes, fire, hurricanes)
- Migrating users

You can do message store back up and restore using command-line utilities imsbackup and imsrestore, or the integrated solution that uses 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. imsbackup 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 567.

This section contains the following subsections:

Creating a Mailbox 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 as this can reduce system load during peak hours. For example, backups are probably best scheduled for early morning hours such as 2:00 AM.

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 and full backups once a week.

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 if you want to reduce backup impact on 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

A backup group is an arbitrary set of user mailboxes defined by regular expressions. By organizing user mailboxes into backup groups, you can define more flexible backup management.

For example, you could create three backup groups, the first containing user IDs starting with the letters A through L, the second with users whose user IDs begin with M-Z, and the third with users whose user IDs begin with a number. Administrators could use these backup groups to backup mailboxes in parallel, or perhaps only certain groups on one day and other groups on another.

There are several things to remember about backup groups:

1. These are arbitrary *virtual* groups of mail users. They do not precisely map to the message store directory (Figure 18-1 on page 516), although it may look like it.
2. They are defined by the administrator using UNIX regular expressions.
3. The regular expressions are defined in the configuration file *msg_svr_base*`/config/backup-groups.conf`
4. When backup groups are referenced in `imsbackup` and `imsrestore`, they use the path format: */partition_name/backup_group*

The format of `backup-groups.conf` is as follows:

```
group_name=definition
group_name=definition
.
.
.
```

Using the example described in the paragraph above, the following definitions would be used to create the three backup groups:

```
groupA=[a-l].*
groupB=[m,-z].*
groupC=[0-9].*
```

You can now scope `imsbackup` and `imsrestore` at several levels.You can backup / restore the whole message store using the backup command:

`imsbackup -f` *device* `/`

To backup all mailboxes for all users in `groupA` use the following:

`imsbackup -f` *device* */partition*`/groupA`

The default partition is called `primary`.

Pre-defined Backup Group

Messaging Server includes one predefined backup group that is available without creating the backup-groups configuration file. This group is called user; it includes all users. For example, the following will backup all users on the primary partition:

```
imsbackup -f backupfile /primary/user
```

Messaging Server Backup and Restore Utilities

To back up and restore your data, Messaging Server provides the imsbackup and imsrestore utilities. Note that the imsbackup and imsrestore 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, and 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 569.

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.

The following example backs up the entire message store to /dev/rmt/0:

```
imsbackup -f /dev/rmt/0 /
```

This backs up the mailboxes of user ID joe to /dev/rmt/0:

```
imsbackup -f /dev/rmt/0 /primary/user/joe
```

This example backs up all the mailboxes of all the users defined in the backup group groupA to backupfile (see "To Create Backup Groups" on page 564):

```
imsbackup -f- /primary/groupA > backupfile
```

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 /primary/user1
```

For a complete syntax description of the `imsbackup` command, see the *Messaging Server Reference Manual*.

Considerations for Partial Restore

The message store uses a single-copy message system. That is, only a single copy of any message is saved in the store as a single file. Any other instances of that message (like when a message is sent to multiple mailboxes) are stored as links to that copy. Because of this, there are implications when restoring messages. For example:

- **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.

The following examples demonstrate what happens to a message used by multiple users when a partial restore is performed. Assume there are three messages, all the same, 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 mailboxes for users B and C.
2. Delete mailboxes of 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 mailboxes for user A.
3. Restore mailboxes for 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 mailboxes for users A and B.
3. Restore mailboxes for user B.

   The restore utilities ask the administrator to restore A/INBOX first.
4. Restore mailboxes for users A and B.
5. Delete mailboxes for 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.

If the backup device is seekable (example: a drive or tape), imsrestore seeks to the position containing A/INBOX/1 and restore it as B/INBOX/1. If the backup device is non-seekable example: a UNIX pipe), imsrestore logs the object ID and the ID of the depending (linked) object to a file, and the administrator must invoke imsrestore again with the -r option to restore the missing message references.

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 `msg_srv_base/lib/msg/imsasm`
2. From Sun or Legato, obtain a copy of the `nsrfile` binary and copy it to the following directory:

   `/usr/bin/nsr`

   Note that this is required only if you are using an older version of Networker (5.x). With Networker 6.0 and above, `nsrfile` is automatically installed under `/usr/bin/nsr`.
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 564.
   b. To verify your configuration, run `mkbackupdir.sh`.

      Look at the directory structure created by `mkbackupdir.sh`.The structure should look similar to that shown in Table 18-4.

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 save group to invoke the `mkbackupdir.sh` script before the backup. See Table 18-4 for an example.

NOTE Earlier versions of Legato Networker have a limitation of 64 characters for the save set name. If the name of this directory plus the logical name of the mailbox (for example, `/primary/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: `inetuser`).

Figure 18-5 shows a sample backup groups directory structure.

Figure 18-5 Backup Group Directory Structure

```
/backup/primary/groupA/amy
                       /bob
                       /carly
                /groupB/mary
                       /nancy
                       /zelda
                /groupC/123go
                       /1bill
                       /354hut
```

The example below shows a sample `res` file named `IMS.res` in the `/nsr/res` directory:

```
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 save group if necessary.
   a. Run `nwadmin`.
   b. Select Customize | Group | Create.
2. Create a backup client using `savepnpc` as the backup command:
   a. Set the save set to the directory created by `mkbackupdir`.

      For a single session backup, use `/backup`

      For parallel backups, use `/backup/`*`server/group`*

      Be sure you've already created *group* as defined in "To Create Backup Groups" on page 564.

      You must also set the parallelism to the number of backup sessions.

      See "Example. Creating A Backup Client in Networker:" on page 571.
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/primary/groupA
   /backup/secondary/groupB
   /backup/tertiary/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)

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 Messaging Server.

1. Divide your users into groups (see "To Create Backup Groups" on page 564) and create a backup-groups.conf file under the directory *msg_svr_base*/config/.

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- /primary/groupA > /bkdata/groupA &
   # imsbackup -f- /primary/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- /primary/groupA/andy`**

Monitoring User Access

Messaging Server provides the command, `imsconnutil`, which allows you to monitor user's message store access via IMAP, POP and http. You can also determine the last log in and log out of users. This command works on a per message store basis and will not work across message stores.

NOTE Use of this function or other Messaging Server functions to monitor, read or otherwise access user's email may constitute a potential source of liability if used in violation of applicable laws or regulations or if used in violation of the customer's own policies or agreements.

This command requires root access by the system user (default: `inetuser`), and you must set the configuration variables `local.imap.enableuserlist`, `local.http.enableuserlist, local.enablelastaccess` to 1.

To list users currently logged on via IMAP or any web mail client, use the following command:

```
# imsconnutil -c
```

To list the last IMAP, POP, or Messenger Express access (log in and log out) of every user on the message store use:

```
# imsconnutil -a
```

The following command does two things: 1) it determines whether the specified user is currently logged on via IMAP or Messenger Express or any client that connects via mshttp (note that this does not work for POP because POP users generally do not stay connected), and 2) it lists the last time the users have logged on and off:

`# imsconnutil -c -a -u` *user_ID*

Note that a list of users can be input from a file, one user per line, using the following command:

```
# imsconnutil -c -a -f filename
```

You can also specify a particular service (`imap` or `http`) using the `-s` flag. For example, to list whether a particular user ID is logged onto IMAP or not, use the following command:

```
# imsconnutil -c -s imap -u user_ID
```

For a complete description of the `imsconnutil` syntax, refer to the *Sun Java System Messaging Server Administration Reference*.

Here is some example output:

```
$ ./imsconnutil -a -u soroork
UID IMAP last accessHTTP last accessPOP last access
========================================================================
soroork 08/Jul/2003:10:49:0510/Jul/2003:14:55:52----NOT-RECORDED----

$ ./imsconnutil -c
IMAP
UID    TIME                    AUTH              TO                 FROM
========================================================================
ed     17/Jun/2003:11:24:03   plain      172.58.73.45:193  129.157.12.73:2631
bill   17/Jun/2003:04:28:43   plain      172.58.73.45:193  129.158.16.34:2340
mia    17/Jun/2003:09:36:54   plain      172.58.73.45:193  192.18.184.103:3744
jay    17/Jun/2003:05:38:46   plain      172.58.73.45:193  129.159.18.123:3687
paul   17/Jun/2003:12:23:28   plaintext  172.58.73.45:193  192.18.194.83:2943
tony   17/Jun/2003:05:38:46   plain      172.58.73.45:193  129.152.18.123:3688
anil   17/Jun/2003:12:26:40   plaintext  172.58.73.45:193  192.18.164.17:1767
anil   17/Jun/2003:12:25:17   plaintext  172.58.73.45:193  129.150.17.34:3117
jack   17/Jun/2003:12:26:32   plaintext  172.58.73.45:193  129.150.17.34:3119
toni   17/Jun/2003:12:25:32   plaintext  172.58.73.45:193  192.18.148.17:1764
========================================================================
10 users were logged in to imap.
Feature is not enabled for http.
------------------------------------------------------------------------
```

Troubleshooting the Message Store

This section provides guidelines for 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 582.

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 *Sun Java System Messaging Server Administration Reference*. Topics covered in this section include:

- “Standard Message Store Monitoring Procedures” on page 575
- “Common Problems and Solutions” on page 586
- “Message Store Startup and Recovery” on page 578
- “Repairing Mailboxes and the Mailboxes Database” on page 582

Standard Message Store Monitoring Procedures

This section outlines standard monitoring procedures for the message store. These procedures are helpful for general message store checks, testing, and standard maintenance.

For additional information, see “Monitoring the Message Store” on page 708.

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 561 and “Monitoring the Message Store” on page 708.

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 *msg_svr_base*/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 20, "Logging and Log Analysis".

Check User IMAP/POP Session

Messaging Server provides a feature called telemetry that can capture a user's entire IMAP or POP session into a file. This feature is useful for debugging client problems. For example, if a user complains that their message access client is not working as expected, this feature can be used to trace the interaction between the access client and Messaging Server.

To capture a session, simply create the following directory:

msg_svr_base/data/telemetry/*pop_or_imap*/*userid*

Messaging Server will create one file per session in that directory. Example output is shown below.

```
LOGIN redb 2003/11/26 13:03:21
>0.017>1 OK User logged in
<0.047<2 XSERVERINFO MANAGEACCOUNTURL MANAGELISTSURL MANAGEFILTERSURL
>0.003>* XSERVERINFO MANAGEACCOUNTURL {67}
http://redb@cuisine.blue.planet.com:800/bin/user/admin/bin/enduser
MANAGELISTSURL NIL MANAGEFIL
TERSURL NIL
2 OK Completed
<0.046<3 select "INBOX"
>0.236>* FLAGS (\Answered ƒlagged ‡raft ‡eleted \Seen $MDNSent Junk)
* OK [PERMANENTFLAGS (\Answered ƒlagged ‡raft ‡eleted \Seen $MDNSent Junk \*)]
* 1538 EXISTS
* 0 RECENT
* OK [UNSEEN 23]
* OK [UIDVALIDITY 1046219200]
* OK [UIDNEXT 1968]
3 OK [READ-WRITE] Completed
<0.045<4 UID fetch 1:* (FLAGS)
>0.117>* 1 FETCH (FLAGS (\Seen) UID 330)
* 2 FETCH (FLAGS (\Seen) UID 331)
* 3 FETCH (FLAGS (\Seen) UID 332)
```

```
LOGIN redb 2003/11/26 13:03:21
* 4 FETCH (FLAGS (\Seen) UID 333)
* 5 FETCH (FLAGS (\Seen) UID 334)
<etc>
```

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. Run `stored -t -v`
- Check for the log file build up in *store_root*/mboxlist.
- Check for stored messages in the default log file *msg_svr_base*/log/default/default
- Check that the time stamps of the following files (in directory *msg_svr_base*/config/) are updated whenever one of the following functions are attempted by the stored process:

Table 18-14 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 write out. Time stamped once an hour.

For more information on the stored process, see "Using the stored Utility" on page 561 and the stored utility in the Messaging Server Command-line Utilities chapter of the *Messaging Server Reference Manual*.

For additional information on monitoring the stored function, see "Monitoring the Message Store" on page 708.

Check Database Log Files

Database log files refer to sleepycat transaction checkpointing log files (in directory *store_root*/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 no fix) 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 582.

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. On Solaris, use `coreadmin` to configure `core` file location.

Message Store Startup and Recovery

Message store data consists of the messages, index data, and the message store database. While this data is fairly robust, on rare occasions there may be message store data problems in the system. These problems will be indicated in the default log file, and almost always will be fixed transparently. In rare cases an error message in the log file may indicate that you need to run the `reconstruct` utility. In addition, as a last resort, messages are protected by the backup and restore processes described in "Backing Up and Restoring the Message Store" on page 563. This section will focus on the automatic startup and recovery process of `stored`.

The message store automates many recovery operations which were previously the responsibility of the administrator. These operations are performed by message store daemon `stored` during startup and include database snapshots and automatic fast recovery as necessary. `stored` thoroughly checks the message store's database and automatically initiates repairs if it detects a problem.

`stored` also provides a comprehensive analysis of the state of the database via status messages to the default log, reporting on repairs done to the message store and automatic attempts to bring it into operation.

Automatic Startup and Recovery—Theory of Operations

The stored daemon starts before the other message store processes. It initializes and, if necessary, recovers the message store database. The message store database keeps folder, quota, subscription, and message flag information. The database is logging and transactional, so recovery is already built in. In addition, some database information is copied redundantly in the message index area for each folder.

Although the database is fairly robust, on the rare occasions that it breaks, in most cases stored recovers and repairs it transparently. However, whenever stored is restarted, you should check the default log files to make sure that additional administrative intervention is not required. Status messages in the log file will tell you to run reconstruct if the database requires further rebuilding.

Before opening the message store database, stored analyzes its integrity and sends status messages to the default log under the category of *warning*. Some messages will be useful to administrators and some messages will consists of coded data to be used for internal analysis. If stored detects any problems, it will attempt to fix the database and try starting it again.

When the database is opened, stored will signal that the rest of the services may start. If the automatic fixes failed, messages in the default log will specify what actions to take. See "Error Messages Signifying that reconstruct -m is Needed" on page 580 for details.

In previous releases, stored could start a recovery process which would take a very long time leaving the administrator wondering if stored was "stuck." This type of long recovery has been removed and stored should determine a final state in less than a minute. However, if stored needs to employ recovery techniques such as recovering from a snapshot, the process may take a few minutes.

After most recoveries, the database will usually be up to date and nothing else will be required. However, some recoveries will require a reconstruct -m in order to synchronize redundant data in the message store. Again, this will be stated in the default log, so it is important to monitor the default log after a startup. Even though the message store will seem to be up and running normally, it is important to run any requested operations such as reconstruct.

Another reason for reading the log file is to determine what caused damage to the database in the first place. Although stored is designed to bring up the message store regardless of any problem on the system, you will still want to try to ascertain cause of the database damage as this may be a sign of a larger hidden problem.

Error Messages Signifying that `reconstruct -m` is Needed

This section describes the type of error messages that require `reconstruct -m` to be run.

When the error message indicates mailbox error, run `reconstruct <mailbox>`. Example:

```
"Invalid cache data for msg 102 in mailbox user/joe/INBOX. Needs reconstruct"

"Mailbox corrupted, missing fixed headers: user/joe/INBOX"

"Mailbox corrupted, start_offset beyond EOF: user/joe/INBOX"
```

When the error message indicates a database error, run `reconstruct -m`. Example:

```
"Removing extra database logs.  Run reconstruct -m soon after startup to resync
redundant data"

"Recovering database from snapshot.  Run reconstruct -m soon after startup to
resync redundant data"
```

Database Snapshots

A snapshot is a hot backup of the database and is used by `stored` to restore a broken database transparently in couple minutes. This is much quicker than using `reconstruct`, which relies on the redundant information stored in other areas.

Message Store Database Snapshot—Theory of Operations

Snapshots of the database, located in the `mboxlist` directory, are taken automatically, by default, once every 24 hours. Snapshots are copied by default into a subdirectory of the `store` directory. By default, there are five snapshots kept at any given time: one live database, three snapshots, and one database/removed copy. The database/removed copy is newer and is an emergency copy of the database which is thrown into a subdirectory `removed` of the `mboxlist` database directory.

If the recovery process decides to remove the current database because it is determined to be bad, `stored` will move it into the `removed` directory if it can. This allows the database to be analyzed if desired.

The data move will only happen once a week. If there is already a copy of the database there, `stored` will not replace it every time the store comes up. It will only replace it if the data in the `removed` directory is older than a week. This is to prevent the original database which had the problem of being replaced too soon by successive startups.

To Specify Message Store Database Snapshot Interval and Location

There should be five times as much space for the database and snapshots combined. It is highly recommended that the administrator reconfigure snapshots to run on a separate disk, and that it is tuned to the system's needs.

If `stored` detects a problem with the database on startup, the best snapshot will automatically be recovered. Three snapshot variables can set the following parameters: the location of the snapshot file, the interval for taking snapshots, number of snapshots saved. These `configutil` parameters are shown in Table 18-15.

Having a snapshot interval which is too small will result in a frequent burden to the system and a greater chance that a problem in the database will be copied as a snapshot. Having a snapshot interval too large can create a situation where the database will hold the state it had back when the snapshot was taken.

A snapshot interval of a day is recommended and a week or more of snapshots can be useful if a problem remains on the system for a number of days and you wish to go back to a period prior to point at which the problem existed.

`stored` monitors the database and is intelligent enough to refuse the latest snapshot if it suspects the database is not perfect. It will instead retrieve the latest most reliable snapshot. Despite the fact that a snapshot may be retrieved from a day ago, the system will use more up to date redundant data and override the older snapshot data, if available.

Thus, the ultimate role the snapshot plays is to get the system as close to up-to-date and ease the burden of the rest of the system trying to rebuild the data on the fly.

Table 18-15 Message Store Database Snapshot Parameters

Parameter	Description
`local.store.snapshotpath`	Location of message store database snapshot files. Either existing absolute path or path relative to the `store` directory. Default: `dbdata/snapshots`
`local.store.snapshotinterval`	Minutes between snapshots. Valid values: 1 - 46080 Default: 1440 (1440 minutes = 1 day)
`local.store.snapshotdirs`	Number of different snapshots kept. Valid values: 2 -367 Default: 3

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 mailbox database, and repair any inconsistencies. See "Error Messages Signifying that `reconstruct -m` is Needed" on page 580.

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 by automatically on startup.

Table 18-16 lists the `reconstruct` options. For detailed syntax and usage requirements, see the *Messaging Server Reference Manual*.

Table 18-16 `reconstruct` Options

Option	Description
`-e`	Remove `store.exp` file upon reconstruct.
`-i`	Initialize `store.idx` file upon reconstruct.
`-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 mailbox's 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/`

`-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`

Table 18-16 `reconstruct` Options

Option	Description
`-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 585.) 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 *store_root*`/mboxlist`.
  - Restart the server processes.

- Run `reconstruct -m` to build a new mailboxes database from the contents of the spool area.

To Remove Orphan Accounts

To search for orphaned accounts (orphaned accounts are mailboxes that do not have corresponding entries in LDAP) use the following command:

```
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
```

Use the following command to create a file listing orphaned mailboxes that can be turned into a script file that deletes the orphaned mailboxes (file is named `orphans.cmd`):

```
reconstruct -o -d orphans.cmd
```

The command output is as 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 the following factors:

- 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.

The following performance was found with a system 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.

Common Problems and Solutions

This section lists common message store problems and solutions:

- "Messenger Express or Communications Express Not Loading Mail Page" on page 586
- "Command Using Wildcard Pattern Doesn't Work" on page 587
- "Unknown/invalid Partition" on page 587
- "User Mailbox Directory Problems" on page 587

Messenger Express or Communications Express Not Loading Mail Page

If the user cannot load any Messenger Express pages or Communications Express mail page, the problem may be that the data is getting corrupted after compression. This can sometimes happen if the system has deployed a outdated proxy server. To solve this problem, try setting `local.service.http.gzip.static` and `local.service.http.gzip.dynamic` to 0 to disable data compression. If this solves the problem, you may want to update the proxy server.

Command Using Wildcard Pattern Doesn't Work

Some UNIX shells may require quotes around wildcard parameters and some will not. For example, the C shell tries to expand arguments containing wild cards (*, ?) as files and will fail if no match is found. These pattern matching arguments may need to be enclosed in quotes to be passed to commands like `mboxutil`.

For example:

```
mboxutil -l -p user/usr44*
```

will work in the Bourne shell, but will fail with tsch and the C shell. These shells would require the following:

```
mboxutil -l -p "user/usr44*"
```

If a command using a wildcard pattern doesn't work, verify whether or not you need to use quotes around wildcards for that shell.

Unknown/invalid Partition

A user can get the message "Unknown / invalid partition" in Messenger Express if their mailbox was moved to a new partition which was just created and Messaging Server was not refreshed or restarted. This problem only occurs on new partitions. If you now add additional user mailboxes to this new partition, you will not have to do a refresh / restart of Messaging Server.

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 *store_root*`/mboxlist/` user directory to another location outside the message store.

3. To find the user folder that might be causing the problem, 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 559 and the `hashdir` utility in the Messaging Server Command-line Utilities chapter of the *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 (*store_root*`/mboxlist`), but exists in the `partition` directory *store_root*`/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 (*store_root*`/partition/` directory), but is apparently not in the database (*store_root*`/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 582.

Store Daemon Not Starting

If `stored` won't start with the following error message:

```
# msg_svr_base/sbin/start-msg

msg_svr_base: Starting STORE daemon ...Fatal error: Cannot find group in name
service
```

This indicates that the UNIX group configured in `local.servergid` cannot be found. `Stored` and others need to set their `gid` to that group. Sometimes the group defined by `local.servergid` gets inadvertently deleted. In this case, create the deleted group, add `inetuser` to the group, change ownership of the *instance_root* and its files to `inetuser` and the group.

Chapter 19

Configuring Security and Access Control

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 Sun Java System 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 *Sun ONE Server Console 5.2 Server Management Guide* 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 the 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.
- **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 18, "Managing the Message Store".
- **End-user account configuration:** End-user account information can be primarily maintained by using the Delegated Administrator product (valid only for Sun LDAP Schema 1). You can also manage end-user accounts by using the Console interface.
- **Filtering unsolicited bulk email (UBE):** See Chapter 17, "Mail Filtering and Access Control"."

There are 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 documentation web site at http://docs.sun.com.

About HTTP Security

Messaging Server supports user ID/password authentication, client certificate authentication, and Identity Server. 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 5, “Configuring POP, IMAP, and HTTP Services”.”

For information on Identity Server see Chapter 6, “Enabling Single Sign-On (SSO)” on page 131.

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 599.

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 599.

- **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.
- **LOGIN** - This is equivalent to PLAIN and exists only for compatibility with pre-standard implementations of SMTP authentication. By default the mechanism is only enabled for use by SMTP.

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 exposed 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.

Table 19-1 shows some SALS and SASL-related `configutil` parameters. For the latest and most complete listing of `configutil` parameters, see the *Sun Java System Messaging Server Administration Reference*.

Table 19-1 Some SASL and SASL-related `configutil` Parameters

Parameter	Description
`sasl.default.ldap.has_plain_passwords`	Boolean to indicate that directory stores plaintext passwords which enables APOP, CRAM-MD5 and DIGEST-MD5. Default: False
`sasl.default.transition_criteria`	No longer supported or used. See `sasl.default.auto_transition`.
`sasl.default.auto_transition`	Boolean. When set and a user provides a plain text password, the password storage format will be transitioned to the default password storage method for the directory server. This can be used to migrate from plaintext passwords to APOP, CRAM-MD5 or DIGEST-MD5. Default: False
`service.imap.allowanonymouslogin`	This enables the SASL ANONYMOUS mechanism for use by IMAP. Default: False
`service.{imap|pop|http}.plaintextmincipher`	If this is > 0, then disable use of plaintext passwords unless a security layer (SSL or TLS) is activated. This forces users to enable SSL or TLS on their client to login which prevents exposure of their passwords on the network. The MMP has an equivalent option "RestrictPlainPasswords". NOTE: the 5.2 release of messaging server would actually check the value against the strength of the cipher negotiated by SSL or TLS. That feature has been eliminated to simplify this option and better reflect common-case usage. Default: 0
`sasl.default.mech_list`	A space-separated list of SASL mechanisms to enable. If non-empty, this overrides the `sasl.default.ldap.has_plain_passwords` option as well as the `service.imap.allowanonymouslogin` option. This option applies to all protocols (imap, pop, smtp). Default: False
`sasl.default.ldap.searchfilter`	This is the default search filter used to look up users when one is not specified in the inetDomainSearchFilter for the domain. The syntax is the same as `inetDomainSearchFilter` (see schema guide). Default: `(&(uid=%U)(objectclass=inetmailuser))`
`sasl.default.ldap.searchfordomain`	By default, the authentication system looks up the domain in LDAP following the rules for domain lookup (ref. needed) then looks up the user. However, if this option is set to "0" rather than the default value of "1", then the domain lookup does not happen and a search for the user (using the `sasl.default.ldap.searchfilter`) occurs directly under the LDAP tree specified by local.ugldapbasedn. This is provided for compatibility with legacy single-domain schemas, but use is not recommended for new deployments as even a small company may go through a merger or name change which requires support for multiple domains.

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 Data 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 to Transition Users).

Note that MMP has an equivalent option: CRAMs.

To Transition Users

You can use `configutil` to specify information about transitioning users. An example would be 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.auto_transition -v` *value*

For value, you can specify one of the following:

- `no` or `0` - Don't transition passwords. This is the default.
- `yes` or `1` - Do transition passwords.

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.

`sasl.default.mech_list` can be used to enable a list of SASL mechanisms. If non-empty, this overrides the `sasl.default.ldap.has_plain_passwords` option as well as the `service.imap.allowanonymouslogin` option. This option applies to all protocols (imap, pop, smtp).

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 117.

User passwords can be transmitted from the user’s client software to your server as cleartext or in encrypted form. 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 606), 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 608.

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 341.

You can use Authenticated SMTP with or without SSL encryption.

Configuring Encryption and Certificate-Based Authentication

This section contains the following subsections:

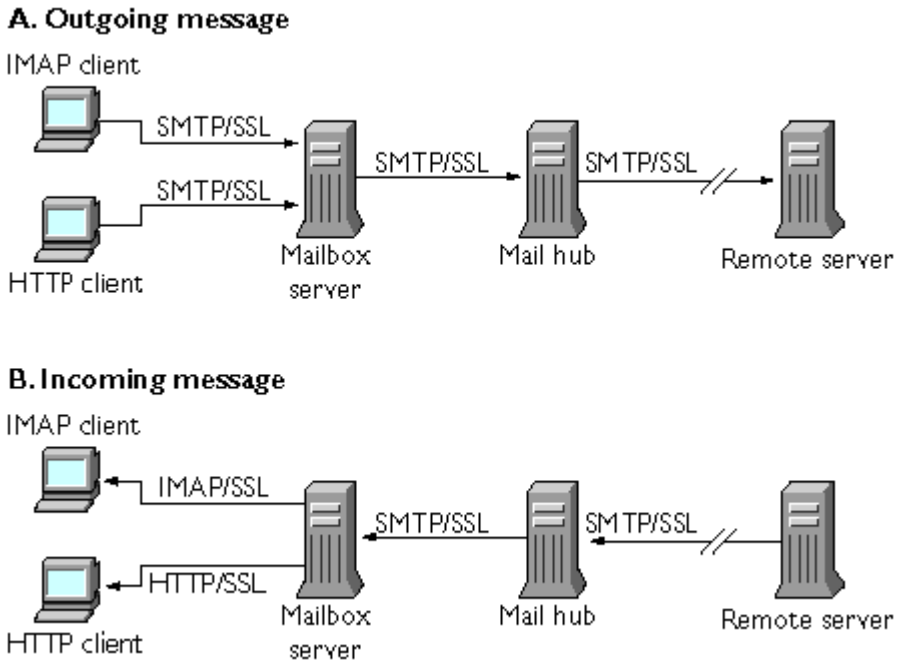
Messaging Server uses the Transport Layer Security (TLS) protocol, otherwise known as the Secure Sockets Layer (SSL) protocol, for encrypted communications and for certificate-based authentication of clients and servers. Messaging Server supports SSL versions 3.0 and 3.1. TLS is fully compatible with 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 iPlanet 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 iPlanet 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, POP3, and SMTP. SMTP and SMTP/SSL use the same port; HTTP and HTTP/SSL require different ports; IMAP and IMAP/SSL, and POP and POP/SSL can use the same port or different ports. SSL acts at a specific stage of message communication, as shown in Figure 19-1, for both outgoing and incoming messages.

Figure 19-1 Encrypted Communications with Messaging Server



SSL provides hop-to-hop encryption, but the message is not encrypted on each intermediate server.

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 *"Transport Layer Security" on page 342* and 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 Sun Java System 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 iPlanet 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).

Sun Java System 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 pre-installed "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 iPlanet 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. 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 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 iPlanet Console*.)

To create a certificate database

1. Log in as or become superuser (`root`).
2. Specify the certificate database password for `certutil` in `/opt/SUNWmsgsr/config/sslPassword`. For example:

   ```
   # echo "password" > /opt/SUNWmsgsr/config/sslpassword
   ```

   where *password* is your specific password.
3. Move to the `sbin` directory and generate the certificate database (`cert7.db`) and key database (`key3.db`). For example:

```
# cd /opt/SUNWmsg/sbin
# ./certutil -N -d /opt/SUNWmsgsr/config -f
/opt/SUNWmsgsr/config/sslpassword
```

4. Generate a default self-signed root Certificate Authority certificate. Example:

```
# . /certutil -S -n SampleRootCA -x -t "CTu,CTu,CTu"
-s "CN=My Sample Root CA, O=sesta.com" -m 25000
-o /opt/SUNWmsgsr/config/SampleRootCA.crt
-d /opt/SUNWmsgsr/config
-f /opt/SUNWmsgsr/config/sslpassword   -z /etc/passwd
```

5. Generate a certificate for the host. For example:

```
../certutil -S -n Server-Cert -c SampleRootCA -t "u,u,u"
-s "CN=hostname.sesta.com, o=sesta.com" -m 25001
-o /opt/SUNWmsgsr/config/SampleSSLServer.crt
-d /opt/SUNWmsgsr/config -f /opt/SUNWmsgsr/config/sslpassword
-z /etc/passwd
```

   where *hostname*.sesta.com is the server host name.

6. Validate the certificates. For example:

```
# ./certutil -V -u V -n  SampleRootCA -d /var/opt/SUNWmsgsr/config
# ./certutil -V -u V -n  Server-Cert -d /opt/SUNWmsgsr/config
```

7. List the certificates. For example:

```
# ./certutil -L -d /opt/SUNWmsgsr/config
# ./certutil -L -n Server-Cert -d /opt/SUNWmsgsr/config
```

8. Use modutil to list the available security modules (secmod.db). For example:

```
# . ./modutil -list -dbdir /opt/SUNWmsgsr/config
```

9. Change the owner of the alias file to icsuser and icsgroup (or the user and group identity under which Calendar Server will run). For example:

```
find /opt/SUNWmsgsr/config/cert7.db -exec chown mailsrv {} \;
find /opt/SUNWmsgsr/config/key3.db -exec chown mailsrv {} \;
find /opt/SUNWmsgsr/config/cert7.db -exec chgrp mail {} \;
find /opt/SUNWmsgsr/config/key3.db -exec chgrp mail {} \;
```

   where mailsrv is the mail server UID and mail is the mail server GID.

10. Restart the messaging services to enable the SSL.

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.
3. Change the certificate nickname from `server-cert` to `Server-Cert`.

   If you do not want to change the certificate nickname, you can change what the system wants the certificate nickname to be by setting the `configutil` parameter `encryption.rsa.nssslpersonalityssl`.

(For more complete instructions, see the chapter on SSL in *Managing Servers with iPlanet 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 Public-Key Cryptography in Managing Servers with iPlanet 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 Sun Java System 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 iPlanet 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 iPlanet Console.*

Creating a Password File

On any Sun Java System 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 Sun Java System 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 *msg_svr_base*`/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. 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 iPlanet Console*.

Table 19-2 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.
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.

To enable SSL and select encryption ciphers, follow these command line steps:

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.

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 *"Transport Layer Security" on page 342* and the *Messaging Server Reference Manual*.

To Set Up Certificate-Based Login

In addition to password-based authentication, Sun Java System 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 601.)

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 604.)

   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 606.)

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 iPlanet 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.

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 624 for instructions on how to install the SMTP Proxy.

Configuring Administrator Access to Messaging Server

This section mostly pertains to the Sun Java System LDAP Schema v. 1. 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 Sun Java System 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 iPlanet Console.*

Hierarchy of Delegated Administration

When you install the first Sun Java System 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 (if Sun Java System LDAP Schema v. 1 is used) for Messaging Server:

1. **Configuration administrator.** The “super user” for the network of Sun Java System 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 612.

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 iPlanet 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 iPlanet Console.*)

ACIs and how to create them are described more fully in the chapter on delegating server administration in *Managing Servers with iPlanet Console.*

Configuring Client Access to POP, IMAP, and HTTP Services

This section contains the following subsections:

- "To Create Access Filters for Services" on page 621
- "To Create Access Filters for HTTP Proxy Authentication" on page 622
- "How Client Access Filters Work" on page 613

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 17, "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 620 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 621 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 616.)

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 618 and "Client User-Name Specification" on page 618.)

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 617.

You can also use the following syntax:

"+" or "-" *serviceList*:*$*next_rule*

+ (allow filter) means the daemon list services are being granted to the client list.

- (deny filter) means the services are being denied to the client list.

* (wildcard filter) allow all clients to used these services.

$ separates rules.

This example enables multiple services on all clients.

```
+imap,pop,http:*
```

This example shows multiple rules, but each rule is simplified to have only one service name and uses wildcards for the client list. (This is the most commonly used method of specifying access control in LDIF files.)

```
+imap:ALL$+pop:ALL$+http:ALL
```

An example of how to disallow all services for a user is:

```
-imap:*$-pop:*$-http:*
```

Wildcard Names

You can use the following wildcard names to represent service names, host names or addresses, or user names:

Table 19-3 Wildcard Names for Service Filters

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 19-3 Wildcard Names for Service Filters

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 Sun Java System 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 17, “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:

   a. 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.

   b. 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.

   c. 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 614. For additional examples, see “Filter Examples” on page 619.

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 614.

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 614.

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 519.) 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, Sun Java System 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 Sun Java System 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 614.

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 614.

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 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 a Messaging Multiplexor (MMP) as described in the *Sun ONE Messaging Server Installation Guide*.

2. Enable the SMTP proxy on the MMP.

   Add the string:

   msg_svr_base/lib/SmtpProxyAService@25|587

   to the ServiceList option in the *msg_svr_base*/config/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. Make sure the IP address that the MMP uses to connect to the SMTP server is not treated as "internal" by the INTERNAL_IP mapping table.

   See the "To Add SMTP Relaying" on page 493 of the Chapter 17, "Mail Filtering and Access Control"for information about the INTERNAL_IP mapping table.

5. Configure the SMTP proxy to Support POP before SMTP.

   a. Edit the *msg_svr_base*/config/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 *Messaging Server Installation Guide* and the description of these options in the Encryption (SSL) Option section in the *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 609).
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 609.)

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 17, "Mail Filtering and Access Control"."

Chapter 20

Logging and Log Analysis

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 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

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. These are located in *msg_svr_base*/data/log. You can customize and view each type of log file individually. Table 20-1 lists the services that can be logged, and describes the log files for each service.

Table 20-1 Logged Services

Service	Log-file description
Admin	Contains logged events related to communication between 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 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 20-1).

For these services, you can use 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 642.

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 636). You can also use logging levels to filter searches for log events. Table 20-2 describes the available levels. These logging levels are a subset of those defined by the UNIX `syslog` facility.

Table 20-2 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 636.

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 20-3 lists the categories that Messaging Server recognizes for logging purposes.

Table 20-3 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 640.

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 20-4 lists the message store log filename conventions.

Table 20-4 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 20-4 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 636.

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 636.

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 20-5 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 20-5 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 20-3 on page 633).
logLevel	The level of logging that the event represents: for example, `Notice` (see Table 20-2 on page 632).
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 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 640.

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 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 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.

If your system does not support http message access, that is, web mail, you can disable http logging by setting the following variables. Do not set these variables if your system requires web mail support (for example, Messenger Express).

```
configutil -o service.http.enable -v no
configutil -o service.http.enablesslport -v no
```

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

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 631). 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 632). 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 634) 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 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 631.

If you wish to have all of your channels log message activity to the logging file, simply add the `logging` keyword to your defaults channel (see “Configuring Channel Defaults” on page 322) of the channel block section of your MTA configuration file. For example:

```
defaults logging notices 1 2 4 7 copywarnpost copysendpost postheadonly
noswitchchannel immnonurgent maxjobs 7 defaulthost siroe.com

l defragment charset7 us-ascii charset8 iso-8859-01
siroe.com
```

Each message is logged as it is enqueued and dequeued. All log entries are made to *msg_svr_base*`/data/logmail.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) 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 the example below.

```
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 (example: `19-Jan-1998 19:16:57.64`).
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 20-6.
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 20-6 describes the logging entry codes.

Table 20-6 Logging Entry Codes

Entry	Description
D	Successful dequeue
DA	Successful dequeue with SASL (authentication)
DS	Successful dequeue with TLS (security)
DSA	Successful dequeue with TLS and SASL (security and authentication)
E	Enqueue
EA	Successful enqueue with SASL (authentication)

Table 20-6 Logging Entry Codes

Entry	Description
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)
K	Recipient message rejected. If the sender requests `NOTIFY=NEVER` DSN flag set or if the message times out or if the message is manually returned (for example: `imsimta qm` “delete” command always generates a "K" record for each recipient, while a qm "return" command will generate a "K" record rather than an "R" record). This indicates that there was no notification sent to the sender per the sender’s own request. This can be compared with "R" records, which are the same sort of rejection/time-out, but where a new notification message (back to the original sender) is also generated regarding this failed message.
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 sent to notify original sender that the message has not been delivered yet, but it is still in the queue being retried.
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 the example below. (The sample log entry line has been wrapped for typographic reasons; the actual log entry would appear on one physical line.)

```
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 337. 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 *msg_svr_base*`/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 644.

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`.

The example below shows 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.

Code Example 20-1 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.

Code Example 20-2 shows a logging entry similar to that shown in Code Example 20-3, 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.

Code Example 20-2 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.
```

Code Example 20-3 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.

Code Example 20-3 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.
```

Code Example 20-4 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).

Code Example 20-4 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>
```

Code Example 20-5 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 Code Example 20-4, 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`.

Code Example 20-5 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>
```

Code Example 20-6 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 496) 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.

Code Example 20-6 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 Code Example 20-4 and Code Example 20-5).
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.

Code Example 20-7 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.

Code Example 20-7 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 Code Example 20-4, 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.

Code Example 20-8 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`.

Code Example 20-8 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
 <01IT5UALL144985180@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
 <01IT5UALL144985180@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.

Code Example 20-9 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.

Code Example 20-9 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 `0` 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.

Code Example 20-10 illustrates log output for an incoming SMTP message when connection logging is enabled, via LOG_CONNECTION=3.

Code Example 20-10 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 12, "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 20-7.

Table 20-7 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.
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.

Table 20-7 Dispatcher Debugging Bits *(Continued)*

Bit	Hexadecimal value	Decimal value	Usage
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 21

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 22, "Monitoring the Messaging Server"

NOTE Prior to reading this chapter, you should review Chapters 5 through 10 in this guide and the MTA configuration and command-line utility chapters in the *Sun Java System Messaging Server Administration Reference*.

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 667
- “Check the Message Queue Directories” on page 667
- “Check the Ownership of Critical Files” on page 667
- “Check that the Job Controller and Dispatcher are Running” on page 668
- “Check the Log Files” on page 670
- “Run a Channel Program Manually” on page 671
- “Starting and Stopping Individual Channels” on page 671
- “An MTA Troubleshooting Example” on page 673

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 *Sun Java System Messaging Server Administration Reference* 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 *msg_svr_base*`/data/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 *Sun Java System Messaging Server Administration Reference* and "imsimta qm counters" on page 719.

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 642). You should then look at the `mail.log_current` file in the directory */msg_svr_base*`/log/`. 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 Messaging Server. The following directories, subdirectories, and files should be owned by this account:

/msg_svr_base`/data/queue/`
/msg_svr_base`/log/`
`/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 /opt/SUNWmsgsr/data/queue
drwx------ 6 nobody bin 512 Feb 7 09:32 /opt/SUNWmsgsr/data/queue
ls -l -p -d /opt/SUNWmsgsr/log/imta
drwx------ 2 nobody bin 1536 Mar 10 09:00 /opt/SUNWmsgsr/log/imta
ls -l -p -d /opt/SUNWmsgsr/imta/tmp
drwx------ 2 nobody bin 512 Feb 7 10:00 /opt/SUNWmsgsr/imta/tmp
```

Check that the files in */msg_svr_base*/data/queue are owned by the MTA account by using commands like in the following UNIX system example:

```
ls -l -p -R /opt/SUNWmsgsr/data/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
inetuser  9567  S   18416    9368   02:00:02    0:00     /opt/SUNWmsgsr/lib/tcp_smtp_server

inetuser  6573  S   18112    5720    Jul_13     0:00     /opt/SUNWmsgsr/lib/job_controller

inetuser  9568  S   18416    9432    02:00:02    0:00    /opt/SUNWmsgsr/lib/tcp_smtp_server

inetuser  6574  S   17848    5328    Jul_13     0:00    /opt/SUNWmsgsr/lib/dispatcher
```

If the Job Controller is not present, the files in the */msg_svr_base*`/data/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 *Sun Java System Messaging Server Administration Reference*.

If neither the Job Controller nor the Dispatcher is present, you should review the `dispatcher.log-*` or `job_controller.log-*` file in */msg_svr_base*`/data/log`

If the log files do not exist or do not indicate an error, start the processes by using the `msg-start` command. For more information, refer to the MTA command-line utilities chapter in the *Sun Java System Messaging Server Administration Reference*.

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 */msg_svr_base*/log. Log file name formats for various MTA processing jobs are shown in Table 21-1.

Table 21-1 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 *Sun Java System Messaging Server Administration Reference.*

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: */msg_svr_base*/config/).

For more information on debugging channel master and slave programs, see the *Sun Java System Messaging Server Administration Reference*.

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 */msg_svr_base*/log as shown in Table 21-1.

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 *Sun Java System Messaging Server Administration Reference*.

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 *Sun Java System Messaging Server Administration Reference*.

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 481.

- 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 */msg_svr_base*`/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 */msg_svr_base*`/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 666). 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 */msg_svr_base*`/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 */msg_svr_base*/log.

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 645 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 671 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 676.

1. Set the `mm_debug=5` in the `option.dat` file in directory */msg_svr_base*/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 */msg_svr_base*/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 673. 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 671.

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 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 */msg_svr_base*/`data/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 */msg_svr_base*`/log`. 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 Step 7 should correlate to the message files that you copied in Step 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 */msg_svr_base*`/config`.

   b. Reset the `mm_debug=0`, and remove `log_message_id=1` in the `option.dat` file in directory */msg_svr_base*`/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 Messaging Server MTA, you will need to repeat the entire troubleshooting process (See "Identify the Channels in the Message Path" on page 673, "Manually Start and Stop Channels to Gather Data" on page 674, 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.

- "Received Message is Encoded" on page 687
- "Server-Side Rules (SSR) Are Not Working" on page 688

TLS Problems

If, during smtp dialog, the STARTTLS command returns the following error:

```
454 4.7.1 TLS library initialization failure
```

and if you have certificates installed and working for pop / imap access, check the following:

- Protections / ownerships of the certificates have to be set so mailsrv account can access the files
- The directory where the certificates are stored need to have protections / ownerships set such that the mailsrv account can access the files within that directory.

After changing protections and installing certificates, you must run:

```
imsimta shutdown dispatcher
start-msg dispatcher
```

Restarting should work, but it is better to shut it down completely, install the certificates, and then start things back up.

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 Messaging Server installation. Refer to the *Sun ONE Messaging Server Installation Guide* 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 *Sun Java System Messaging Server Administration Reference* for more information.

Dispatcher (SMTP Server) Won't Start Up

If the dispatcher won't start up, first check the `dispatcher.log-*` for relevant error messages. If the log indicates problems creating or accessing the `/tmp/.SUNWmsgsr.dispatcher.socket` file, then verify that the `/tmp` protections are set to 1777. This would show up in the permissions as follows:

```
drwxrwxrwt    8 root    sys          734  Sep 17 12:14      tmp/
```

.

Do not remove the `.SUNWmsgsr.dispatcher.file` and do not create it if it's missing. The dispatcher will create the file. If protections are not set to 1777, the dispatcher will not start or restart because it won't be able to create/access the socket file. In addition, there may be other problems occurring not related to the Messaging Server.

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 (Sun Java System Messaging Server 6.1
(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.

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 (Step 1)

Server: LOCALHOST
Address: 127.0.0.1

Non-authoritative answer:
XTEL.CO.UK  preference = 10, mail exchanger = nsfnet-relay.ac.uk (Step 2)

% telnet nsfnet-relay.ac.uk 25 (Step 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 338.
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 Messaging Server on a TCP/IP network that does not use DNS, you can skip steps (Step 1) and (Step 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 */msg_svr_base*`/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 187 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 686. 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 */msg_svr_base*/`data/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 *Sun Java System Messaging Server Administration Reference* 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).

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 */msg_svr_base*`/config` in order to get sufficient debugging information.

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 667.

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 *Sun Java System Messaging Server Administration Reference* and chapters 5 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:

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 *Sun Java System Messaging Server Administration Reference* 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 *Sun Java System Messaging Server Administration Reference* 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 *Sun Java System Messaging Server Administration Reference* and Chapter 12, "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 *Sun Java System Messaging Server Administration Reference* 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`

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 *Sun Java System Messaging Server Administration Reference*.

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 667.

"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 667 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 332 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 22

Monitoring the 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 Messaging Server. It consists of the following sections:

Troubleshooting procedures can be found in Chapter 21, "Troubleshooting the MTA".

Automatic Monitoring and Restart

Messaging Server provides a way to transparently monitor services and automatically restart them if they fail or become unresponsive (the services hangs or freeze up). It can monitor all message store, MTA, and MMP services including the IMAP, POP, HTTP, job controller, dispatcher, and MMP servers. It does not monitor other services such as ENS, SMS, LMTP or TCP/SNMP servers. (LMTP and TCP/SNMP are monitored by the job controller.) Refer to "Automatic Restart of Failed or Unresponsive Services" on page 100 for details.

In addition, this feature generates the log file *msg_svr_base*/data/log/watcher, shown below, which records all server starts and stops. This is a very important file for monitoring system health.

```
watcher process 13425 started at Tue Oct 21 15:29:44 2003

Watched 'imapd' process 13428 exited abnormally
Received request to restart:  store imap pop http
Connecting to watcher ...
Stopping http server 13440 .... done
Stopping pop server 13431 ... done
Stopping pop server 13434 ... done
Stopping pop server 13435 ... done
Stopping pop server 13433 ... done
imap server is not running
Stopping store server 13426 .... done
Starting store server .... 13457
checking store server status ...... ready
Starting imap server ..... 13459
Starting pop server ....... 13462
Starting http server ...... 13471
```

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

Messaging Server creates a separate set of log files for each of the major protocols, or services, it supports: SMTP, IMAP, POP, and HTTP. These are located in *msg_svr_base*/data/log. 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 20, “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 710.

Monitoring System Performance

This chapter focuses on 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

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. See "immonitor-access" on page 710.

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 22-1 on page 711 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 185, "Messages are Not Dequeued" on page 681, and "MTA Messages are Not Delivered" on page 683 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 719.

You can also monitor the number of files in the queue directories (*msg_svr_base*`/data/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 record in the file *msg_svr_base*`/data/log/mail.log_current`. Example:

```
mail.log:06-Oct-2003 00:24:03.66 501d.0b.9 ims-ms    Q  5
durai.balusamy@Sun.COM rfc822;durai.balusamy@Sun.COM durai@ims-ms-daemon
<00ce01c38bda$c7e2b240$6501a8c0@guindy> Mailbox is busy
```

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_svr_base*/log/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 668.

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:

```
Client 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 watcher. See "Automatic Restart of Failed or Unresponsive Services" on page 100.
- 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_svr_base*/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 20, "Logging and Log Analysis".")

  If a server was shut down you might see something like this:

  ```
  imap.12.1065431243:[07/Oct/2003:01:15:43 -0700] gotmail-2
  imapd[20525]: General Warning: Sun Java System Messaging Server
  IMAP4 6.1 (built Sep 24 2003) shutting down
  ```
- Can be checked with `counterutil`. See "counterutil" on page 712 and the *Sun Java System Messaging Server Administration Reference*.
- 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`.
- You can set alarms for specified server performance thresholds by setting the server response configuration parameters described in "Recommended `stored` Parameters" on page 711.
- See "immonitor-access" on page 710.

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 *Sun Java System Messaging Server Administration Reference*.

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_svr_base*/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
  inetuser 28250     1  0   Jan 05 ?       8:44 /opt/SUNWmsgsr/lib/stored -d
  ```

- Check for log file build up in *msg_svr_base*/store/mboxlist. 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_svr_base*/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_svr_base*/log/default/default

Monitoring LDAP Directory Server

This section consists of the following subsection:

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 Console to monitor slapd.
- See also "immonitor-access" on page 710

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

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 mboxlist Directory

Database log files refer to sleepycat transaction checkpointing log files (*msg_svr_base*`/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_svr_base*`/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" on page 710

immonitor-access

immonitor-access monitors the status of the following Messaging Server components / processes: Mail Delivery (SMTP server), Message Access and Store (POP and IMAP servers), Directory Service (LDAP server) and HTTP server. This utility measures the response times of the various services and the total round trip time taken to send and retrieve a message. The Directory Service is monitored by looking up a specified user in the directory and measuring the response time. Mail Delivery is monitored by sending a message (SMTP) and the Message Access and Store is monitored by retrieving it. Monitoring the HTTP server is limited to finding out weather or not it is up and running.

For complete instructions, refer to *Sun Java System Messaging Server Administration Reference*.

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 706).

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: /opt/SUNWmsgsr
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 22-1 shows useful `stored` parameters along with their default setting.

Table 22-1 Recommended `stored` Parameters

Parameter	Description (Default in parenthesis)
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.
alarm.serverresponse.msgalarmthreshold	(10) If server response time in seconds exceeds this value, alarm issued.

Table 22-1 Recommended `stored` Parameters *(Continued)*

Parameter	Description (Default in parenthesis)
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:

```
# /opt/SUNWmsgsr/sbin/counterutil -l
Listing registry (/opt/SUNWmsgsr/data/counter/counter)
numobjects = 11
refcount = 1
created = 25/Sep/2003:02:04:55 -0700
modified = 02/Oct/2003:22:48:55 -0700
       entry = alarm
       entry = diskusage
       entry = serverresponse
       entry = db_lock
       entry = db_log
       entry = db_mpool
       entry = db_txn
       entry = imapstat
       entry = httpstat
       entry = popstat
       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 *Sun Java System Messaging Server Administration Reference.*

`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 22-2 `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.

Table 22-2 `counterutil alarm` Statistics

Suffix	Description
`alarm.timereset`	The last time reset was performed.
`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 22-3. The `popstat` and `httpstat` objects provide the same information in the same format and structure.

Table 22-3 `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 22-4 `counterutil diskstat` Statistics

Suffix	Description
`diskusage.availSpace`	Total space available in the disk partition.
`diskusage.lastStatTime`	The last time statistic was taken.

Table 22-4 `counterutil diskstat` Statistics

Suffix	Description
`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 22-5 `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 20, "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 20, “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 spent noticeable 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 `inetuser` to run this utility. The output fields are the same as those described in "imsimta counters" on page 716. See also *Sun Java System Messaging Server Administration Reference* for usage details.

Example:

```
# imsimta counters -create
# imsimta qm counters show

Channel                Messages    Recipients     Blocks
---------------------  ----------  ----------    ----------
tcp_intranet
   Received              13077        13859         264616
   Stored                   92           91           -362
   Delivered             12985        13768         264978
   Submitted              2594         2594           3641
...
```

Every time you restart the MTA, you must run: **`# imsimta counters -create`**

MTA Monitoring Using SNMP

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 Messaging Server. Refer to Appendix A, "SNMP Support"" for details.

imquotacheck for Mailbox Quota Checking

You can monitor mailbox quota usage and limits by using the `imquotacheck` utility. The `imquotacheck` utility generates a report that lists defined quotas and limits, and provides information on quota usage.

For example, the following command lists all user quota information:

```
% imquotacheck
-------------------------------------------------------------------------
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:

```
% imquotacheck -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 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 Messaging Server. For more information on monitoring the Messaging Server refer to Chapter 22, "Monitoring the 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 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 platforms running Solaris 8 and later. 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.

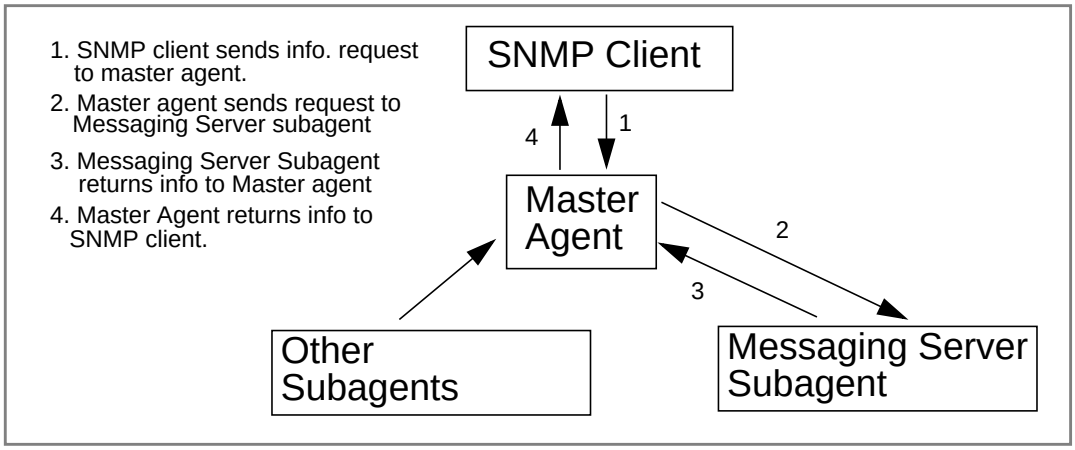
Limitations of the Messaging Server SNMP support are as follows:

- 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 Messaging Server on Solaris 8

Although the overhead imposed by SNMP monitoring is very small, the Messaging Server nonetheless ships with SNMP support disabled. To enable the SNMP support, run the following commands:

```
# su user-id-for-ims
# configutil -o local.snmp.enable -v 1
# start-msg snmp
```

Once you have enabled SNMP, the `start-msg` command, without any parameters specified, will automatically start the SNMP subagent process along with the other Messaging Server processes.

Note that the Solaris native SNMP master agent must be running in order for the Messaging Server SNMP subagent to operate. The Solaris native SNMP master agent is the `snmpdx` daemon which is normally started as part of the Solaris boot procedure.

The SNMP subagent will automatically select a UDP port on which to listen. Should you require, you can assign a fixed UDP port to the subagent with the following command:

```
# configutil -o local.snmp.port -v port-number
```

You may later undo this setting by specifying a value of zero for the port number. A value of zero, the default setting, tells Messaging Server to allow the subagent to automatically select any available UDP port.

Two SNMP subagent configuration files are placed in the `/etc/snmp/conf` directory: `ims.acl` which contains SNMP access control information, and `ims.reg` which contains SNMP MIB OID registration information.

Normally, there should be no reason to edit either of these files. The MIBs served out by Messaging Server are read-only and there's no need to specify a port number in the `ims.reg` file. If you do specify a port number, then it will be honored unless you also set a port number with the `configutil` utility. In that case, the port number set with `configutil` is the port number which will be used by the subagent. If you do edit the files, then you will need to stop and restart the SNMP subagent in order for your changes to take effect:

```
# stop-msg snmp
# start-msg snmp
```

Monitoring from an SNMP Client

The base OIDs for RFC 2788 and RFC 2789 are

mib-2.27 = 1.3.6.1.2.1.27

mib-2.28 = 1.3.6.1.2.1.28

Point your SNMP client at those two OIDs and access as the "public" SNMP community.

If you wish to load copies of the MIBs into your SNMP client, ASCII copies of the MIBs are located in the *msg_svr_base*`/lib/config-templates` 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 Sun Java System Products on Unix Platforms

Other Netscape or Sun Java System products which provide SNMP support may do so by displacing the platform's native SNMP master agent. If you will be running such Sun Java System products on the same host as Messaging Server and wish to monitor both via SNMP, then configure the Sun Java System Proxy SNMP Agent as described in Chapter 11 of *Managing Servers with iPlanet Console (*http://docs.sun.com/source/816-5572-10/11_snmp.htm*)*. This allows the Messaging Server SNMP subagent—a native SNMP subagent—to co-exist with the non-native Sun Java System SNMP subagents in the other Sun Java System 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 = Sun Java System Messaging Server 6.1
    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] = tcp_intranet[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 tcp_intranet 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 tcp_intranet 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).

Appendix B

Administering Event Notification Service in Messaging Server

This appendix describes what you need to do to enable the Event Notification Service Publisher (ENS Publisher) and administer Event Notification Service (ENS) in Messaging Server.

This chapter / appendix contains these sections:

- Loading the ENS Publisher in Messaging Server
- Running Sample Event Notification Service Programs
- Administering Event Notification Service

For more information on ENS and ENS APIs, see the *Event Service Notification Manual for Sun Java System Communications Services* at the Sun Java System Calendar Server at http://docs.sun.com/db/coll/CalendarServer_04q2 and Messaging Server Documentation web page at http://docs.sun.com/db/coll/MessagingServer_04q2.

Loading the ENS Publisher in Messaging Server

The Event Notification Service (ENS) is the underlying publish-and-subscribe service. ENS acts as a dispatcher used by Sun Java System 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 Messaging Server) are bundled starting with Messaging Server. By default ENS is enabled, however, iBIFF is not loaded. (See "To Load the ENS Publisher on Messaging Server."

In order to subscribe to notifications in Messaging Server, you need to load the `libibiff` file on the Messaging Server host then stop and restart the messaging server.

To Load the ENS Publisher on Messaging Server

Perform the following steps from the command line. In these steps, the location of the Messaging Server installation directory is *msg_svr_base*, and the Messaging Server user is *inetuser*. Typical values for these variables are `/opt/SUNWmsgsr`, and `inetuser`, respectively.

1. As `inetuser`, run the `configutil` utility to load the `libibiff` file.

   `cd` *msg_svr_base*

   `./configutil -o "local.store.notifyplugin" -v "`*msg_svr_base*`/lib/libibiff"`**

2. As `root`, stop then restart the messaging server.

   `cd` *msg_svr_base*`/sbin`

   `./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

Messaging Server contains sample programs to help you learn how to receive notifications. These sample programs are located in the *msg_svr_base*`/examples` directory.

To Run the Sample ENS Programs

1. Change to the *msg_svr_base*/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 *msg_svr_base*/examples 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 *msg_svr_base*/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:

  msg_svr_base`/sbin/start-msg ens`

- To stop ENS:

  msg_svr_base`/sbin/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 B-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 0 bytes.
`local.store.notifyplugin.maxBodySize`	Specifies the maximum size (in bytes) of the body that will be transmitted with the notification. The default is 0 bytes.
`local.store.notifyplugin.`*`eventType`*`.enable`	Specifies if the given event type will generate a notification. See the *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 C

Managing Mail Users and Mailing Lists with the Console Interface (NOT RECOMMENDED)

This appendix is for reference purposes only. **DO NOT use the Console interface as described in this appendix for creating and managing user and mailing lists.** Use another approved provisioning tool such as the User Management Utility.

CAUTION Using the Console interface to create users and groups will cause a variety of problems. Use another approved provisioning tool such as the User Management Utility. (Refer to the *Sun Java System Messaging Server Administration Reference*.)

This appendix is for reference purposes only. We recommend that you **do not** use the Console interface to create and manage your users' mail accounts and mailing lists.

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 the chapter on User and Group Administration, of the *Sun ONE Server Console 5.2 Server Management Guide*.)

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 the chapter on User and Group Administration, of the *Sun ONE Server Console 5.2 Server Management Guide*.
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 739.
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).

If the iPlanet Delegated Administrator for Messaging is used, it also provides an end-user HTML interface through which users can themselves enable and configure these options. The Console interface and the iPlanet 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.

NOTE The Delegated Administrator for Messaging only supports Sun Java System LDAP Schema v. 1, not v.2

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 739.
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 743.
   - To specify program delivery, follow the instructions in “Specifying Program Delivery” on page 744.
   - To specify UNIX delivery, follow the instructions in “To Specify UNIX Delivery” on page 745.
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 513.
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 535), 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 535), 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.

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.

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.

NOTE The Delegated Administrator for Messaging only supports Sun Java System LDAP Schema v. 1, not v.2

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 739.
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 Messaging Server lets you specify an automatic response to incoming mail for a user. You can specify two different auto-reply modes: vacation mode and auto-reply mode.

The 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.

NOTE The Delegated Administrator for Messaging only supports Sun Java System LDAP Schema v. 1, not v.2

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 739.
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.

   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 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 739.

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 the chapter on User and Group Administration, of the *Sun ONE Server Console 5.2 Server Management Guide*.)

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 the chapter on User and Group Administration, of the *Sun ONE Server Console 5.2 Server Management Guide.*

   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 748.
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 the chapter on User and Group Administration, of the *Sun ONE Server Console 5.2 Server Management Guide.*

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 the chapter on User and Group Administration of the *Sun ONE Server Console 5.2*

Server Management Guide), might have additional privileges or responsibilities beyond those of mailing-list members. For more information on groups, see the chapter on User and Group Administration of the *Sun ONE Server Console 5.2 Server Management Guide.*

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 the chapter on User and Group Administration of the *Sun ONE Server Console 5.2 Server Management Guide.* See also the Sun Java System 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 C-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 748.
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 753.
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 748.
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 the chapter on User and Group Administration, of the *Sun ONE Server Console 5.2 Server Management Guide*.

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 748.
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 the chapter on User and Group Administration, of the *Sun ONE Server Console 5.2 Server Management Guide.*
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.

Appendix D

Short Message Service (SMS)

This chapter describes how to implement the Short Message Service (SMS) on the Sun™ ONE Messaging Server. It covers the following topics:

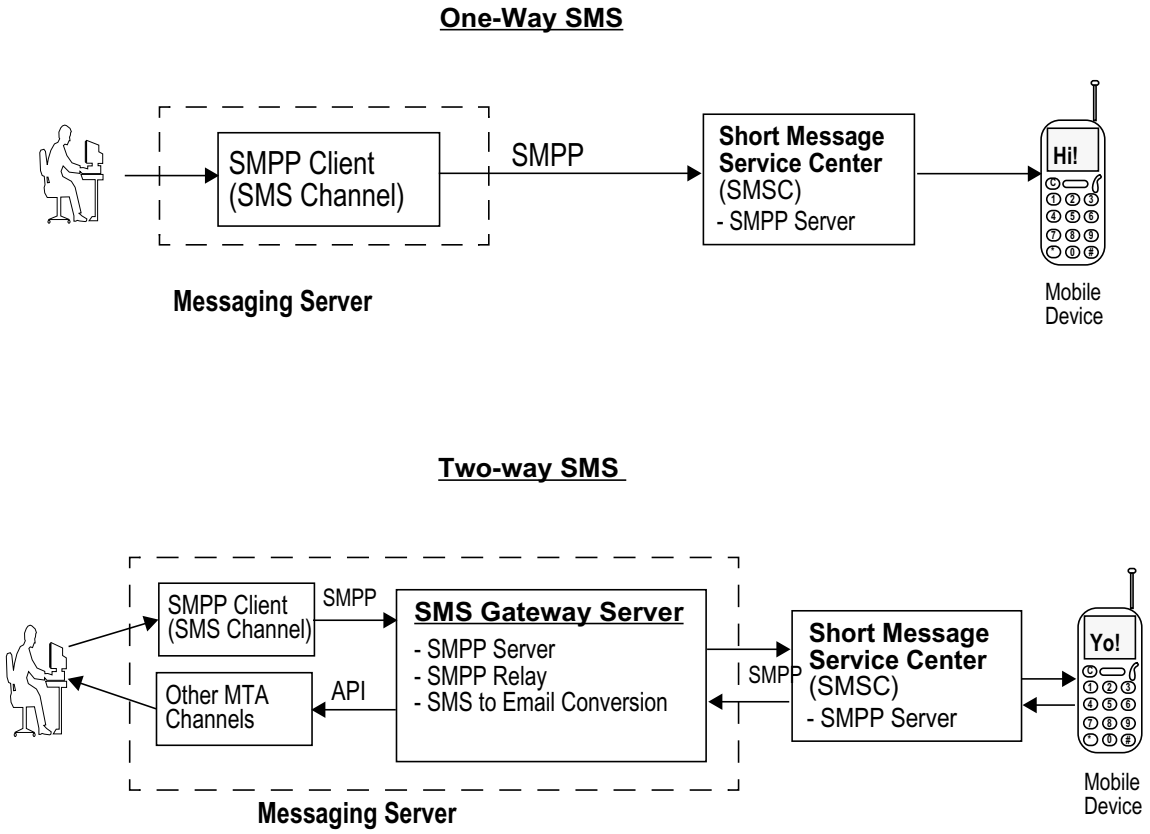
Introduction

Sun Java System Messaging Server implements email-to-mobile and mobile-to-email messaging using a Short Message Service (SMS). SMS can be configured to be either one-way (email-to-mobile only) or two-way (both email-to-mobile and mobile-to-email). To enable one-way service only, you must add and configure the SMS channel. To enable two-way service, you must add and configure the SMS channel, and in addition, configure the SMS Gateway Server.

For both one- and two-way SMS, the generated SMS messages are submitted to a Short Message Service Center (SMSC) using the Short Message Peer to Peer (SMPP) protocol. Specifically, the SMSC must provide a V3.4 or later SMPP server that supports TCP/IP.

Figure D-1 illustrates the logical flow of messages for both one-way and two-way SMS.

Figure D-1 Logical Flow For One-Way and Two-Way SMS



One-Way SMS

To enable one-way service, the Messaging Server implements an SMPP client (the MTA SMS channel) that communicates with remote SMSCs. The SMS channel converts enqueued email messages to SMS messages as described in "The Email to SMS Conversion Process" on page 764. This conversion process includes handling of multipart MIME messages as well as character set translation issues.

Operating in this capacity, the SMS channel functions as an (SMPP) External Short Message Entity (ESME).

Two-Way SMS

Two-Way SMS enables the mail server not only to send email to remote devices, but allows for receiving replies from the remote devices and for remote device email origination.

Enabling two-way SMS service requires both the MTA SMS channel (SMPP client), as explained in the previous topic, and the SMS Gateway Server. Sun Java System Messaging Server installs an SMS Gateway Server as part of its general installation process, which you must then configure. The SMS Gateway Server performs two functions:

- SMPP relay

  The SMS Gateway Server acts as a transparent SMPP client between the MTA SMS channel and SMSCs. However, in addition, while acting as a relay, the SMS Gateway Server generates unique SMS source addresses for relayed messages, and saves the message IDs returned by the remote SMSCs for later correlation with SMS notification messages.

- SMPP server

  The SMS Gateway Server acts as an SMPP server to receive mobile originated SMS messages, replies to prior email messages, and SMS notifications. The SMS Gateway Server extracts destination email addresses from the SMS messages using profiles that define the conversion process. Profiles also describe how to handle notification messages returned by remote SMSCs in response to previously sent email-to-mobile messages.

NOTE Sun Java System Messaging Server does not support the two-way SMS on the Windows platform.

Requirements

This manual assumes that you have read LogicaCMG's SMPP specification, and the SMPP documentation for your SMSC.

In order to implement SMS, you must have the following:

- Sun Java System Messaging Server 6 or greater. (One-way SMS is also implemented in iPlanet Messaging Server 5.2.)
- The SMSC must support SMPP V3.4, or later, over TCP/IP and there must be TCP/IP connectivity between the host running Messaging Server and the SMSC.

For storage planning information for the SMS Gateway Server, see “SMS Gateway Server Storage Requirements” on page 836.

SMS Channel Theory of Operation

The SMS channel is a multi-threaded channel which converts queued email messages to SMS messages and then submits them for delivery to an SMSC.

The following channel operation topics are covered in this section:

Directing Email to the Channel

When the SMS channel is configured as per “SMS Channel Configuration” on page 778, one or more host names will be associated with the channel. For purposes of discussion, let us assume that the host name `sms.siroe.com` is a host name associated with the channel. In that case, email is directed to the channel with an address of the form:

```
local-part@sms.siroe.com
```

in which `local-part` is either the SMS destination address (for example, a wireless phone number, pager ID, etc.) or an attribute-value pair list in the format:

```
/attribute1=value1/attribute2=value2/.../@sms.siroe.com
```

The recognized attribute names and their usages are given in Table D-1. These attributes allow for per-recipient control over some channel options.

Table D-1 SMS Attributes

Attribute Name	Attribute Value and Usage
`ID`	SMS destination address (for example, wireless phone number, pager ID, etc.) to direct the SMS message to. This attribute and associated value must be present.
`FROM`	SMS source address. Ignored when option `USE_HEADER_FROM=0`.

Table D-1 SMS Attributes

Attribute Name	Attribute Value and Usage
FROM_NPI	NPI. Use the specified NPI value. Ignored when option USE_HEADER_FROM=0.
FROM_TON	TON. Use the specified TON value. Ignored when option USE_HEADER_FROM=0.
MAXLEN	The maximum, total bytes (that is, eight bit bytes) to place into the generated SMS message or messages for this recipient. The lower value of either MAXLEN and the value specified by the MAX_MESSAGE_SIZE channel option is used.
MAXPAGES	The maximum number of SMS messages to split the email message into for this recipient. The lower value of either MAXPAGES and the value specified by the MAX_PAGES_PER_MESSAGE channel option is used.
NPI	Specify a Numeric Plan Indicator (NPI) value for the destination SMS address specified with the ID attribute. See the description of the DEFAULT_DESTINATION_NPI channel option for information on the accepted values for this attribute. When this attribute is used, its value overrides the value given by the DEFAULT_DESTINATION_NPI channel option.
PAGELEN	Maximum number of bytes to place into a single SMS message for this recipient. The minimum of this value and that specified with the MAX_PAGE_SIZE channel option is used.
TO	Synonym for ID.
TO_NPI	Synonym for NPI.
TO_TON	Synonym for TON.
TON	Specify a Type of Number (TON) value for the destination SMS address given with the ID attribute. See the description of the DEFAULT_DESTINATION_TON channel option for information on the accepted values for this attribute. When this attribute is used, its value overrides the value given by the DEFAULT_DESTINATION_TON channel option.

Some example addresses:

```
123456@sms.siroe.com
/id=123456/@sms.siroe.com
/id=123456/maxlen=100/@sms.siroe.com
/id=123456/maxpages=1/@sms.siroe.com
```

For information on performing translations, validity checks, and other operations on the SMS destination address portion of the email address, see “Site-defined Address Validity Checks and Translations” on page 772.

The Email to SMS Conversion Process

In order for email to be sent to a remote site, email must be converted to SMS messages that can be understood by the remote SMSCs. This section describes the process of converting an email message queued to the SMS channel to one or more SMS messages. As described below, options allow control over the maximum number of SMS messages generated, the maximum total length of those SMS messages, and the maximum size of any single SMS message. Only text parts (that is, MIME text content types) from the email message are used and the maximum number of parts converted may also be controlled.

Character sets used in the email message's header lines and text parts are all converted to Unicode and then converted to an appropriate SMS character set.

When there is no `SMS_TEXT` mapping table (see "Site-defined Text Conversions" on page 773), an email message queued to the SMS channel receives the processing illustrated in Figure D-2.

Figure D-2 SMS Channel Email Processing

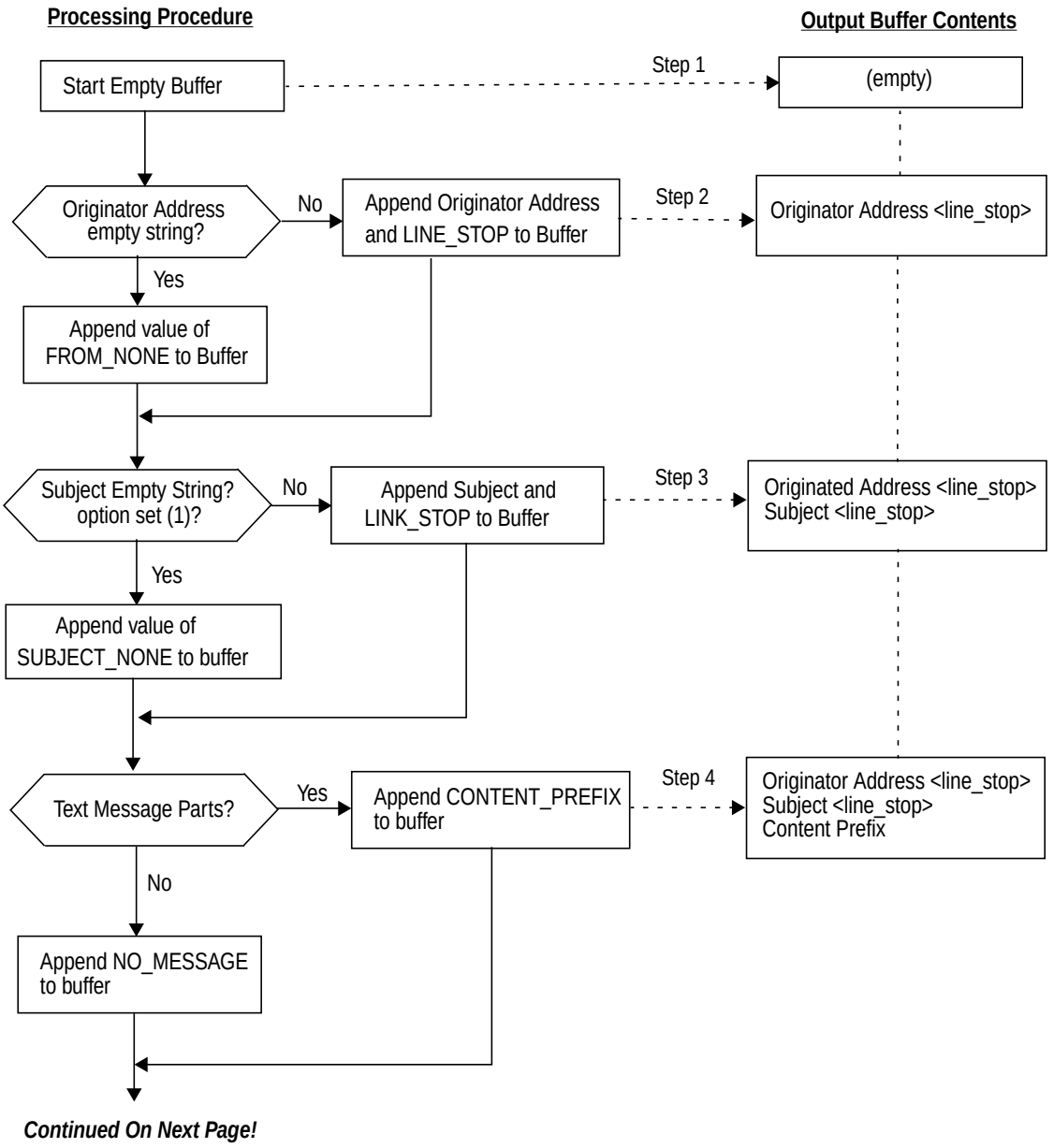
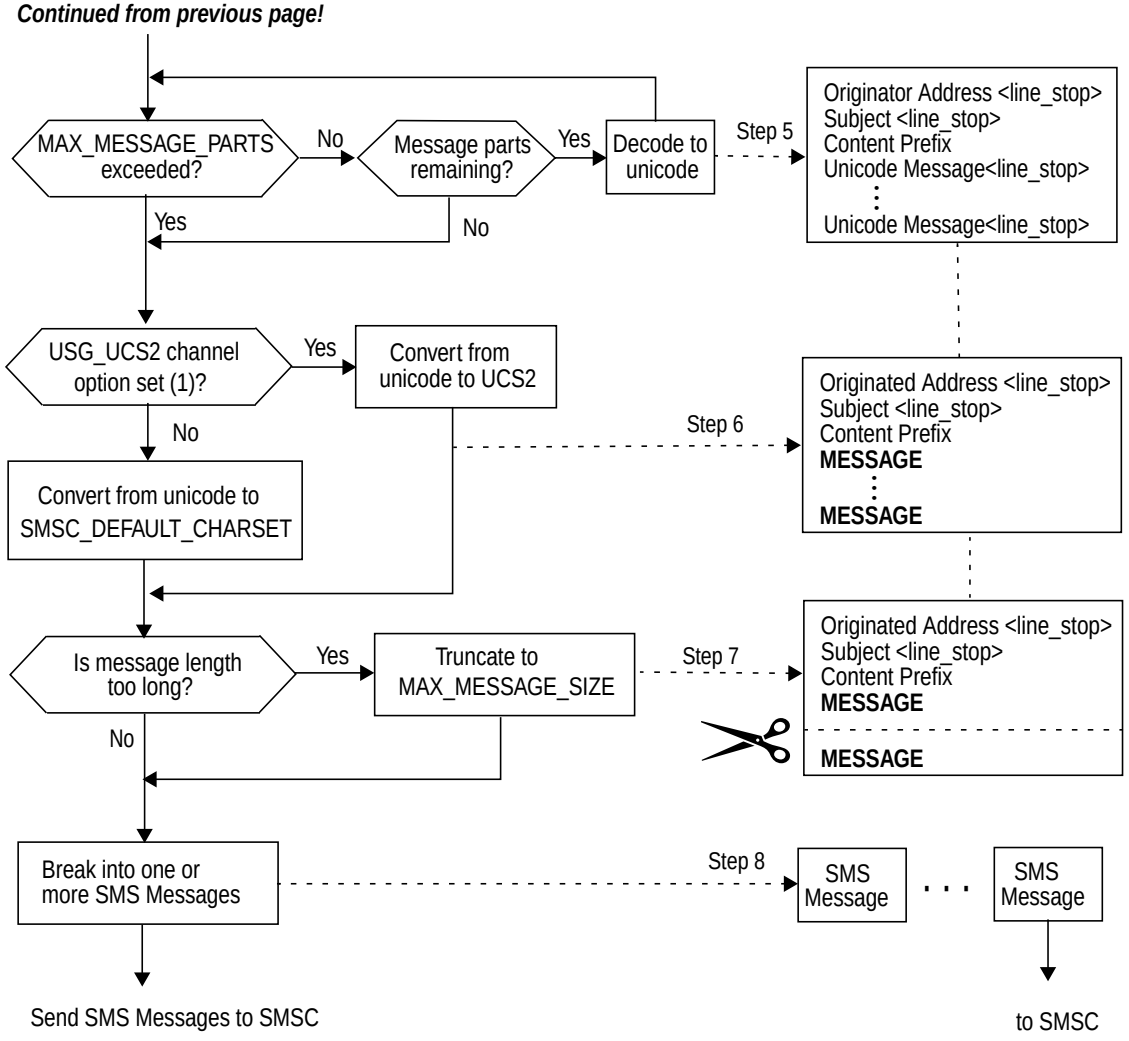


Figure D-3 SMS Channel Email Processing (*continued*)



The following steps correspond to the numbered boxes in Figure D-2:

1. An empty output buffer is started. The character set used for the buffer is Unicode.

2. The email message's originator address is taken from one of the following five sources, shown in decreasing order of preference:

   1. `Resent-from:`
   2. `From:`
   3. `Resent-sender:`
   4. `Sender:`
   5. `Envelope From:`

   If the originator address is an empty string, then the value of the `FROM_NONE` channel option is instead appended to the buffer.

   If, however, the originator address is a non-empty string, then the result of processing the `FROM_FORMAT` channel option, and the value of the `LINE_STOP` channel option are appended to the output buffer.

   Note that the `Resent-from:` and `Resent-sender:` header lines are only considered if the `USE_HEADER_RESENT` option has the value 1. Otherwise, `Resent-` header lines are ignored.

3. If a `Subject:` header line is not present or is empty, then the value of the `SUBJECT_NONE` option is appended to the output buffer.

   Otherwise, the result of processing the `SUBJECT_FORMAT` option, and the value of the `LINE_STOP` channel option are appended to the output buffer.

4. If there are no text message parts, then the value of the `NO_MESSAGE` channel option is appended to the output buffer.

   If there are text message parts, then the value of the `CONTENT_PREFIX` channel option is appended to the output buffer.

   Non-text message parts are discarded.

5. For each text part, while the `MAX_MESSAGE_PARTS` limit has not been reached, the text part is decoded to Unicode and appended to the buffer, along with the value of the `LINE_STOP` channel option.

6. The resulting output buffer is then converted from Unicode to either the SMSC's default character set or UCS2 (UTF-16). The SMSC's default character set is specified with the `SMSC_DEFAULT_CHARSET` option.

7. After being converted, it is then truncated to not exceed `MAX_MESSAGE_SIZE` bytes.

8. The converted string from Step 6 is then broken into one or more SMS messages, no single SMS message longer than MAX_PAGE_SIZE bytes. At most, MAX_PAGES_PER_MESSAGE SMS messages will be generated.

NOTE As an email message may have multiple recipients, Step 6 through Step 8 may need to be done for each recipient address which makes use of the MAXLEN, MAXPAGES, or PAGELEN attributes described in "Directing Email to the Channel," on page 4.

Sample Email Message Processing

For example, with the channel's default settings, the email message:

```
From: John Doe
To: 1234567@sms.siroe.com
Subject: Today's meeting
Date: Fri, 26 March 2001 08:17

The staff meeting is at 14:30 today in the big conference room.
```

Would be converted to the SMS message:

```
jdoe@siroe.com (Today's meeting) The staff meeting is at 14:30 today in the big
conference room.
```

A different set of option settings, that follows:

```
CONTENT_PREFIX=Msg:
FROM_FORMAT=From:${pa}
SUBJECT_FORMAT=Subj:$s
```

would instead produce:

```
From:John Doe Subj:Today's meeting Msg:The staff meeting is at 14:30 today in
the big conference room.
```

The SMS Message Submission Process

Once an email message has been converted to one or more SMS messages, with possibly different sets for each recipient, the SMS messages are then submitted to the destination SMSC. The submissions are effected using SMPP V3.4 over TCP/IP. The hostname (SMPP_SERVER) of the SMPP server is taken to be the official host name associated with the SMS channel; the TCP port (SMPP_PORT) to use is specified with the port channel keyword.

When there are messages to process, the channel is started. The channel binds to the SMPP server as a transmitter, presenting the credentials specified with the ESME_ channel options described in"SMPP Options" on page 796. Table D-2 lists the fields set in a BIND_TRANSMITTER PDU (Protocol Data Unit), and gives their values:

Table D-2 Fields in Generated in a BIND_TRANSMITTER PDU

Field	Value
system_id	ESME_SYSTEM_ID channel option; default value is an empty string
password	ESME_PASSWORD channel option; default value is an empty string
system_type	ESME_SYSTEM_TYPE channel option; default value is an empty string
interface_version	0x34 indicating SMPP V3.4
addr_ton	ESME_ADDRESS_TON; default value is 0x00 indicating an unknown TON
addr_npi	ESME_ADDRESS_NPI; default value is 0x00 indicating an unknown NPI
addr_range	ESME_IP_ADDRESS channel option; default value is an empty string

Note that the channel is multithreaded. Depending on how much mail there is to send, the channel may have multiple dequeue thread running. (There can even be multiple channel processes running.) Each thread does a BIND_TRANSMITTER and then on that TCP/IP connection, sends all of the SMS messages it has to send, and then sends an UNBIND, and then closes the connection. No attempt is made to hold a connection open for a period of idle time for potential reuse. If the remote SMPP server sends back a throttle error, then an UNBIND is issued, the TCP/IP connection is closed, and a new connection and BIND established.It behaves similarly if the remote SMPP server sends an UNBIND before it is finished sending its SMS messages.

The SMS messages are then submitted using SMPP SUBMIT_SM PDUs. If a permanent error is returned (for example, ESME_RINVDSTADR), then the email message is returned as undeliverable. If a temporary error is returned, then the email message is re-enqueued for a later delivery attempt. To clarify, a permanent error is one for which the condition is likely to exist indefinitely and for which repeated delivery attempts will have no positive effect, such as invalid SMS destination addresses. Whereas, a temporary error is one for which the condition is likely to not exist in the near future, such as a server down or server congested condition.

If the USE_HEADER_FROM option has the value 1, then the source address for the submitted SMS message is set. The value used is derived from the originating email message and is chosen to be the most likely (email) address to which any replies should be directed. Accordingly, the source address taken from one of the following seven sources, shown in decreasing order of preference:

1. Resent-reply-to:
2. Resent-from:
3. Reply-to:
4. From:
5. Resent-sender:
6. Sender:
7. Envelope From:

Note that the Resent-reply-to: and Reply-to: header lines are only considered if the USE_HEADER_REPLY_TO option has the value 1. Moreover, the Resent-reply-to:, Resent-from:, and Resent-sender: header lines are only considered if the USE_HEADER_RESENT option has the value 1. (Note that this means that both of those options must have the value 1 for the Resent-reply-to: header line to be considered.) The default value for both of these options is the value 0. As such, only items 4, 6, and 7 are considered by the default configuration. Finally, since the source address in an SMS message is limited to 20 bytes, the source address chosen will be truncated if it exceeds that limit.

Table D-3 shows the mandatory fields set in a SUBMIT_SM PDU:

Table D-3 Mandatory Fields in Generated SUBMIT_SM PDUs

Field	Value
service_type	DEFAULT_SERVICE_TYPE channel option; default value is an empty string.
source_addr_ton	DEFAULT_SOURCE_TON channel option; if USE_HEADER_FROM=1, then this field is usually forced to the value 0x05 indicating an alphanumeric TON; otherwise, the default value is 0x01 indicating an international TON.
source_addr_npi	DEFAULT_SOURCE_NPI channel option; default value is 0x00.
source_addr	DEFAULT_SOURCE_ADDRESS channel option if USE_HEADER_FROM=0; otherwise, an alphanumeric string representing the originator of the email message.
dest_addr_ton	TON addressing attribute or DEFAULT_DESTINATION_TON channel option; default value is 0x01 indicating an international TON.
dest_addr_npi	NPI addressing attribute or DEFAULT_SOURCE_NPI channel option; default value is 0x00 indicating an unknown NPI.
dest_addr	Destination SMS address derived from the local part of the email envelope To: address; see “Directing Email to the Channel” on page 762.
esm_class	For one-way SMS, set to 0x03, indicating store and forward mode, default SMSC message type, and do not set reply path. For a two-way MSM message, set to 0x83.
protocol_id	0x00; unused for CDMA and TDMA; for GSM, 0x00 indicates no Internet, but SME-to-SME protocol.
priority_flag	0x00 for GSM & CDMA and 0x01 for TDMA, all indicating normal priority; See the description of the DEFAULT_PRIORITY channel option.

Table D-3 Mandatory Fields in Generated SUBMIT_SM PDUs

Field	Value
schedule_delivery_time	Empty string indicating immediate delivery.
validity_period	DEFAULT_VALIDITY_PERIOD channel option; default value is an empty string indicating that the SMSC's default should be used.
registered_delivery	0x00 indicating no registered delivery.
replace_if_present_flag	0x00 indicating that any previous SMS messages should not be replaced.
data_coding	0x00 for the SMSC's default character set; 0x08 for the UCS2 character set.
sm_default_msg_id	0x00 indicating not to use a pre-defined message.
sm_length	Length and content of the SMS message; see "The Email to SMS Conversion Process" on page 764 for details.
short_message	Length and content of the SMS message; see "The Email to SMS Conversion Process" on page 764 for details.

Table D-4 shows the optional fields in a SUBMIT_SM PDU:

Table D-4 Optional Fields in Generated SUBMIT_SM PDUs

Field	Value
privacy	See the description of the DEFAULT_PRIVACY channel keyword; default is to not provide this field unless the email message has a Sensitivity: header line
sar_refnum	See the description of the USE_SAR channel keyword; default is to not provide these fields
sar_total	See sar_refnum above.
sar_seqnum	See sar_refnum above.

The channel remains bound to the SMPP server until either it has no more SMS messages to submit (the message queue is empty), or MAX_PAGES_PER_BIND has been exceeded. In the latter case, a new connection is made and bind operation performed if there remain further SMS messages to send.

Note that the SMS channel is multithreaded. Each processing thread in the channel maintains its own TCP connection with the SMPP server. For example, if there are three processing threads each with SMS messages to submit, then the channel will have three open TCP connections to the SMPP server. Each connection will bind to

the SMPP server as a transmitter. Moreover, any given processing thread will only have one outstanding SMS submission at a time. That is, a given thread will submit an SMS message, then wait for the submission response (that is, SUBMIT_SM_RESP PDU) before submitting another SMS message.

Site-defined Address Validity Checks and Translations

Sites may wish to apply validity checks or translations to SMS destination addresses encoded in the recipient email addresses described in "Directing Email to the Channel" on page 762. For example, sites may wish to:

- Strip non-numeric characters (for example, translating 800.555.1212 to 8005551212)
- Prepend a prefix (for example, translating 8005551212 to +18005551212)
- Validate for correctness (for example, 123 is too short)

The first two tasks can be done specifically with the DESTINATION_ADDRESS_NUMERIC and DESTINATION_ADDRESS_PREFIX channel options. In general, all three of these tasks, and others can be implemented using mapping tables: either mapping table callouts in the rewrite rules or by means of a FORWARD mapping table. Using a mapping table callout in the rewrite rules will afford the most flexibility, including the ability to reject the address with a site-defined error response. The remainder of this section will focus on just such an approach -- using a mapping table callout from the rewrite rules.

Let us suppose that destination addresses need to be numeric only, be 10 or 11 digits long, and be prefixed with the string "+1". This can be accomplished with the following rewrite rule

```
sms.siroe.com     ${X-REWRITE-SMS-ADDRESS,$U}@sms.siroe.com
sms.siroe.com     $?Invalid SMS address
```

The first rewrite rule above calls out to the site-define mapping table named X-REWRITE-SMS-ADDRESS. That mapping table is passed the local part of the email address for inspection. If the mapping process decides that the local part is acceptable, then the address is accepted and rewritten to the SMS channel. If the mapping process does not accept the local part, then the next rewrite rule is applied. Since it is a $? rewrite rule, the address is rejected with the error text "Invalid SMS address".

The X-REWRITE-SMS-ADDRESS mapping table is shown below. It performs the necessary validation steps for local parts in either attribute-value pair list format or just a raw SMS destination address.

```
X-VALIDATE-SMS-ADDRESS

! Iteratively strip any non-numeric characters
    $_*$[$ -/:-~]%*  $0$2$R
! Accept the address if it is of the form 1nnnnnnnnnn or nnnnnnnnnn
! In accepting it, ensure that we output +1nnnnnnnnnn
   1%%%%%%%%%%      +1$0$1$2$3$4$5$6$7$8$9$Y
   %%%%%%%%%%       +1$0$1$2$3$4$5$6$7$8$9$Y
! We didn't accept it and consequently it's invalid
   *                 $N

X-REWRITE-SMS-ADDRESS

    */id=$_*/*       $C$0/id=$|X-VALIDATE-SMS-ADDRESS;$1|/$2$Y$E
    */id=$_*/*       $N
    *                $C$|X-VALIDATE-SMS-ADDRESS;$0|$Y$E
    *                $N
```

With the above set up, be sure that DESTINATION_ADDRESS_NUMERIC option has the value 0 (the default). Otherwise, the "+" will be stripped from the SMS destination address.

Site-defined Text Conversions

Sites may customize Steps 1 - 6 described in "The Email to SMS Conversion Process" on page 764 with a table of conversion rules. These rules are specified via a mapping table in the MTA's mapping file.

The name of the mapping table should be *SMS_Channel*_TEXT where *SMS_Channel* is the name of the SMS channel; for example, SMS_TEXT if the channel is named sms or SMS_MWAY_TEXT if the channel is named sms_mway.

Two types of entries may be made in this mapping table. However, before explaining the format of those entries, let it be made clear that an understanding of how to use the mapping file is essential in order to understand how to construct and use these entries. An example mapping table is given after the description of these two types of entries.

Now, the two types of entries are:

- Message Header Entries
- Message Body Entries

Message Header Entries

These entries specify which message header lines should be included in an SMS message and how they should be abbreviated or otherwise converted. Only if a header line is successfully mapped to a string of non-zero length by one of these entries will it be included in the SMS message being generated. Each entry has the format

H|*pattern* *replacement-text*

If a message header line matches the pattern then it will be replaced with the replacement text replacement-text using the mapping file's pattern matching and string substitution facilities. The final result of mapping the header line will then be included in the SMS message provided that the metacharacter $Y was specified in the replacement text. If a header line does not match any pattern string, if it maps to a string of length zero, or if the $Y metacharacter is not specified in the replacement text, then the header line will be omitted from the SMS message. The two entries

```
H|From:*  F:$0$Y
H|Subject:*   S:$0$Y
```

cause the `From:` and `Subject:` header lines to be included in SMS messages with `From:` and `Subject:` abbreviated as `F:` and `S:`. The entries:

```
H|Date:*  H|D:$0$R$Y
H|D:*,*%19%%*:*:*  H|D:$0$ $5:$6$R$Y
```

cause the `Date:` header line to be accepted and mapped such that, for instance, the header line

```
Date: Wed, 16 Dec 1992 16:13:27 -0700 (PDT)
```

will be converted to

```
D: Wed 16:13
```

Very complicated, iterative mappings may be built. Sites wishing to set up custom filters will first need to understand how the mapping file works. The H| in the right-hand-side of the entry may be omitted, if desired. The H| is allowed in that side so as to cut down on the number of table entries required by sets of iterative mappings.

Message Body Entries

These entries establish mappings to be applied to each line of the message body. Each line of the message body will be passed through these mappings before being incorporated into the SMS message being built. These entries take the form:

B|*pattern* B|*replacement-text*

If a line of the message body matches a *pattern* pattern then it will be replaced with the replacement text *replacement-text*. Again, very complicated, iterative mappings may be constructed using this facility. The B| in the right-hand-side of the entry may be omitted, if desired.

Example SMS Mapping Table

An example SMS_TEXT mapping table is shown in Code Example D-1. The numbers inside parentheses at the end of each line correspond to the item numbers in the section titled “Explanatory Text” that follows this table.

Code Example D-1 Example SMS_TEXT Mapping Table.

```
SMS_TEXT

    H|From:*          H|F:$0$R$Y  (1.)
    H|Subject:*       H|S:$0$R$Y  (1.)
    H|F:*<*>*         H|F:$1$R$Y  ()
    H|F:*(*)*         H|F:$0$2$R$Y  (2.)
    H|F:*"*"*         H|F:$0$2$R$Y  (3.)
    H|F:*@*           H|F:$0$R$Y  (4.)
    H|%:$ *           H|$0:$1$R$Y  (5.)
    H|%:*$            H|$0:$1$R$Y  (5.)
    H|%:*$ $ *        H|$0:$1$ $2$R$Y  (6.)
    B|*--*            B|$0-$1$R  (7.)
    B|*..*            B|$0.$1$R  (7.)
    B|*!!*            B|$0!$1$R  (7.)
    B|*??*            B|$0?$1$R  (7.)
    B|*$ $ *          B|$0$ $1$R  (6.)
    B|$ *             B|$0$R  (5.)
    B|*$              B|$0$R  (5.)
```

Explanatory Text

The entries in the example SMS_TEXT mapping table above are explained below:

In the example above, the metacharacter $R is used to implement and control iterative application of the mappings. By iterating on these mappings, powerful filtering is achieved. For instance, the simple mappings to remove a single leading or trailing space (6) or reduce two spaces to a single space (7) become, when taken as a whole, a filter which strips all leading and trailing spaces and reduces all consecutive multiple spaces to a single space. Such filtering helps reduce the size of each SMS message.

1. These two entries cause `From:` and `Subject:` header lines to be included in an SMS message. `From:` and `Subject:` are abbreviated as, respectively, `F:` and `S:`. Some of the other entries may have further effects on `From:` and `Subject:` header lines.

   This entry will reduce a `From:` header line containing a <...> pattern to only the text within the angle brackets. For example:

   ```
   F: "John C. Doe" <jdoe@siroe.com> (Hello)
   ```

   will be replaced with:

   ```
   F: jdoe@siroe.com
   ```

2. This entry will remove, inclusively, everything inside of a (...) pattern in a `From:` header line. For example:

   ```
   F: "John C. Doe" <jdoe@siroe.com> (Hello)
   ```

   will be replaced with:

   ```
   F: "John C. Doe" <jdoe@siroe.com>
   ```

3. This entry will remove, inclusively, everything inside of a "..." pattern in a `From:` header line. For example:

   ```
   F: "John C. Doe" <jdoe@siroe.com> (Hello)
   ```

   will be replaced with:

   ```
   F: <jdoe@siroe.com> (Hello)
   ```

4. This entry will remove, inclusively, everything to the right of an at-sign, @, in a `From:` header line. For example:

   ```
   F: "John C. Doe" <jdoe@siroe.com> (Hello)
   ```

   will be replaced with:

   ```
   F: "John C. Doe" <jdoe@
   ```

5. These four entries remove leading and trailing spaces from lines in the message header and body.

6. These two entries reduce two spaces to a single space in lines of the message header and body.

7. These four entries reduce double dashes, periods, exclamation and question marks to single occurrences of the matching character. Again, this helps save bytes in an SMS message.

The order of the entries is very important. For instance, with the given ordering, the body of the message `From:` header line:

```
From: "John C. Doe"  (Hello)
```

will be reduced to:

```
jdoe
```

The steps taken to arrive at this are as follows:

1. We begin with the From: header line:

   ```
   From: "John C. Doe"  (Hello)
   ```

   The pattern in the first mapping entry matches this and produces the result:

   ```
   F: "John C. Doe"  (Hello)
   ```

   The $R metacharacter in the result string causes the result string to be remapped.

2. The mapping is applied to the result string of the last step. This produces:

   ```
   F: jdoe@siroe.com
   ```

   The $R in the mapping causes the entire set of mappings to be re-applied to the result of this step.

3. Next, the mapping is applied producing:

   ```
   F: jdoe
   ```

   The $R in the mapping causes the entire set of mappings to be re-applied to the result of this step.

4. Next, the mapping is applied producing:

   ```
   F:jdoe
   ```

   The $R in the mapping causes the entire set of mappings to be re-applied to the result of this step.

5. Since no other entries match, the final result string:

   ```
   F:jdoe
   ```

is incorporated into the SMS message.

NOTE The imsimta test-mapping utility may be used to test a mapping table. For instance,

```
# imsimta test -mapping -noimage_file -mapping_file=test.txt
Enter table name: SMS_TEXT
Input string: H|From: "John C. Doe"  (Hello)
Output string: H|F:jdoe
Output flags: [0,1,2,89]
Input string: ^D
#
```

For further details on the imsimta utility, see the *Sun Java System Messaging Server Administration Reference*.

SMS Channel Configuration

This section gives directions on how to set up the SMS channel for both one-way (email-to-mobile) and two-way (email-to-mobile and mobile-to-email) functionality. The SMS channel is set up the same for both one-way and two-way functionality, with the exceptions noted in the topic "Configuring the SMS Channel for Two-Way SMS" on page 807.

This section includes the following topics:

Adding an SMS Channel

Two steps are required to add an SMS channel to a Messaging Server configuration:

1. "Adding the Channel Definition and Rewrite Rules" on page 779.
2. "Creating an SMS Channel Option File" on page 781.

While there are no channel options which must be set in all situations, it is likely that one or more of the following options may need to be set: ESME_PASSWORD, ESME_SYSTEM_ID, MAX_PAGE_SIZE, DEFAULT_SOURCE_TON, and DEFAULT_DESTINATION_TON. And, as described, the SMPP server's hostname or IP address and TCP port must be set either through the channel definition in the imta.cnf file or the channel option file.

You may configure more than one SMS channel, giving different characteristics to different SMS channels. See "Adding Additional SMS Channels" on page 803 for further information on the use of multiple SMS channels.

Note for the instructions that follow: if you change the imta.cnf file you must recompile. if you change just a channel option file, there is no need to recompile.

Note also that the time before a channel change takes effect can differ depending on what the change is. Many channel option changes take effect in all channels started since the change was made, which may seem almost instantaneous since the Job Controller is often starting new channels. Some of the changes don't take effect until you recompile and restart the SMTP server. These options are processed as a message is enqueued to the channel and not when the channel itself runs.

Adding the Channel Definition and Rewrite Rules

To add the channel definition and rewrite rules, do the following:

1. Before adding an SMS channel to the MTA's configuration, you need to pick a name for the channel. The name of the channel may be either sms or sms_x where x is any case-insensitive string whose length is between one and thirty-six bytes. For example, sms_mway.
2. To add the channel definition, edit the imta.cnf file located in the *installation-directory*/config/ directory. At the bottom of the file add a blank line followed by the two lines:

   ```
   channel-name port p threaddepth t \
     backoff pt2m pt5m pt10m pt30m notices 1
   smpp-host-name
   ```

   where *channel-name* is the name you chose for the channel, *p* is the TCP port the SMPP server listens on, *t* is the maximum simultaneous number of SMPP server connections per delivery process, and *smpp-host-name* is the host name of the system running the SMPP server.

For example, you might specify a channel definition as follows:

```
sms_mway port 55555 threaddepth 20 \
  backoff pt2m pt5m pt10m pt30m notices 1
smpp.siroe.com
```

For instructions on how to calculate threaddepth, see "Controlling the Number of Simultaneous Connections" on page 781.

See "Adjusting the Frequency of Delivery Retries" on page 804 for a discussion of the backoff and notices channel keywords.

If you wish to specify an IP address rather than a host name, for *smpp-host-name*, specify a domain literal. For example, if the IP address is 127.0.0.1, then specify [127.0.0.1] for *smpp-host-name*. Alternatively, consider using the SMPP_SERVER channel option.

NOTE For Sun Java System Messaging Server 6.1, the use of the master channel keyword has been deprecated. It is ignored if present.

3. Once the channel definition has been added, go to the top half of the file and add a rewrite rule in this format:

   smpp-host-name $u@*smpp-host-name*

   For example,

   smpp.siroe.com $u@smpp.siroe.com

4. Save the imta.cnf file.
5. Recompile the configuration with the imsimta cnbuild command.
6. Restart the SMTP server with the imsimta restart dispatcher command.
7. With the above configuration, email messages are directed to the channel by addressing them to id@smpp-host-name (for example, 123456@smpp.siroe.com). See "The Email to SMS Conversion Process" on page 764 for further information on addressing.

8. Optionally, if you wish to hide the SMPP server's host name from users or associate other host names with the same channel, then add additional rewrite rules. For instance, to associate host-name-1 and host-name-2 with the channel, add the following to rewrite rules:

   ```
   host-name-1 $U%host-name-1@smpp-host-name
   host-name-2 $U%host-name-2@smpp-host-name
   ```

   For example, if the SMPP server's host name is smpp.siroe.com but you want users to address email to id@sms.sesta.com, then add the rewrite rule:

   ```
   sms.sesta.com $U%sms.sesta.com@smpp.siroe.com
   ```

Note that the SMPP_SERVER and SMPP_PORT channel options will override the channel's official host name and port channel keyword settings. When the SMPP_PORT option is used, it is not necessary to also use the port keyword. The advantage of using these two options is that they can be put into effect and subsequently changed without needing to recompile the configuration. An additional use of the SMPP_SERVER option is described in"Adding Additional SMS Channels" on page 803.

Controlling the Number of Simultaneous Connections

The threaddepth channel keyword controls the number of messages to assign to each delivery thread within a delivery process. To calculate the total number of concurrent connections allowed, multiply the values of the two following options: SMPP_MAX_CONNECTIONS and job_limit (SMPP_MAX_CONNECTIONS * job_limit). The SMPP_MAX_CONNECTIONS option controls the maximum number of delivery threads in a delivery process. And, the job_limit option, for the Job Controller processing pool in which the channel is run, controls the maximum number of simultaneous delivery processes.

To limit the total number of concurrent connections, you must adjust appropriately either or both of these options. For instance, if the remote SMPP server allows only a single connection, then both SMPP_MAX_CONNECTIONS and job_limit must be set to 1. When adjusting the values, it's preferable to allow job_limit to exceed 1.

Creating an SMS Channel Option File

In general, a channel option file contains site-specific parameters required for the operation of the channel. A channel option file is not required for SMS. If you determine that one is necessary for your installation, store it in a text file in the *installation-directory*/config/ directory. As with other channel option files, the name of the file takes the form:

*channel_name*_option

For instance, if the channel is named `sms_mway` then the channel option file is:

`installation-directory``/config/sms_mway_option`

Each option is placed on a single line in the file using the format:

`option_name=`*`option_value`*

For example,

```
PROFILE=GSM
SMSC_DEFAULT_CHARSET=iso-8859-1
USE_UCS2=1
```

For a list of available SMS channel options and a description of each, see "Available Options"" that follows.

Available Options

The SMS channel contains a number of options which divide into six broad categories:

- *Email to SMS conversion*: Options which control the email to SMS conversion process.
- *SMS Gateway Server Option:* Gateway profile option.
- *SMS fields*: Options which control SMS-specific fields in generated SMS messages.
- *SMPP protocol*: Options associated with the use of the SMPP protocol over TCP/IP.
- *Localization*: Options which allow for localization of text fields inserted into SMS messages.
- *Miscellaneous*: Debug option.

These options are summarized in the table below, and described more fully in the sections which follow.

Table D-5 SMS Channel Options

Email to SMS Conversion Options

Table D-5 SMS Channel Options *(Continued)*

Option (Page number)	Description	Default
GATEWAY_NOTIFICATIONS	Specify whether or not to convert email notification messages to SMS messages.	0
MAX_MESSAGE_PARTS	Max. number of message parts to extract from an email message	2
MAX_MESSAGE_SIZE	Maximum number of bytes to extract from an email message	960
MAX_PAGE_SIZE	Maximum number of bytes to put into a single SMS message	160
MAX_PAGES_PER_MESSAGE	Max. number of SMS messages to break an email message into	6
ROUTE_TO	Route SMS messages to the specified IP host name.	
SMSC_DEFAULT_CHARSET	The default character set used by the SMSC.	US-ASCII
USE_HEADER_FROM	Set the SMS source address	0
USE_HEADER_PRIORITY	Control the use of priority information from the email message's header	1
USE_HEADER_REPLY_TO	Control the use of `Reply-to:` header lines when generating SMS source addresses	0
USE_HEADER_RESENT	Control the use of `Resent-*:` header lines when generating originator information	0
USE_HEADER_SENSITIVITY	Control the use of privacy information from the email message's header	1
USE_UCS2	Use the UCS2 character set in SMS messages when applicable	1
SMS Gateway Server Option		
GATEWAY_PROFILE	Match the gateway profile name configured in the SMS Gateway Server's configuration file, `sms_gateway.cnf`	N/A
SMS Fields Options		
DEFAULT_DESTINATION_NPI	Default NPI for SMS destination addresses	0x00
DEFAULT_DESTINATION_TON	Default TON for SMS destination addresses	0x01
DEFAULT_PRIORITY	Default priority setting for SMS messages	0=GSM, CDMA 1=TDMA
DEFAULT_PRIVACY	Default privacy value flag for SMS messages	-1
DEFAULT_SERVICE_TYPE	SMS application service associated with submitted SMS messages	N/A

Table D-5 SMS Channel Options *(Continued)*

DEFAULT_SOURCE_ADDRESS	Default SMS source address	0
DEFAULT_SOURCE_NPI	Default NPI for SMS source addresses	0x00
DEFAULT_SOURCE_TON	Default TON for SMS source addresses	0x01
DEFAULT_VALIDITY_PERIOD	Default validity period for SMS messages	N/A
DESTINATION_ADDRESS_NUMERIC	Reduce the destination SMS address to only the characters 0 - 9	0
DESTINATION_ADDRESS_PREFIX	Text string to prefix destination SMS addresses with	N/A
PROFILE	SMS profile to use	GSM
USE_SAR	Sequence multiple SMS messages using the SMS sar_ fields	0
SMPP Protocol Options		
ESME_ADDRESS_NPI	ESME NPI to specify when binding to the SMPP server	0x00
ESME_ADDRESS_TON	ESME TON to specify when binding to the SMPP server	0x00
ESME_IP_ADDRESS	IP address of the host running Sun Java System Messaging Server	N/A
ESME_PASSWORD	Password to present when binding to the SMPP server	N/A
ESME_SYSTEM_ID	System identification to present to the SMSC when binding	N/A
ESME_SYSTEM_TYPE	System type to present to the SMSC when binding	N/A
MAX_PAGES_PER_BIND	Maximum number of SMS messages to submit during a single session with an SMPP server	1024
REVERSE_ORDER	Transmission sequence of multi-part SMS messages	0
SMPP_MAX_CONNECTIONS	Maximum number of simultaneous SMPP server connections	20
SMPP_PORT	For one-way SMS, TCP port the SMPP server listens on. For two-way SMS, same TCP port used for the LISTEN_PORT for the SMPP relay.	N/A
SMPP_SERVER	For one-way SMS, host name of the SMPP server to connect to. For two-way SMS, set to point to the host name or IP address of the SMS Gateway server. If using the SMPP relay's LISTEN_INTERFACE_ADDRESS option, then be sure to use the host name or IP address associated with the specified network interface address.	N/A
TIMEOUT	Timeout for completion of reads and writes with the SMPP server	30

Table D-5 SMS Channel Options *(Continued)*

Localization Options		
CONTENT_PREFIX	Text to introduce the content of the email message	Msg:
DSN_DELAYED_FORMAT	Formatting string for delivery delay notifications	an empty string
DSN_FAILED_FORMAT	Formatting string for delivery failure notifications	see description
DSN_RELAYED_FORMAT	Formatting string for relay notifications.	see description
DSN_SUCCESS_FORMAT	Formatting string to successful delivery notifications.	see description
FROM_FORMAT	Text to display indicating the originator of the email message	$a
FROM_NONE	Text to display when there is no originator	N/A
LANGUAGE	(i-default) Language group to select text fields from	i-default
LINE_STOP	Text to place at the end of each line extracted from the email message	space character
NO_MESSAGE	Text to indicate that the message had no content	]no message]
SUBJECT_FORMAT	Text to display indicating the subject of the email message	$s
SUBJECT_NONE	Text to display when there is no subject for the email message	N/A
Miscellaneous Option		
DEBUG	Enable verbose debug output	-1

Email to SMS Conversion Options

The following options control the conversion of email messages to SMS messages. The value range for the options are in parenthesis. In general, a given email message may be converted into one or more SMS messages. See "The Email to SMS Conversion Process" on page 764 for a description of this conversion process.

GATEWAY_NOTIFICATIONS

(*0 or 1*) Specifies whether or not to convert email notifications to SMS notifications. Email notification messages must conform to RFCs 1892, 1893, 1894. The default value is 0.

When GATEWAY_NOTIFICATIONS=0, such notifications are discarded and are not converted to SMS notifications.

To enable the notifications to be converted to SMS notifications, set GATEWAY_NOTIFICATIONS=1. When the option set to 1, the localization options (DSN_*_FORMAT) control which notification types (success, failure, delay, relayed) are converted into SMS messages and sent through the gateway. (If the notification type has a value of an empty string, then that type notification is not converted into SMS messages.)

MAX_MESSAGE_PARTS

(*integer*) When converting a multi-part email message to an SMS message, only the first MAX_MESSAGE_PARTS number of text parts will be converted. The remaining parts are discarded. By default, MAX_MESSAGE_PARTS is 2. To allow an unlimited number of message parts, specify a value of -1. When a value of 0 is specified, then no message content will be placed into the SMS message. This has the effect of using only header lines from the email message (for example, Subject:) to generate the SMS message.

Note that an email message containing both text and an attachment will typically consist of two parts. Note further that only plain text message parts are converted. All other MIME content types are discarded.

MAX_MESSAGE_SIZE

(*integer, >= 10*) With this option, an upper limit may be placed on the total number of bytes placed into the SMS messages generated from an email message. Specifically, a maximum of MAX_MESSAGE_SIZE bytes will be used for the one or more generated SMS messages. Any additional bytes are discarded.

By default, an upper limit of 960 bytes is imposed. This corresponds to MAX_MESSAGE_SIZE=960. To allow any number of bytes, specify a value of zero.

The count of bytes used is made after converting the email message from Unicode to either the SMSC's default character set or UCS2. This means, in the case of UCS2, that a MAX_MESSAGE_SIZE of 960 bytes will yield, at most, 480 characters since each UCS2 character is at least two bytes long.

Note that the MAX_MESSAGE_SIZE and MAX_PAGES_PER_MESSAGE options both serve the same purpose: to limit the overall size of the resulting SMS messages. Indeed, MAX_PAGE_SIZE=960 and MAX_PAGE_SIZE=160 implies MAX_PAGES_PER_MESSAGE=6. So why are there two different options? So as to allow control of the overall size or number of pages without having to consider the maximal size of a single SMS message, MAX_PAGE_SIZE. While this may not be important in the channel option file, it is important when using the MAXPAGES or MAXLEN addressing attributes described in "Directing Email to the Channel" on page 762.

Finally, note that the smaller of the two limits of MAX_MESSAGE_SIZE and MAX_PAGE_SIZE * MAX_PAGES_PER_MESSAGE is used.

MAX_PAGE_SIZE

(*integer, >= 10*) The maximum number of bytes to allow in a single SMS message is controlled with the MAX_PAGE_SIZE option. By default, a value of 160 bytes is used. This corresponds to MAX_PAGE_SIZE=160.

MAX_PAGES_PER_MESSAGE

(*integer, 1 - 255*) The maximum number of SMS messages to generate for a given email message is controlled with this option. In effect, this option truncates the email message, only converting to SMS messages that part of the email message which fits into MAX_PAGES_PER_MESSAGE SMS messages. See the description of the MAX_PAGE_SIZE option for further discussion.

By default, MAX_PAGES_PER_MESSAGE is set to the larger of 1 or MAX_MESSAGE_SIZE divided by MAX_PAGE_SIZE.

ROUTE_TO

(*string, IP host name, 1-64 bytes*) All SMS messages targeted to the profile will be rerouted to the specified IP host name using an email address of the form:

```
SMS-destination-address@route-to
```

where SMS-destination-address is the SMS message's destination address and the route-to is the IP host name specified with this option. The entire content of the SMS message is sent as the content of the resulting email message. The PARSE_RE_* options are ignored.

NOTE Use of PARSE_RE_* and ROUTE_TO options are mutually exclusive. Use of both in the same gateway profile is a configuration error.

SMSC_DEFAULT_CHARSET

(*string*) With this option, the SMSC's default character set may be specified. Use the character set names given in the file

installation-directory/config/charsets.txt

When this option is not specified, then US-ASCII is assumed. Note that the mnemonic names used in charsets.txt are defined in charnames.txt in the same directory.

When processing an email message, the header lines and text message parts are first decoded and then converted to Unicode. Next, the data is then converted to either the SMSC's default character set or UCS2, depending on the value of the USE_UCS2 option and whether or not the SMS message contains at least one glyph not found in the default SMSC character set. Note that the UCS2 character set is a 16-bit encoding of Unicode and is often referred to as UTF-16.

USE_HEADER_FROM

(*integer, 0-2*) Set this option to allow the From: address to be passed to the SMSC. The value indicates where the From: address is taken from and what format it will have. Table D-6 shows the allowable values and their meaning.

Table D-6 USE_HEADER_FROM Values

Value	Description
0	SMS source address never set from the From: address. Use attribute-value pair found
1	SMS source address set to from-local@from-domain, where the From: address is: @from-route:from-local@from-domain
2	SMS source address set to from-local, where the From: address is: @from-route:from-local@from-domain

USE_HEADER_PRIORITY

(*0 or 1*) This option controls handling of RFC 822 Priority: header lines. By default, information from the Priority: header line is used to set the resulting SMS message's priority flag, overriding the default SMS priority specified with the DEFAULT_PRIORITY option. This case corresponds to USE_HEADER_PRIORITY=1. To disable use of the RFC 822 Priority: header line, specify USE_HEADER_PRIORITY=0.

See the description of the DEFAULT_PRIORITY option for further information on the handling the SMS priority flag.

USE_HEADER_REPLY_TO

(*0 or 1*) When USE_HEADER_FROM =1, this option controls whether or not a Reply-to: or Resent-reply-to: header line is considered for use as the SMS source address. By default, Reply-to: and Resent-reply-to: header lines are ignored. This corresponds to an option value of 0. To enable consideration of these header lines, use an option value of 1.

Note that RFC 2822 has deprecated the use of Reply-to: and Resent-reply-to: header lines.

USE_HEADER_RESENT

(*0 or 1*) When USE_HEADER_FROM =1, this option controls whether or not Resent- header lines are considered for use as the SMS source address. By default, Resent- header lines are ignored. This corresponds to an option value of 0. To enable consideration of these header lines, use an option value of 1.

Note that RFC 2822 has deprecated the use of Resent- header lines.

USE_HEADER_SENSITIVITY

(*0 or 1*) The USE_HEADER_SENSITIVITY option controls handling of RFC 822 Sensitivity: header lines. By default, information from the Sensitivity: header line is used to set the resulting SMS message's privacy flag, overriding the default SMS privacy specified with the DEFAULT_PRIVACY option. This case, which is the default, corresponds to USE_HEADER_SENSITIVITY=1. To disable use of RFC 822 Sensitivity: header lines, specify USE_HEADER_SENSITIVITY=0.

See the description of the DEFAULT_PRIVACY option for further information on the handling the SMS privacy flag.

USE_UCS2

(*0 or 1*) When appropriate, the channel will use the UCS2 character set in the SMS messages it generates. This is the default behavior and corresponds to USE_UCS2=1. To disable the use of the UCS2 character set, specify USE_UCS2=0. See the description of the SMSC_DEFAULT_CHARSET option for further information on character set issues.

Table D-7 Valid Values for USE_UCS2

USE_UCS2 Value	Result
1 (default)	The SMSC default character set will be used whenever possible. When the originating email message contains glyphs not in the SMSC default character set, then the UCS2 character set will be used.
0	The SMSC default character set will always be used. Glyphs not available in that character set will be represented by mnemonics (for example, "AE" for AE-ligature).

SMS Gateway Server Option

GATEWAY_PROFILE

The name of the gateway profile in the SMS Gateway Server configuration file, sms_gateway.cnf.

SMS Options

The following options allow for specification of SMS fields in generated SMS messages.

DEFAULT_DESTINATION_NPI

(*integer, 0 - 255*) By default, destination addresses will be assigned an NPI (Numeric Plan Indicator) value of zero. With this option, an alternate integer value in the range 0 to 255 may be assigned. Typical NPI values include those found in Table D-8 that follows:

Table D-8 Numeric Plan Indicator Values

Value	Description
0	Unknown
1	ISDN (E.163, E.164)
3	Data (X.121)
4	Telex (F.69)
6	Land Mobile (E.212)
8	National
9	Private
10	ERMES
14	IP address (Internet)
18	WAP client ID
>= 19	Undefined

Values for this option may be specified in one of three ways:

- A decimal value (for example, 10).
- A hexadecimal value prefixed by “0x” (for example, 0x0a).
- One of the following case-insensitive text strings (the associated decimal value is shown in parentheses): data (3), default (0), e.163 (1), e.164 (1), e.212 (6), ermes (10), f.69 (4), Internet (14), ip (14), isdn (1), land-mobile (6), national (8), private (9), telex (4), unknown (0), wap (18), x.121 (3).

DEFAULT_DESTINATION_TON

(*integer, 0 - 255*) By default, destination addresses will be assigned a TON (Type of Number) designator value of zero. With this option, an alternate integer value in the range 0 to 255 may be assigned. Typical TON values include those found in Table D-9 that follows:

Table D-9 Typical TON Values

Value	Description
0	Unknown
1	International
2	National
3	Network specific
4	Subscriber number
5	Alphanumeric
6	Abbreviated
>=7	Undefined

Values for this option may be specified in one of three ways:

- A decimal value (for example, 10)
- A hexadecimal value prefixed by “0x” (for example, 0x0a)
- One of the following case-insensitive text strings (the associated decimal value is shown in parentheses): abbreviated (6), alphanumeric (5), default (0), international (1), national (2), network-specific (3), subscriber (4), unknown (0).

DEFAULT_PRIORITY

(integer, 0 - 255) SMS messages have a mandatory priority field. The interpretation of SMS priority values is shown in Table D-10 that follows:

Table D-10 SMS Priority Values Interpreted for Each SMS Profile Type

Value	GSM	TDMA	CDMA
0	Non-priority	Bulk	Normal
1	Priority	Normal	Interactive
2	Priority	Urgent	Urgent
3	Priority	Very urgent	Emergency

With this option, the default priority to assign to SMS messages may be specified. When not specified, a default priority of 0 is used for PROFILE=GSM and CDMA, and a priority of 1 for PROFILE=TDMA.

Note that if USE_HEADER_PRIORITY=1 and an email message has an RFC 822 Priority: header line, then the priority specified in that header line will instead be used to set the priority of the resulting SMS message. Specifically, if USE_HEADER_PRIORITY=0, then the SMS priority flag is always set in accord with the DEFAULT_PRIORITY option and the RFC 822 Priority: header line is always ignored. If USE_HEADER_PRIORITY=1, then the originating email message's RFC 822 Priority: header line is used to set the SMS message's priority flag. If that header line is not present, then the SMS priority flag is set using the DEFAULT_PRIORITY option.

The mapping used to translate RFC 822 Priority: header line values to SMS priority flags is shown in table that follows:

Table D-11 Mapping for Translating Priority: Header to SMS Priority Flags

RFC 822	SMS priority flag		
Priority: value	**GSM**	**TDMA**	**CDMA**
Third	Non-priority (0)	Bulk (0)	Normal (0)
Second	Non-priority (0)	Bulk (0)	Normal (0)
Non-urgent	Non-priority (0)	Bulk (0)	Normal (0)
Normal	Non-priority (0)	Normal (1)	Normal (0)
Urgent	Priority (1)	Urgent (2)	Urgent (2)

DEFAULT_PRIVACY

(*integer, -1, 0 - 255*) Whether or not to set the privacy flag in an SMS message, and what value to use is controlled with the DEFAULT_PRIVACY and USE_HEADER_SENSITIVITY options. By default, a value of -1 is used for DEFAULT_PRIVACY. Table D-12 that follows shows the result of setting the DEFAULT_PRIVACY and USE_HEADER_SENSITIVITY options to various values.

Table D-12 Result of Values for DEFAULT_PRIVACY and USE_HEADER_SENSITIVITY

DEFAULT_PRIVACY	USE_HEADER_SENSITIVITY	Result
-1	0	The SMS privacy flag is never set in SMS messages.
n >= 0	0	The SMS privacy flag is always set to the value n. RFC 822 Sensitivity: header lines are always ignored.

Table D-12 Result of Values for DEFAULT_PRIVACY and USE_HEADER_SENSITIVITY

DEFAULT_PRIVACY	USE_HEADER_SENSITIVITY	Result
-1 (default)	1 (default)	The SMS message's privacy flag is only set when the originating email message has an RFC 822 Sensitivity: header line. In that case, the SMS privacy flag is set to correspond to the Sensitivity: header line's value. This is the default.
n >= 0	1	The SMS message's privacy flag is set to correspond to the originating email message's RFC 822 Sensitivity: header line. If the email message does not have a Sensitivity: header line, then the value of the SMS privacy flag is set to n.

The SMS interpretation of privacy values is shown in Table D-13 that follows:

Table D-13 SMS Interpretation of Privacy Values

Value	Description
0	Unrestricted
1	Restricted
2	Confidential
3	Secret
>= 4	Undefined

The mapping used to translate RFC 822 Sensitivity: header line values to SMS privacy values is shown in Table D-14 that follows:

Table D-14 Mapping Translation of Sensitivity: Headers to SMS Privacy Values

RFC 822 Sensitivity: value	SMS privacy value
Personal	1 (Restricted)
Private	2 (Confidential)
Company confidential	3 (Secret)

DEFAULT_SERVICE_TYPE

(*string, 0 - 5 bytes*) Service type to associate with SMS messages generated by the channel. By default, no service type is specified (that is, a zero length string). Some common service types are: `CMT` (cellular messaging), `CPT` (cellular paging), `VMN` (voice mail notification), `VMA` (voice mail alerting), `WAP` (wireless application protocol), and `USSD` (unstructured supplementary data services).

DEFAULT_SOURCE_ADDRESS

(*string, 0 - 20 bytes*) Source address to use for SMS messages generated from email messages. Note that the value specified with this option is overridden by the email message's originator address when `USE_HEADER_FROM=1`. By default, the value is disabled, that is, has a value of `0`.

DEFAULT_SOURCE_NPI

(*integer, 0 - 255*) By default, source addresses will be assigned an NPI value of zero. With this option, an alternate integer value in the range 0 to 255 may be assigned. See the description of the `DEFAULT_DESTINATION_NPI` option for a table of typical NPI values.

DEFAULT_SOURCE_TON

(*integer, 0 - 255*) By default, source addresses will be assigned a TON designator value of zero. With this option, an alternate integer value in the range 0 to 255 may be assigned. See the description of the `DEFAULT_DESTINATION_TON` option for a table of typical TON values.

DEFAULT_VALIDITY_PERIOD

(*string, 0 - 252 bytes*) By default, SMS messages are not given a relative validity period; instead, they use the SMSC's default value. Use this option to specify a different relative validity period. Values may be specified in units of seconds, minutes, hours, or days. Table D-15 that follows specifies the format and description of the various values for this option:

Table D-15 `DEFAULT_VALIDITY_PERIOD` Format and Values

Format	Description
nnn	Implicit units of seconds; for example, 604800
*nnn*s	Units of seconds; for example, 604800s
*nnn*m	Units of minutes; for example, 10080m
*nnn*h	Units of hours; for example, 168h
*nnn*d	Units of days; for example, 7d

A specification of 0, 0s, 0m, 0h, or 0d may be used to select the SMSC's default validity period. That is, when a specification of 0, 0s, 0m, 0h, or 0d is used, an empty string is specified for the validity period in generated SMS messages.

Note that this option does not accept values in UTC format.

DESTINATION_ADDRESS_NUMERIC

(*0 or 1*) Use this option to strip all non-numeric characters from the SMS destination address extracted from the email envelope `To:` address. For instance, if the envelope `To:` address is:

```
"(800) 555-1212"@sms.siroe.com
```

then it will be reduced to:

```
8005551212@sms.siroe.com
```

To enable this stripping, specify a value of 1 for this option. By default, this stripping is disabled which corresponds to an option value of 0. Note that when enabled, the stripping is done before any destination address prefix is added via the `DESTINATION_ADDRESS_PREFIX` option.

DESTINATION_ADDRESS_PREFIX

(*string*) In some instances, it may be necessary to ensure that all SMS destination addresses are prefixed with a fixed text string; for example, "+". This option may be used to specify just such a prefix. The prefix will then be added to any SMS destination address which lacks the specified prefix. To prevent being stripped by the `DESTINATION_ADDRESS_NUMERIC` option, this option is applied after the `DESTINATION_ADDRESS_NUMERIC` option.

PROFILE

(*string*) Specify the SMS profiling to be used with the SMSC. Possible values are `GSM`, `TDMA`, and `CDMA`. When not specified, GSM is assumed. This option is only used to select defaults for other channel options such as `DEFAULT_PRIORITY` and `DEFAULT_PRIVACY`.

USE_SAR

(0 or 1) Sufficiently large email messages may need to be broken into multiple SMS messages. When this occurs, the individual SMS messages can optionally have sequencing information added using the SMS sar_ fields. This produces a "segmented" SMS message which can be re-assembled into a single SMS message by the receiving terminal. Specify `USE_SAR=1` to indicate that this sequencing information is to be added when applicable. The default is to not add sequencing information and corresponds to `USE_SAR=0`.

When USE_SAR=1 is specified, the REVERSE_ORDER option is ignored.

SMPP Options

The following options allow for specification of SMPP protocol parameters. The options with names beginning with the string "ESME_" serve to identify the MTA when it acts as an External Short Message Entity (ESME); that is, when the MTA binds to an SMPP server in order to submit SMS messages to the server's associated SMSC.

ESME_ADDRESS_NPI

(integer, 0 - 255) By default, bind operations will specify an ESME NPI value of zero indicating an unknown NPI. With this option, an alternate integer value in the range 0 to 255 may be assigned. See the description of the DEFAULT_DESTINATION_NPI option for a table of typical NPI values.

ESME_ADDRESS_TON

(integer, 0 - 255) By default, bind operations will specify an ESME TON value of 0. With this option, an alternate integer value in the range 0 to 255 may be assigned. See the description of the DEFAULT_DESTINATION_TON option for a table of typical TON values.

ESME_IP_ADDRESS

(*string, 0 - 15 bytes*) When binding to the SMPP server, the BIND PDU indicates that the client's (that is, ESME's) address range is an IP address. This is done by specifying a TON of 0x00 and an NPI of 0x0d. The value of the address range field is then set to be the IP address of the host running the SMS channel. Specify the IP address in dotted decimal format; for example, 127.0.0.1.

ESME_PASSWORD

(*string, 0 - 8 bytes*) When binding to the SMPP server, a password may be required. If so, then specify that password with this option. By default, a zero-length password string is presented.

ESME_SYSTEM_ID

(*string, 0 - 15 bytes*) When binding to the SMPP server, a system ID for the MTA may be supplied. By default, no system ID is specified (that is, a zero-length string is used). To specify a system ID, use this option.

ESME_SYSTEM_TYPE

(*string, 0 - 12 bytes*) When binding to the SMPP server, a system type for the MTA may be supplied. By default, no system type is specified (that is, a zero-length string is used).

MAX_PAGES_PER_BIND

(*integer, >= 0*) Some SMPP servers may limit the maximum number of SMS messages submitted during a single, bound session. In recognition of this, this option allows specification of the maximum number of SMS messages to submit during a single session. Once that limit is reached, the channel will unbind, close the TCP/IP connection, re-connect, and then rebind.

By default, a value of 1024 is used for `MAX_PAGES_PER_BIND`. Note that the channel will also detect `ESME_RTHROTTLED` errors and adjust `MAX_PAGES_PER_BIND` during a single run of the channel accordingly.

REVERSE_ORDER

(*0 or 1*) When an email message generates more than one SMS message, those SMS messages can be submitted to the SMSC in sequential order (`REVERSE_ORDER=0`), or reverse sequential order (`REVERSE_ORDER=1`). Reverse sequential order is useful for situations where the receiving terminal displays the last received message first. In such a case, the last received message will be the first part of the email message rather than the last. By default, `REVERSE_ORDER=1` is used.

Note that this option is ignored when `USE_SAR=1` is specified.

SMPP_MAX_CONNECTIONS

(*integer, 1 - 50*) This option controls the maximum number of simultaneous SMPP connections per process. As each connection has an associated thread, this option also places a limit on the maximum number of "worker" threads per process. By default, `SMPP_MAX_CONNECTIONS=20`.

SMPP_PORT

(*integer, 1 - 65535*) The TCP port which the SMPP server listens on may be specified with either this option or the `port` channel keyword. This port number must be specified through either of these two mechanisms. If it is specified with both mechanisms, then the setting made with the `SMPP_PORT` option takes precedence. Note that there is no default value for this option.

For two-way SMS, make sure its the same port as the `LISTEN_PORT` for the SMPP relay.

SMPP_SERVER

(*string, 1 - 252 bytes*) For one-way SMS, by default, the IP host name of the SMPP server to connect to is the official host name associated with the channel; that is, the host name shown on the second line of the channel's definition in MTA's configuration. This option may be used to specify a different host name or IP address which will override that specified in the channel definition. When specifying an IP address, use dotted decimal notation; for example, 127.0.0.1.

For two-way SMS, set to point to the host name or IP address of the SMS Gateway Server. If using the SMPP relay's LISTEN_INTERFACE_ADDRESS option, then be sure to use the host name or IP address associated with the specified network interface address.

TIMEOUT

(*integer*, *>= 2*) By default, a timeout of 30 seconds is used when waiting for data writes to the SMPP server to complete or for data to be received from the SMPP server. Use the TIMEOUT option to specify, in units of seconds, a different timeout value. The specified value should be at least 1second.

Localization Options

In constructing SMS messages, the SMS channel has a number of fixed text strings it puts into those messages. These strings, for example, introduce the email's From: address and Subject: header line. With the channel options described in this section, versions of these strings may be specified for different languages and a default language for the channel then specified. Code Example D-2 shows the language part of the option file:

Code Example D-2 Language Specification Part of Channel Option File

```
LANGUAGE=default-language

[language=i-default]
FROM_PREFIX=From:
SUBJECT_PREFIX=Subj:
CONTENT_PREFIX=Msg:
LINE_STOP=
NO_MESSAGE=[no message]
REPLY_PREFIX=Re:

[language=en]
FROM_PREFIX=From:
SUBJECT_PREFIX=Subj:
CONTENT_PREFIX=Msg:
LINE_STOP=
NO_MESSAGE=[no message]
REPLY_PREFIX=Re:
 ...
```

Within each [language=x] block, the localization options relevant to that language may be specified. If a particular option is not specified within the block, then the global value for that option is used. A localization option specified outside of a [language=x] block sets the global value for that option.

For the options listed below, the string values must be specified using either the US-ASCII or UTF-8 character sets. Note that the US-ASCII character set is a special case of the UTF-8 character set.

CONTENT_PREFIX

(*string, 0 - 252 bytes*) Text string to place in the SMS message before the content of the email message itself. Default global value is the US-ASCII string "`Msg:`".

DSN_DELAYED_FORMAT

(string, 0-256 characters) Formatting string for delivery delay notifications. By default, an empty string is used for this option, thereby inhibiting the conversion to SMS of delay notifications. Note that `GATEWAY_NOTIFICATIONS` must be set to `1` for this option to be in effect. This option is ignored when `GATEWAY_NOTIFICATIONS=0`.

DSN_FAILED_FORMAT

(string, 0-256 characters) Formatting string for permanent delivery failure notifications. The default value of this option is the string:

```
Unable to deliver your message to $a; no further delivery attempts will be
made.
```

To inhibit conversion of failure notifications, specify an empty string for this option. Note that `GATEWAY_NOTIFICATIONS` must be set to `1` for this option to be in effect. This option is ignored when `GATEWAY_NOTIFICATIONS=0`.

DSN_RELAYED_FORMAT

(string, 0-256 characters) Formatting string for relay notifications. The default value is the string:

```
Your message to $a has been relayed to a messaging system which may not
provide a final delivery confirmation
```

To inhibit conversion of relay notifications, specify an empty string for this option. Note that `GATEWAY_NOTIFICATIONS` must be set to `1` for this option to be in effect. This option is ignored when `GATEWAY_NOTIFICATIONS=0`.

DSN_SUCCESS_FORMAT

(string, 0-256 characters) Formatting string for successful delivery notifications. The default value is the string:

```
Your message to $a has been delivered
```

To inhibit conversion of successful delivery notifications, specify an empty string for this option. Note that `GATEWAY_NOTIFICATIONS` must be set to `1` for this option to be in effect. This option is ignored when `GATEWAY_NOTIFICATIONS=0`.

FROM_FORMAT

(*string, 0 - 252 bytes*) Formatting template to format the originator information to insert into the SMS message. The default global value is the US-ASCII string "$a" which substitutes in the originator's email address. See "Formatting Templates" on page 801 for further details.

FROM_NONE

(*string, 0 - 252 bytes*) Text string to place in the SMS message when there is no originator address to display. The default global value is an empty string.

Note that normally, this option will never be used as sites will typically reject email messages which lack any originator address.

LANGUAGE

(*string, 0 - 40 bytes*) The default language group to select text strings from. If not specified, then the language will be derived from the host's default locale specification. If the host's locale specification is not available or corresponds to "C", then i-default will be used. (i-default corresponds to "English text intended for an international audience.")

LINE_STOP

(*string, 0 - 252 bytes*) Text string to place in the SMS message between lines extracted from the email message. The default global value is the US-ASCII space character, " ".

NO_MESSAGE

(*string, 0 - 252 bytes*) Text string to place in the SMS message to indicate that the email message had no content. The default global value is the US-ASCII string "[no message]".

SUBJECT_FORMAT

(*string, 0 - 252 bytes*) Formatting template to format the content of the `Subject:` header line for display in the SMS message. The global default value for this option is the US-ASCII string "($s)". See "Formatting Templates" on page 801 for further details.

See the `SUBJECT_NONE` option for a description of the handling when there is no `Subject:` header line or the content of that header line is an empty string.

SUBJECT_NONE

(*string, 0 - 252 bytes*) Text string to display when the originating email message either has no `Subject:` header line, or the `Subject:` header line's value is an empty string. The default global value for this option is the empty string.

DEBUG

(integer, bitmask) Enable debug output. The default value is `6` which selects warning and error messages. Any non-zero value enables debug output for the channel itself, the same as specifying `master_debug` on the channel definition.Table D-16 defines the bit values of the `DEBUG` bitmask.

Table D-16 DEBUG Bitmask

Bit	Value	Description
0-31	`-1`	Extremely verbose output
0	1	Informational messages
1	2	Warning messages
3	4	Error messages
3	8	Subroutine call tracing
4	16	Hash table diagnostics
5	32	I/O diagnostics, receive
6	64	I/O diagnostics, transmit
7	128	SMS to email conversion diagnostics (mobile originate and SMS notification)
8	256	PDU diagnostics, header data
9	512	PDU diagnostics, body data
10	1024	PDU diagnostics, type-length-value data
11	2048	Option processing; sends all option settings to the log file.

Formatting Templates

The formatting templates specified with the `FROM_FORMAT`, `SUBJECT_FORMAT`, and all the `DSN_*` channel options are UTF-8 strings which may contain a combination of literal text and substitution sequences. Assuming the sample email address of

```
Jane Doe <user@siroe>
```

The recognized substitution sequences are shown in Table D-17 that follows:

Table D-17 Substitution Sequences

Sequence	Description
$a	Replace with the local and domain part of the originator's email address (for example, "`user@siroe`")

Table D-17 Substitution Sequences

Sequence	Description
`$d`	Replace with the domain part of the originator's email address (for example, "`domain`")
`$p`	Replace with the phrase part, if any, of the originator's email address (for example, "`Jane Doe`")
`$s`	Replace with the content of the `Subject:` header line
`$u`	Replace with the local part of the originator's email address (for example, "`user`")
`\x`	Replace with the literal character "`x`"

For example, the formatting template

```
From: $a
```

produces the text string

```
From: user@siroe
```

The construct,

```
${xy:alternate text}
```

may be used to substitute in the text associated with the sequence `x`. If that text is the empty string, the text associated with the sequence `y` is instead used. And, if that text is the empty string, to then substitute in the alternate text. For example, consider the formatting template

```
From: ${pa:unknown sender}
```

For the originator email address

```
John Doe <jdoe@siroe.com>
```

which has a phrase part, the template produces:

```
From: John Doe
```

However, for the address

```
jdoe@siroe.com
```

which lacks a phrase, it produces

```
From: jdoe@siroe.com
```

And for an empty originator address, it produces

```
From: unknown sender
```

Adding Additional SMS Channels

You may configure the MTA to have more than one SMS channel. There are two typical reasons to do this:

1. To communicate with different SMPP servers.

   This is quite straightforward: just add an additional SMS channel to the configuration, being sure to (a) give it a different channel name, and (b) associate different host names with it. For example,

   ```
   sms_mway port 55555 threaddepth 20
   smpp.siroe.com
   ```

   ```
   sms_ace port 777 threaddepth 20
   sms.ace.net
   ```

   Note that no new rewrite rule is needed. If there is no directly matching rewrite rule, Messaging Sever looks for a channel with the associated host name. For example, if the server is presented with `user@host.domain`, it would look for a channel of the name “host.domain”. If it finds such a channel, it routes the message there. Otherwise, it starts looking for a rewrite rule for the “.domain” and if none is there, then for the dot (“.”) rule. For more information on rewrite rules, see the *Sun Java System Messaging Server Administration Guide.*

2. To communicate with the same SMPP server but using different channel options.

   To communicate with the same SMPP server, using different channel options, specify the same SMPP server in the SMPP_SERVER channel option for each channel definition.

   Using this mechanism is necessary since two different channels cannot have the same official host name (that is, the host name listed in the second line of the channel definition). To allow them to communicate with the same SMPP server, define two separate channels, with each specifying the same SMPP server in their SMPP_SERVER in their channel option files.

   For example, you could have the following channel definitions,

   ```
   sms_mway_1 port 55555 threaddepth 20
   SMS-DAEMON-1
   ```

   ```
   sms_mway_2 port 55555 threaddepth 20
   SMS-DAEMON-2
   ```

   and rewrite rules,

   ```
   sms-1.siroe.com $u%sms-1.siroe.com@SMS-DAEMON-1
   sms-2.siroe.com $U%sms-2.siroe.com@SMS-DAEMON-2
   ```

   Then, to have them both use the same SMPP server, each of these two channels would specify SMPP_SERVER=smpp.siroe.com in their channel option file.

Adjusting the Frequency of Delivery Retries

When an SMS message cannot be delivered owing to temporary errors (for example, the SMPP server is not reachable), the email message is left in the delivery queue and retried again later. Unless configured otherwise, the Job Controller will not re-attempt delivery for an hour. For SMS messaging, that is likely too long to wait. As such, it is recommended that the backoff channel keyword be used with the SMS channel to specify a more aggressive schedule for delivery attempts. For example,

```
sms_mway port 55555 threaddepth 20 \
  backoff pt2m pt5m pt10m pt30m notices 1
smpp.siroe.com
```

With the above settings, a redelivery attempt will be made at two minutes after the first attempt. If that then fails, then five minutes after the second attempt. Then ten minutes later and finally every thirty minutes. The `notices 1` channel keyword causes the message to be returned as undeliverable if it cannot be delivered after a day.

Sample One-Way Configuration (MobileWay)

The MTA SMS channel may be used with any SMPP V3.4 compatible SMPP server. For purposes of illustrating an example configuration, this section explains how to configure the SMS channel for use with a MobileWay SMPP server. MobileWay (http://www.mobileway.com/) is a leading provider of global data and SMS connectivity. By routing your SMS traffic through MobileWay, you can reach SMS subscribers on most of the major SMS networks throughout the world.

When requesting an SMPP account with MobileWay, you may be asked to answer the following questions:

- IP address of your SMPP client: Supply the IP address of your Messaging Server system as seen by other domains on the Internet.
- Default validity period: This is the SMS validity period which MobileWay will use should a validity period not be specified in the SMS messages you submit. SMS messages which cannot be delivered before this validity period expires will be discarded. Supply a reasonable value (for example, 2 days, 7 days, etc.).
- Window size: This is the maximum number of SMS messages your SMPP client will submit before it will stop and wait for responses from the SMPP server before submitting any further SMS messages. You must supply a value of 1 message.
- Timezone: Specify the timezone in which your Messaging Server system operates. The timezone should be specified as an offset from GMT.
- Timeout: Not relevant to one-way SMS messaging.
- IP address and TCP port for outbind requests: Not relevant for one-way SMS messaging.

After supplying MobileWay with the answers to the above questions, they will provide you with an SMPP account and information necessary to communicate with their SMPP servers. This information includes

```
Account Address: a.b.c.d:p
Account Login: system-id
Account Passwd: secret
```

The Account Address field is the IP address, a.b.c.d, and TCP port number, p, of the MobileWay SMPP server you will be connecting to. Use these values for the SMPP_SERVER and SMPP_PORT channel options. The Account Login and Passwd are, respectively, the values to use for the ESME_SYSTEM_ID and ESME_PASSWORD channel options. Using this information, your channel's option file should include

```
SMPP_SERVER=a.b.c.d
SMPP_PORT=p
ESME_SYSTEM_ID=system-id
ESME_PASSWORD=secret
```

Now, to interoperate with MobileWay you need to make two additional option settings

```
ESME_ADDRESS_TON=0x01
DEFAULT_DESTINATION_TON=0x01
```

The rewrite rule in the imta.cnf file can appear as

```
sms.your-domain $u@sms.your-domain
```

And, the channel definition in the imta.cnf file can appear as

```
sms_mobileway
sms.your-domain
```

Once the channel option file, rewrite rule, and channel definition are in place, a test message may be sent. MobileWay requires International addressing of the form

```
+<country-code><subscriber-number>
```

For instance, to send a test message to the North American subscriber with the subscriber number (800) 555-1212, you would address your email message to

```
+18005551212@sms.your-domain
```

Debugging

To debug the channel, specify the master_debug channel keyword in the channel's definition. For example,

```
sms_mway port 55555 threaddepth 20 \
  backoff pt2m pt5m pt10m pt30m notices 1 master_debug
```

With the master_debug channel keyword, basic diagnostic information about the channel's operation will be output to the channel's log file. For verbose diagnostic information about the SMPP transactions undertaken by the channel, also specify

```
DEBUG=-1
```

in the channel's option file.

Configuring the SMS Channel for Two-Way SMS

For general directions on configuring the SMS channel, refer to earlier topics starting with "SMS Channel Configuration" on page 778. Configure the SMS channel as though it will be talking directly to the remote SMSC, with the exceptions listed in Table D-18 that follows:

Table D-18 Two-Way Configuration Exceptions

Exception	Explanation
`master` channel keyword	Remove the `master` channel keyword, if present. It is no longer needed for SMS channel configuration.
`SMPP_SERVER`	Set to point to the host name of IP address of the SMS Gateway Server. If using the SMPP relay's `LISTEN_INTERFACE_ADDRESS` option (see "Configuration Options" on page 819), then be sure to use the host name or IP address associated with the specified network interface address.
`SMPP_PORT`	Same TCP port as used for the `LISTEN_PORT` setting used to instantiate the SMPP relay (see "An SMPP Relay" on page 816).
`DEFAULT_SOURCE_ADDRESS`	Pick a value and then configure the remote SMSC to route this address back to the Gateway SMPP server. In the SMS channel's option file, specify the chosen value with this option.
`GATEWAY_PROFILE`	Set to match the gateway profile name. See "A Gateway Profile" on page 815.
`USE_HEADER_FROM`	Set to `0`.

All other channel configurations should be done as described in the SMS Channel documentation.

As mentioned in "Setting Up Bidirectional SMS Routing" on page 813, the remote SMSC needs to be configured to route the SMS address, defined in the `DEFAULT_SOURCE_ADDRESS` channel option, to the Gateway's SMPP server using the TCP port number specified with the `LISTEN_PORT` option. (For how to specify the `LISTEN_PORT`, see "An SMPP Server" on page 817.)

Note that multiple SMS channels may use the same SMPP relay. Similarly, there need be only one SMPP server or gateway profile to handle SMS replies and notifications for multiple SMS channels. The ability to configure multiple relays, servers, and gateway profiles exists to effect different usage characteristics through configuration options.

SMS Gateway Server Theory of Operation

The SMS Gateway Server facilitates two-way SMS through mechanisms that allow mobile originated SMS messages to be matched to the correct email address. The following SMS Gateway Server topics are covered in this section:

Function of the SMS Gateway Server

The SMS Gateway Server simultaneously functions as both an SMPP relay and server. It may be configured to have multiple "instantiations" of each function. For instance, it may be configured to have three different SMPP relays, each listening on different TCP ports or network interfaces and relaying to different remote SMPP servers. Similarly, it may be configured to have four different SMPP servers, each listening on different combinations of TCP ports and network interfaces.

The SMS Gateway Server may be configured with zero or more gateway profiles for sending SMS messages to email. Each gateway profile describes which destination SMS addresses match the profile, how to extract the destination email addresses from SMS messages, and various characteristics of the SMS to email conversion process. Each SMS message presented to the SMS Gateway Server through either its SMPP relay or server are compared to each profile. If a match is found, then the message is routed to email.

Finally, the gateway profiles also describe how to handle notification messages returned by remote SMSCs in response to previous email-to-mobile messages.

Behavior of the SMPP Relay and Server

When acting as an SMPP relay, the SMS Gateway Server attempts to be as transparent as possible, relaying all requests from local SMPP clients on to a remote SMPP server and then relaying back the remote server's responses. There are two exceptions:

- When a local SMPP client submits a message whose SMS destination address matches one of the configured gateway profiles, the submitted SMS message is sent directly back to email; the SMS message is not relayed to a remote SMPP server.

- When a local or remote SMPP client submits a message whose SMS destination address matches a unique SMS source address previously generated by the SMPP relay, the SMS message is a reply to a previously relayed message. This reply is directed back to the originator of the original message.

Note that typically the SMS Gateway Server will be configured such that the unique SMS source addresses which it generates match one of the gateway profiles.

NOTE The SMS Gateway Server's SMPP relay is only intended for use with qualified, Sun Java System SMPP clients, that is, the Sun Java System Messaging Server's SMS channel. It is not intended for use with arbitrary SMPP clients.

When acting as an SMPP server, the SMS Gateway Server directs SMS messages to email for three circumstances:

- The SMS messages are mobile originated and match a gateway profile.
- The SMS messages are mobile originated and the SMS destination address matches a previously generated unique SMS source address.
- The SMS messages are SMS notifications which correspond to email-to-mobile messages previously relayed by the SMS Gateway Server's SMPP relay.

All other SMS messages are rejected by the SMPP server.

Remote SMPP to Gateway SMPP Communication

Remote SMPP clients communicate to the Gateway SMPP server with Protocol Data Units (PDUs). Remote SMPP clients emit request PDUs to which the Gateway SMPP server responds. The Gateway SMPP server operates synchronously. It completes the response to a request PDU before it processes the next request PDU from the connected remote SMPP client.

Table D-19 that follows lists the request PDUs the Gateway SMPP server handles, and specifies the Gateway SMPP server's response.

Table D-19 SMPP Server Protocol Data Units

Request PDU	SMPP Server Response
BIND_TRANSMITTER BIND_TRANSCEIVER UNBIND	Responded to with the appropriate response PDU. Authentication credentials are ignored.
OUTBIND	Gateway SMPP server sends back a BIND_RECEIVER PDU. Authentication credentials presented are ignored.
SUBMIT_SM DATA_SM	Attempts to match the destination SMS address with either a unique SMS source address or the SELECT_RE setting of a Gateway profile. If neither is matched, the PDU is rejected with an ESME_RINVDSTADR error.
DELIVER_SM	Attempts to find either the destination SMS address or the receipted message ID in the historical record. If neither is matched, returns the error ESME_RINVMSGID.
BIND_RECEIVER	Not supported. Returns a GENERIC_NAK PDU with an ESME_RINVCMDID error.
SUBMIT_MULTI	Not supported. Returns a GENERIC_NAK PDU with an ESME_RINVCMDID error.
REPLACE_SM	Not supported. Returns a GENERIC_NAK PDU with an ESME_RINVCMDID error.
CANCEL_SM	Not supported. Returns a GENERIC_NAK PDU with an ESME_RINVCMDID error.
QUERY_SM	Not supported. Returns a GENERIC_NAK PDU with an ESME_RINVCMDID error.
QUERY_LAST_MSGS	Not supported. Returns a GENERIC_NAK PDU with an ESME_RINVCMDID error.
QUERY_MSG_DETAILS	Not supported. Returns a GENERIC_NAK PDU with an ESME_RINVCMDID error.
ENQUIRE_LINK	Returns ENQUIRE_LINK_RESP PDU.
ALERT_NOTIFICATION	Accepted but ignored.

SMS Reply and Notification Handling

The SMS Gateway Server maintains a historical record of each SMS message relayed through its SMPP relays. The need to use historical data arises from the fact that when submitting an email message to SMS it is generally not possible to convert the email address of the message's originator to an SMS source address. Since any SMS replies and notifications will be directed to this SMS source address, a problem arises. This is resolved by using automatically generated, unique SMS source addresses in relayed messages. The remote SMSCs are then configured to route these SMS source addresses back to the Gateway SMPP server.

The historical data is represented as an in-memory hash table of message IDs and generated, unique SMS source addresses. This data along with the associated email origination data are also stored on disk. The disk based storage is a series of files, each file representing `HASH_FILE_ROLLOVER_PERIOD` seconds of transactions (the default is 30 minutes). Each file is retained for `RECORD_LIFETIME` seconds (the default is 3 days). See the *Sun Java System Messaging Server Deployment Guide* for a discussion of the in-memory and on-disk resource requirements of the historical data.

Each record has three components:

- Email origination data (such as, envelope From: and To: addresses). This data is supplied by the MTA SMS channel when it submits a message.
- The unique SMS source address generated by the SMPP relay and inserted into the relayed SMS message.
- The resulting receipted message ID returned by the remote SMSC's SMPP server when it accepts a submission.

Routing Process for SMS Replies

The Gateway SMPP relays and servers use historical records to handle SMS replies, notifications and mobile originated messages. When an SMS message is presented to the SMPP relay or server, the following routing process is followed:

1. The SMS destination address is compared against the historical record to see if there is a matching, unique SMS source address that the SMPP relay previously generated. If a match is found, see Step 6.
2. If there is no match, but the message is an SMS notification (SMPP `DELIVER_SM` PDU), then the receipted message ID, if present, is compared against the historical record. If a match is found, go to Step 8. [The SMS Gateway Server actually allows these to be presented to either the SMPP relay or SMPP server.]

3. If there is no match, then the destination SMS address is compared against the SELECT_RE option expressions for each configured gateway profile. If a match is found, then go to Step 9.
4. If there is no match and the SMS message was presented to the Gateway SMPP relay, then the message is relayed to the remote SMPP server.
5. If there is no match and the SMS message was presented to the Gateway SMPP server, then the message is determined to be an invalid message and an error response is returned in the SMPP response PDU. For email to SMS, a Non Delivery Notification (NDN) is eventually generated.
6. If a matching unique SMS source address was found, then the SMS message is further inspected to see if it is a reply or a notification message. To be a notification message it must be a SUBMIT_SM PDU with a receipted message ID. Otherwise, it is considered to be a reply.
7. If it is a reply then the SMS message is converted to an email message using the origination email information from the historical record.
8. If it is a notification, then the SMS message is converted to an email Delivery Status Notification (DSN) as per RFC 1892-1894. Note that the ESMTP NOTIFY flags (RFC 1891) of the original email message will be honored (For example, if the SMS message is a “success” DSN but the original email message requested only “failure” notifications, then the SMS notification will be discarded).
9. If the destination SMS address matches a SELECT_RE option in a configured gateway profile, then the SMS message is treated as a mobile originated message and converted back to email message as per the PARSE_RE_n rules for that gateway profile. If the conversion fails, then the SMS message is invalid and an error response is returned.

SMS Gateway Server Configuration

This section gives directions on how to set up the SMS Gateway Server for both email-to-mobile and mobile-to-email functionality. This section includes the following topics:

Setting Up Bidirectional SMS Routing

The recommended way to set up bidirectional email and SMS routing between the MTA and SMSC is a three step process:

- Set the SMS Address Prefix – Choose an SMS address prefix. Any prefix may be used, so long as it is ten characters or less.
- Set the Gateway Profile – Reserve the prefix for use with the SMS Gateway Server (by setting the gateway profile).
- Configure the SMSC – Configure the SMSC to route SMS destination addresses to the SMS Gateway SMPP server that start with the prefix. Mobile originated email will have only the prefix. Replies and notifications will have the prefix followed by exactly ten decimal digits.

Set the SMS Address Prefix

The source SMS addresses generated by the MTA SMS channel should be set to match the selected SMS address prefix. Do this by setting the following:

- MTA SMS channel options:

  `USE_HEADER_FROM=0`

  `DEFAULT_SOURCE_ADDRESS=`*`prefix`*

  The first setting causes the channel to not attempt to set the SMS source address from information contained in the email message. The second setting causes the SMS source address to be set (to the selected prefix) when it is not set from any other source.

- Recognize the prefix as an SMS destination address to accept and route to email. Do this by specifying the `SELECT_RE` gateway profile option as follows:

  `SELECT_RE=`*`prefix`*

Set the Gateway Profile

The SMS Gateway Server's gateway profile should then be set to make all relayed SMS source addresses unique. This is the default setting but may be explicitly set by specifying the gateway profile option `MAKE_SOURCE_ADDRESSES_UNIQUE=1`. This will result in relayed SMS source addresses of the form:

`prefix``nnnnnnnnnn`

where *nnnnnnnnn* will be a unique, ten digit decimal number.

Configure the SMSC

Finally, the SMSC should be configured to route all SMS destination addresses matching the prefix (either just the prefix, or the prefix plus a ten digit number) to the SMS Gateway Server's SMPP server. The regular expression for such a routing will be similar to:

`prefix``([0-9]{10,10}){0,1}`

where *`prefix`* is the value of `DEFAULT_SOURCE_ADDRESS`, [0-9] specifies the allowed values for the ten digit number, {10, 10} specifies that there will be a minimum of ten digits and a maximum of ten digits, and {0, 1} specifies that there can be zero or one of the 10-digit numbers.

Enabling and Disabling the SMS Gateway Server

- To enable the SMS Gateway Server, the configutil parameter `local.smsgateway.enable` must be set to the value `1`. Use the following configuration utility command to set it:

  ```
  # configutil -o local.smsgateway.enable -v 1
  ```

- To disable the gateway server, set `local.smsgateway.enable` to the value `0`, using the following command:

  ```
  # configutil -o local.smsgateway.enable -v 0
  ```

Starting and Stopping the SMS Gateway Server

After the SMS Gateway Server is enabled, it may be started and stopped with the following commands:

```
# start-msg sms
```

and

```
# stop-msg sms
```

SMS Gateway Server Configuration File

In order to operate, the SMS Gateway Server requires a configuration file. The configuration file is a Unicode text file encoded using UTF-8. The file can be an ASCII text file. The name of the file must be:

installation-directory/config/sms_gateway.cnf

Each option setting in the file has the following format:

```
option-name=option-value
```

Options that are part of an option group appear in the following format:

```
[group-type=group-name]
option-name-1=option-value-1
option-name-2=option-value-2
...
option-name-n=option-value-n
```

Configuring Email-To-Mobile on the Gateway Server

To implement the email-to-mobile part of two-way SMS, you must configure the following:

- “A Gateway Profile” on page 815
- “An SMPP Relay” on page 816
- “An SMPP Server” on page 817

A Gateway Profile

To configure an email-to-mobile gateway profile, follow these steps:

1. Add a gateway profile to the SMS Gateway Server configuration file.

   To add an option group, use the following format:

   ```
   [GATEWAY_PROFILE=profile_name]
   option-name-1=option-value-1
   option-name-2=option-value-2a
   ...
   option-name-n=option-value-n
   ```

   The length of the gateway profile name, `profile_name` in the preceding format, must not exceed 11 bytes. The name must be the same as the name for the `GATEWAY_PROFILE` channel option in the SMS channel option file. The name is case insensitive. For a list of the valid channel options, see "Available Options" on page 782.

2. Set the gateway profile options (for example, `SMSC_DEFAULT_CHARSET`) to match characteristics of the remote SMSC.

3. Set the other gateway profile options to match the SMS channel's email characteristics.

   A complete description of gateway profile options, see "Gateway Profile Options" on page 828.

4. Set the `CHANNEL` option.

   Set its value to the name of the MTA SMS channel.

   When a notification is sent to email through the gateway, the resulting email message will be enqueued to the MTA using this channel name.

An SMPP Relay

To configure an SMPP Relay, complete the following steps:

1. Add an SMPP relay instantiation (option group) to the SMS Gateway Server's configuration file.

   To add an option group, use the following format:

   ```
   [SMPP_RELAY=relay_name]
   option-name-1=option-value-1
   option-name-2=option-value-2
   ...
   option-name-n=option-value-n
   ```

   Any name may be used for the relay's name. All that matters is that the name not be used for any other SMPP relay instantiation within the same configuration file.

2. Set the LISTEN_PORT option.

   The value used for the SMS channel's SMPP_PORT option must match that used for the relay's LISTEN_PORT option. For the LISTEN_PORT, select a TCP port number which is not used by any other SMPP relay or server instantiation nor by any other server running on the same computer.

3. Set the SERVER_HOST option.

   The relay's SERVER_HOST option should give the host name for the remote SMSC's SMPP server. An IP address may be used in place of a host name.

4. Set the SERVER_PORT option.

   The relay's SERVER_PORT option should give the TCP port for the remote SMSC's SMPP server.

For a complete description of all SMPP relay options, see "SMPP Relay Options" on page 823.

An SMPP Server

To configure an SMPP server, complete the following steps:

1. Add an SMPP server instantiation (option group) to the SMS Gateway Server's configuration file.

   To add an option group, use the following format:

   ```
   [SMPP_SERVER=server_name]
   option-name-1=option-value-1
   option-name-2=option-value-2
   ...
   option-name-n=option-value-n
   ```

   Any name may be used for the server's name. All that matters is that the name not be used for any other SMPP server instantiation within the same configuration file.

2. Set LISTEN_PORT option.

   Select a TCP port number which is not being used by any other server or relay instantiation. Additionally, the port number must not be being used by any other server on the same computer.

   The remote SMSC needs to be configured to route notifications via SMPP to the SMS Gateway Server system using this TCP port.

For a complete description of all SMPP server options, see "SMPP Server Options" on page 826.

Configuring Mobile-to-Email Operation

To configure mobile-to-email functionality, two configuration steps must be performed:

- “Configure a Mobile-to-Email Gateway Profile” on page 818
- “Configure a Mobile-to-Email SMPP Server” on page 819

Note that multiple gateway profiles may use the same SMPP server instantiation. Indeed, the same SMPP server instantiation may be used for both email-to-mobile and mobile-to-email applications.

Configure a Mobile-to-Email Gateway Profile

For mobile origination, a gateway profile provides two key pieces of information: how to identify SMS messages intended for that profile and how to convert those messages to email messages. Note that this profile can be the same one used for email-to-mobile with the addition of the `SELECT_RE` option.

To configure the gateway profile, follow these steps:

1. Add a gateway profile (option group) to the SMS Gateway Server’s configuration file.

   To add an option group, use the following format:

   ```
   [GATEWAY_PROFILE=profile_name]
   option-name-1=option-value-1
   option-name-2=option-value-2
   ...
   option-name-n=option-value-n
   ```

   Any name of 11 characters or less may be used for the profile’s name. All that matters is that it is not already used for another gateway profile within the same configuration file.

2. Set the SELECT_RE option must be specified for each gateway profile.

   The value of this option is an ASCII regular expression with which to compare SMS destination addresses. If an SMS destination address matches the regular expression, then the SMS message is sent through the gateway to email using the characteristics described by the matching profile.

   It is important to note that it is possible to configure multiple gateway profiles which have overlapping sets of SMS addresses (for example, a profile which matches the address 000 and another which matches any other three-digit address). However, so doing should be avoided as an SMS message will be passed off to only one gateway profile: the first one which matches. And, moreover, the order in which they are compared is undefined.

3. Set the CHANNEL option.

   Its value should be the name of the MTA's SMS channel.

For a complete description of all mobile origination options, see "Gateway Profile Options" on page 828.

Configure a Mobile-to-Email SMPP Server

Adding an SMPP server is the same as for the email-to-mobile SMPP server (see "An SMPP Server" on page 817.

The remote SMSC needs to be configured to route SMS traffic to the gateway SMPP server. To do this, the SMS destination address used by the SMSC to route mobile-to-email traffic should be the value set for the gateway profile option SELECT_RE.

For example, if the SMS address 000 is to be used for mobile-to-email traffic, then the SMSC needs to be configured to route traffic for the SMS destination address 000 to the gateway SMPP server. The gateway profile should use the option setting SELECT_RE=000.

Configuration Options

The SMS Gateway Server configuration file options are detailed in this section. The tables that follow list all the available configuration options with a brief description of each. There is a table each for global options, SMPP relay options, SMPP server options, and SMS Gateway Server profile options.

In the subsections that follow, complete descriptions are given for all the available configuration options. The subsections are:

Global Options

The SMS Gateway Server presently has three categories of global options:

- Thread Tuning Options
- Historical Data Tuning
- Miscellaneous

All global options must be specified at the top of the configuration file, before any option groups are specified. Table D-20 lists all global configuration options.

Table D-20 Global Options

Options	Default	Description
DEBUG	6	Selects the types of diagnostic output generated
HISTORY_FILE_DIRECTORY		Absolute directory path for files of historical data
HISTORY_FILE_MODE	0770	Permissions for files of historical data
HISTORY_FILE_ROLLOVER_PERIOD	30 mins	Maximum length of time to write to the same file of historical data
LISTEN_CONNECTION_MAX		Maximum number of concurrent inbound connections across all SMPP relay and server instantiations
RECORD_LIFETIME	3 days	Lifetime of a record in the historical data archive
THREAD_COUNT_INITIAL	10 threads	Initial number of worker threads
THREAD_COUNT_MAXIMUM	50 threads	Maximum number of worker threads
THREAD_STACK_SIZE	64 Kb	Stack size for each worker thread

Thread Tuning Options

Each inbound TCP connection represents an SMPP session. The processing for a session is handled by a worker thread from a pool of threads. When the session processing needs to wait for completion of an I/O request, the worker thread parks the session and is given other work to perform. When the I/O request completes, the session is resumed by an available worker thread from the pool.

The following options allow for tuning of this pool of worker thread processes: THREAD_COUNT_INITIAL, THREAD_COUNT_MAXIMUM, THREAD_STACK_SIZE.

THREAD_COUNT_INITIAL

(integer, > 0) Number of threads to initially create for the pool of worker threads. This count does not include the dedicated threads used to manage the in-memory historical data (2 threads) nor the dedicated threads used to listen for incoming TCP connections (one thread per TCP port/interface address pair the SMS Gateway Server listens on). The default value is for THREAD_COUNT_INITIAL is 10 threads.

THREAD_COUNT_MAXIMUM

(integer, >= THREAD_COUNT_INITIAL) Maximum number of threads to allow for the pool of worker threads. The default value is 50 threads.

THREAD_STACK_SIZE

(integer, > 0) Stack size in bytes for each worker thread in the pool of worker threads. The default value is 65,536 bytes (64 Kb).

Historical Data Tuning

When an SMS message is relayed, the message ID generated by the receiving, remote SMPP server is saved in an in-memory hash table. Along with this message ID, information about the original email message is also saved. Should that message ID subsequently be referenced by an SMS notification, this information may be retrieved. The retrieved information can then be used to send the SMS notification to the appropriate email recipient.

The in-memory hash table is backed to disk by a dedicated thread. The resulting disk files are referred to as “history files”. These history files serve two purposes: to save, in nonvolatile form, the data necessary to restore the in-memory hash table after a restart of the SMS Gateway Server, and to conserve virtual memory by storing potentially lengthy data on disk. Each history file is only written to for HASH_FILE_ROLLOVER_PERIOD seconds after which time it is closed and a new history file created. When a history file exceeds an age of RECORD_LIFETIME seconds, it is deleted from disk.

The following options allow for tuning historical files: HISTORY_FILE_DIRECTORY, HISTORY_FILE_MODE, HISTORY_FILE_ROLLOVER_PERIOD, RECORD_LIFETIME.

HISTORY_FILE_DIRECTORY

(string, absolute directory path) Absolute path to the directory to which to write the history files. The directory path will be created if it does not exist. The default value for this option is:

msg_svr_base/data/sms_gateway_cache/

The directory used should be on a reasonably fast disk system and have more than sufficient free space for the anticipated storage; see "SMS Gateway Server Storage Requirements" on page 836 for storage planning information. Sites are encouraged to change this option to a more appropriate value.

HISTORY_FILE_MODE

(integer, octal value) File permissions to associated with the history files. By default, a value of 0770 (octal) is used.

HISTORY_FILE_ROLLOVER_PERIOD

(integer, seconds) The current history file is closed and a new one created every HASH_FILE_ROLLOVER_PERIOD seconds. By default, a value of 1800 seconds (30 minutes) is used.

RECORD_LIFETIME

(integer, seconds > 0) Lifetime in seconds of a historical record. Records older than this lifetime will be purged from memory; history files older than this lifetime will be deleted from disk. By default, a value of 259,200 seconds (3 days) is used. Records stored in memory are purged in sweeps by a thread dedicated to managing the in-memory data. These sweeps occur every HASH_FILE_ROLLOVER_PERIOD seconds. Files on disk are purged when it becomes necessary to open a new history file.

Miscellaneous

There are two miscellaneous options: DEBUG and LISTEN_CONNECTION_MAX.

DEBUG

(integer, bitmask) Enable debug output. The default value is 6 which selects warning and error messages.

Table D-21 defines the bit values of the DEBUG bitmask.

Table D-21 DEBUG Bitmask

Bit	Value	Description
0-31	-1	Extremely verbose output
0	1	Informational messages
1	2	Warning messages
3	4	Error messages
3	8	Subroutine call tracing
4	16	Hash table diagnostics
5	32	I/O diagnostics, receive
6	64	I/O diagnostics, transmit
7	128	SMS to email conversion diagnostics (mobile originate and SMS notification)
8	256	PDU diagnostics, header data
9	512	PDU diagnostics, body data
10	1024	PDU diagnostics, type-length-value data
11	2048	Option processing; sends all option settings to the log file.

LISTEN_CONNECTION_MAX

(integer, >= 0) The maximum number of concurrent, inbound TCP connections to allow across all SMPP relay and server instantiations. A value of 0 (zero) indicates that there is no global limit on the number of connections. There may, however, be per relay or server limits imposed by a given relay or server instantiation.

SMPP Relay Options

The SMS Gateway Server can have multiple instantiations of its SMPP relay, each with different characteristics chief of which will be the TCP port and interface listened on. Put differently, for each network interface and TCP port pair the SMPP relay listens on, distinct characteristics may be ascribed. These characteristics are specified using the options described in this section.

Each instantiation should be placed within an option group of the form:

```
[SMPP_RELAY=relay-name]
option-name-1=option-value-1
option-name-2=option-value-2
...
option-name-n=option-value-n
```

The string `relay-name` merely serves to differentiate this instantiation from other instantiations.

Table D-22 lists the SMPP relay configuration options.

Table D-22 SMPP Relay Options

Options	Default	Description
LISTEN_BACKLOG	255	Connection backlog for inbound SMPP client connections
LISTEN_CONNECTION_MAX		Maximum number of concurrent inbound connections
LISTEN_INTERFACE_ADDRESS		Network interface for inbound SMPP client connections
LISTEN_PORT		TCP port for inbound SMPP client connections
LISTEN_RECEIVE_TIMEOUT	600 s	Read timeout for inbound connections from SMPP clients
LISTEN_TRANSMIT_TIMEOUT	120 s	Write timeout for inbound connections from SMPP clients
MAKE_SOURCE_ADDRESSES_UNIQUE	1	Make relayed SMS source addresses unique and able to be replied to
SERVER_HOST		Host name or IP address of the SMPP server to relay to
SERVER_PORT		TCP port of the SMPP server to relay to
SERVER_RECEIVE_TIMEOUT	600 s	Read timeout for outbound SMPP server connections
SERVER_TRANSMIT_TIMEOUT	120 s	Write timeout for outbound SMPP server connections

LISTEN_BACKLOG

(*integer, in [0,255]*) Connection backlog allowed by the TCP stack for inbound SMPP client connections. The default value is 255.

LISTEN_CONNECTION_MAX

(*integer, >= 0*) The maximum number of concurrent, inbound TCP connections to allow for this SMPP relay instantiation. Note that this value will be ignored if it exceeds the global `LISTEN_CONNECTION_MAX` setting.

LISTEN_INTERFACE_ADDRESS

(*string, "INADDR_ANY" or dotted decimal IP address*) The IP address of the network interface to listen to for inbound SMPP client connections. May be either the string "`INADDR_ANY`" (all available interfaces) or an IP address in dotted decimal form. (For example, 193.168.100.1) The default value is "`INADDR_ANY`". Clustered HA configurations will need to set this value to correspond to the HA logical IP address.

LISTEN_PORT

(integer, TCP port number) TCP port to bind to for accepting inbound SMPP client connections. Specification of this option is mandatory; there is no default value for this option. Note also that there is no Internet Assigned Numbers Authority (IANA) assignment for this service.

LISTEN_RECEIVE_TIMEOUT

(integer, seconds > 0) Timeout to allow when waiting to read data from an SMPP client. The default value is 600 seconds (10 minutes).

LISTEN_TRANSMIT_TIMEOUT

(integer, seconds > 0) Timeout to allow when sending data to an SMPP client. The default value is 120 seconds (2 minutes).

MAKE_SOURCE_ADDRESSES_UNIQUE

(0 or 1) By default, the SMPP relay will append to each SMS source address a unique, ten digit string. The resulting SMS source address is then saved along with the other historical data. The result is a unique SMS address which may then be replied to by SMS users. The SMPP server will detect this address when used as an SMS destination address and will then send the SMS message to the correct email originator.

To disable this generating of unique SMS source addresses (for one-way SMS), specify a value of 0 (zero) for this option.

SERVER_HOST

(string, TCP hostname or dotted decimal IP address) SMPP server to relay SMPP client traffic to. Either a hostname or IP address may be specified. Specification of this option is mandatory; there is no default value for this option.

SERVER_PORT

(integer, TCP port number) TCP port for the remote SMPP server to which to relay. Specification of this option is mandatory; there is no default value for this option. There is no IANA assignment for this service; do not confuse with the IANA assignment for SNPP.

SERVER_RECEIVE_TIMEOUT

(integer, seconds > 0) Timeout to allow when waiting to read data from the SMPP server. The default value is 600 seconds (10 minutes).

SERVER_TRANSMIT_TIMEOUT

(integer, seconds > 0) Timeout to allow when sending data to the SMPP server. The default value is 120 seconds (2 minutes).

SMPP Server Options

The SMS Gateway Server can have multiple instantiations of its SMPP server, each with different characteristics chief of which will be the TCP port and interface listened on. Put differently, for each network interface and TCP port pair the SMPP server listens on, distinct characteristics may be ascribed. These characteristics are specified using the options described in this section.

Each instantiation should be placed within an option group of the form:

```
[SMPP_SERVER=server-name]
option-value-1=option-value-1
option-value-2=option-value-2
...
option-name-n=option-value-n
```

The string `server-name` merely serves to differentiate the instantiation from other instantiations.

Table D-23 lists the SMPP server configuration options.

Table D-23 SMPP Server Options

Options	Default	Description
`LISTEN_BACKLOG`	255	Connection backlog for inbound SMPP server connections
`LISTEN_CONNECTION_MAX`		Maximum number of concurrent inbound connections
`LISTEN_INTERFACE_ADDRESS`		Network interface for inbound SMPP server connections

Table D-23 SMPP Server Options *(Continued)*

Options	Default	Description
LISTEN_PORT		TCP port for inbound SMPP server connections
LISTEN_RECEIVE_TIMEOUT	600 s	Read timeout for inbound SMPP server connections
LISTEN_TRANSMIT_TIMEOUT	120 s	Write timeout for inbound SMPP server connections

LISTEN_BACKLOG

(*integer in [0,255])* Connection backlog allowed by the TCP stack for inbound SMPP client connections. The default value is 255.

LISTEN_CONNECTION_MAX

(integer >= 0) The maximum number of concurrent, inbound TCP connections to allow for this SMPP server instantiation. Note that this value will be ignored if it exceeds the global LISTEN_CONNECTION_MAX setting.

LISTEN_INTERFACE_ADDRESS

(string, "INADDR_ANY" or dotted decimal IP address) The IP address of the network interface to listen to for inbound SMPP client connections on. May be either the string "INADDR_ANY" (all available interfaces) or an IP address in dotted decimal form. (For example, 193.168.100.1.) The default value is "INADDR_ANY".

LISTEN_PORT

(integer, TCP port number) TCP port to bind to for accepting inbound SMPP client connections. Specification of this option is mandatory; there is no default value for this option. Note that there is no IANA assignment for this service.

LISTEN_RECEIVE_TIMEOUT

(integer, seconds > 0) Timeout to allow when waiting to read data from an SMPP client. The default value is 600 seconds (10 minutes).

LISTEN_TRANSMIT_TIMEOUT

(integer, seconds > 0) Timeout to allow when sending data to an SMPP client. The default value is 120 seconds (2 minutes).

Gateway Profile Options

There may be zero or more gateway profiles. In the SMS Gateway Sever's configuration file, each gateway profile is declared within an option group in the following format:

```
[GATEWAY_PROFILE=profile-name]
option-name-1=option-value-1
option-name-2=option-value-2
...
option-name-n=option-value-n
```

The string `profile-name` merely serves to differentiate the profile from other origination profiles.

Table D-24 lists the SMS Gateway Server profile options.

Table D-24 SMS Gateway Server Profile Options

Options	Default	Description
CHANNEL	sms	Channel to enqueue message as
EMAIL_BODY_CHARSET	US-ASCII	Character set for email message bodies
EMAIL_HEADER_CHARSET	US-ASCII	Character set for email message headers
FROM_DOMAIN		Domain name for routing email back to SMS
PARSE_RE_0, PARSE_RE_1, ..., PARSE_RE_9		Regular expressions for parsing SMS message text
PROFILE	GSM	SMS profile to operate under: GSM, TDMA, or CDMA
SELECT_RE		Regular expression for selecting the plugin
SMSC_DEFAULT_CHARSET	US-ASCII	SMSC's default character set
USE_SMS_PRIORITY	0	Gateway SMS priority flags to email
USE_SMS_PRIVACY	0	Gateway SMS privacy indicators to email

CHANNEL

(string, 1-40 characters) Name of the MTA channel used to enqueue email messages. If not specified, then "sms" is assumed. The specified channel must be defined in the MTA's configuration.

EMAIL_BODY_CHARSET

(string, character set name) The character set to translate SMS text to prior to insertion into an email message's body. If necessary, the translated text will be MIME encoded. The default value is US-ASCII. If the SMS message contains glyphs not available in the charset, they will be converted to mnemonic characters, which may or may not be meaningful to the recipient.

A list of the character sets known to the MTA may be found in the following file:

installation-directory/config/charsets.txt

EMAIL_HEADER_CHARSET

(string, character set name) The character set to translate SMS text to prior to insertion into an RFC 822 Subject: header line. If necessary, the translated string will be MIME encoded. The default value is US-ASCII. If the SMS message contains glyphs not available in the charset, they will be converted to mnemonic characters, which may or may not be meaningful to the recipient

FROM_DOMAIN

(string, IP host name, 1-64 characters) Domain name to append to SMS source addresses when constructing envelope From: addresses for email messages. The name specified should be the correct name for routing email back to SMS. (For example, the host name associated with the MTA SMS channel.) If not specified, then the official host name of the channel specified with the CHANNEL option will be used.

PARSE_RE_0, PARSE_RE_1, ..., PARSE_RE_9

(string, UTF-8 regular expression) For mobile origination of email, the gateway profile needs to extract a destination email address from the text of the SMS message. This is done by means of one or more POSIX-compliant regular expressions (REs). The text of the SMS message will be evaluated by each regular expression until either a match producing a destination email address is found or the list of regular expressions exhausted.

NOTE Use of PARSE_RE_* and ROUTE_TO options are mutually exclusive. Use of both in the same gateway profile is a configuration error.

Each regular expression must be POSIX compliant and encoded in the UTF-8 character set. The regular expressions must output as string 0 the destination address. They may optionally output text to use in a `Subject:` header line as string 1, and text to use in the message body as string 2. Any text not "consumed" by the regular expression will also be used in the message body, following any text output as string 2.

The regular expressions will be tried in the order PARSE_RE_0, PARSE_RE_1, ..., up to PARSE_RE_9. If no regular expressions are specified, then the following default regular expression is used:

```
[ \t]*([^\( ]*)[ \t]*(?:\(([^\)]*\))?[ \t]*(.*)
```

This default regular expression breaks into the following components:

```
[ \t]*
```

Ignore leading white space characters (`SPACE` and `TAB`).

```
([^\( ]*)
```

Destination email address. This is the first reported string.

```
[ \t]*
```

Ignore white space characters.

```
(?:\(([^\)]*)$1\))?
```

Optional subject text enclosed in parentheses. This is the second reported string. The leading `?:` causes the outer parentheses to not report a string. They are being used merely for grouping their contents together into a single RE for the trailing `?`. The trailing `?` causes this RE component to match only zero or one time and is equivalent to the expression {0,1}.

```
[ \t]*
```

Ignore white space characters.

```
(.*)
```

Remaining text to message body. This is the third reported string.

For example, with the above regular expression, the sample SMS message:

```
dan@sesta.com(Testing)This is a test
```

yields the email message:

```
To: dan@sesta.com
Subject: Testing

This is a test
```

As a second example, the SMS message:

```
sue@sesta.com This is another test
```

would yield:

```
To: sue@sesta.com

This is another test
```

Note that the SMS message, prior to evaluation with these regular expressions, will be translated to the UTF-16 encoding of Unicode. The translated text is then evaluated with the regular expressions which were previously converted from UTF-8 to UTF-16. The results of the evaluation are then translated to US-ASCII for the destination email address, `EMAIL_HEADER_CHARSET` for the `Subject:` text, if any, and `EMAIL_BODY_CHARSET` for the message body, if any.

PROFILE

(string, "GSM", "TDMA", or "CDMA") SMS profile to assume. Presently this information is only used to map SMS priority flags to RFC 822 `Priority:` header lines. Consequently, this option has no effect when `USE_SMS_PRIORITY=0` which is the default setting for that option.

SELECT_RE

(string, US-ASCII regular expression) A US-ASCII POSIX-compliant regular expression to compare against each SMS message's SMS destination address. If an SMS message's destination address matches this RE, then the SMS message will be sent through the gateway to email in accord with this gateway profile.

Note that since an SMS message's destination address is specified in the US-ASCII character set, this regular expression must also be expressed in US-ASCII.

SMSC_DEFAULT_CHARSET

(string, character set name) The name of the default character set used by the remote SMSC. The two common choices for this option are US-ASCII and UTF-16-BE (USC2). If not specified, US-ASCII is assumed.

USE_SMS_PRIORITY

(integer, 0 or 1) By default (with `USE_SMS_PRIORITY=0`), priority flags in SMS messages are ignored and not sent with the email messages. To have the priority flags passed with the email, specify `USE_SMS_PRIORITY=1`. When passed with the email, the mapping from SMS to email is as shown in Table D-25:

Table D-25 Priority Flag Mapping from SMS to Email

SMS Profile	SMS Priority Flag	Email Priority: Header Line
GSM	`0` (Non-priority)	No header line (implies `Normal`)
	`1`, 2, 3 (Priority)	`Urgent`
TDMA	`0` (Bulk)	`Nonurgent`
	1 (Normal)	No header line (implies `Normal`)
	2 (Urgent)	`Urgent`
	3 (Very Urgent)	`Urgent`
CDMA	`0` (Normal)	No header line (implies `Normal`)
	1 (Interactive)	`Urgent`
	2 (Urgent)	`Urgent`
	3 (Emergency)	`Urgent`

Note that the email `Priority:` header line values are `Nonurgent`, `Normal`, and `Urgent`.

USE_SMS_PRIVACY

(integer, 0 or 1) By default (with `USE_SMS_PRIVACY=0`), SMS privacy indications are ignored and not sent with the email messages. To have this information passed with the email, specify `USE_SMS_PRIVACY=1`. When passed along with email, the mapping from SMS to email is as shown in Table D-26:

Table D-26 Privacy Flags Mapping from SMS to Email

SMS Privacy Flag	Email Sensitivity: Header Line
`0` (Not restricted)	No header line
`1` (Restricted)	`Personal`
`2` (Confidential)	`Private`
`3` (Secret)	`Company-confidential`

Note that the values of the email `Sensitivity:` header line are `Personal`, `Private`, and `Company-confidential`.

Configuration Example for Two-Way SMS

Assumptions on Behavior

For the sake of this example, let us assume that the following behavior is desired:

- Email messages addressed to

  `sms-id@sms.domain.com`

  are to be sent to the SMS address

  `sms-id`

  and given a unique SMS source address in the range `000`*`nnnnnnnnnn`*.

- Mobile SMS messages addressed to the SMS address `000` are to be sent through the gateway to email with the email address extracted from the start of the SMS message text.

  For example, if the SMS message text is:

  `jdoe@domain.com Interested in a movie?`

  then the message “Interested in a movie?” is to be sent to jdoe@domain.com.

- SMS notifications sent to 000*nnnnnnnnnn* are to be sent through the gateway to email and directed to the originator of the message being receipted.

In order to bring about this behavior, the following assumptions and assignments are made

Further Assumptions and Assignments

- The MTA's SMS channel uses the domain name sms.domain.com.
- The SMS Gateway Server runs on the host gateway.domain.com and uses:
  - TCP port 503 for its SMPP relay
  - TCP port 504 for its SMPP server
- The remote SMSC's SMPP server runs on the host smpp.domain.com and listens on TCP port 377.
- The remote SMSC's default character set is UCS2 (aka, UTF-16).

SMS Channel Configuration

To effect the above behavior, the following SMS channel configuration may be used in the imta.cnf file (add these lines to the bottom of the file):

```
(blank line)
sms
sms.domain.com
```

SMS Channel Option File

The channel's option file, sms_option, would then contain the following settings:

```
SMPP_SERVER=gateway.domain.com
SMPP_PORT=503
USE_HEADER_FROM=0
DEFAULT_SOURCE_ADDRESS=000
GATEWAY_PROFILE=sms1
SMSC_DEFAULT_CHARSET=UCS2
```

SMS Gateway Server Configuration

Finally, the Gateway Server configuration file, sms_gateway.cnf, should look like the following:

```
HISTORY_FILE_DIRECTORY=/sms_gateway_cache/
```

```
[SMPP_RELAY=relay1]
LISTEN_PORT=503
SERVER_HOST=smpp.domain.com
SERVER_PORT=377

[SMPP_SERVER=server1]
LISTEN_PORT=504

[GATEWAY_PROFILE=sms1]
SELECT_RE=000([0-9]{10,10}){0,1}
SMSC_DEFAULT_CHARSET=UCS2
```

Testing This Configuration

If you do not have an SMSC to test on, you may want to perform some loopback tests. With some additional settings in the `sms_option` file, some simple loop back tests may be performed for the above configuration.

Additional sms_option File Settings

The additional settings for the `sms_option` file are:

```
! So that we don't add text to the body of the SMS message
FROM_FORMAT=
SUBJECT_FORMAT=
CONTENT_PREFIX=
```

Without these settings, an email containing:

```
user@domain.com (Sample subject) Sample text
```

would get converted into the SMS message:

```
From:user@domain.com Subject:Sample Subject Msg:Sample text
```

That, in turn, would not be in the format expected by the mobile-to-email code, which wants to see:

```
user@domain.com (Sample subject) Sample text
```

Hence the need (for loopback testing) to specify empty strings for the `FROM_FORMAT`, `SUBJECT_FORMAT`, and `CONTENT_PREFIX` options.

Performing the Loopback Test

Send test email messages addressed to `000@sms.domain.com`, such as:

```
user@domain.com (Test message) This is a test message which should loop back
```

The result is that this email message should be routed back to the email recipient `user@domain.com`.Be sure to have added `sms.domain.com` to your DNS or host tables for the test.

SMS Gateway Server Storage Requirements

To determine the amount of resources you will need for the SMS Gateway Server, use the numbers you generate from the requirements in Table D-27 along with your expected number of relayed messages per second and the RECORD_LIFETIME setting.

Table D-27 covers the requirements for the historical records, the SMPP relay, and SMPP server.

Table D-27 SMS Gateway Server Storage Requirements

Component	Requirements
In-memory historical record	Each relayed message requires 33+m+s bytes of virtual memory, where m is the length of the message's SMS message ID (1 <= m <= 64) and s is the length of the message's SMS source address (1 <= s<= 20). When MAKE_SOURCE_ADDRESS_UNIQUE=0, then only 16+m bytes are used. For 64 bit operating systems, 49+m+s bytes of virtual memory are consumed per record [24+m when MAKE_SOURCE_ADDRESS_UNIQUE=0]. Note also, that the heap allocator may actually allocate larger size pieces of virtual memory for each record. The maximum number of records is 43 billion (2**32-1). For less than 16.8 million records (2**24), the hash table consumes approximately 16 Mb; for less than 67.1 million records (2**26), the hash table consumes approximately 64 Mb; for more than 67.1 million records, the hash table consumes approximately 256 Mb. Double the memory consumptions for 64 bit operating systems. These consumptions are in addition to the memory consumption required for each record itself.

Table D-27 SMS Gateway Server Storage Requirements *(Continued)*

Component	Requirements
On-disk historical record	Each relayed message requires on average the following number of bytes: `81+m+2s+3a+S+2i` where: • `m` is the average length of SMS message IDs, and 1<=m<=64 • `s` is the average length of SMS source addresses, and 1 <= s <= 20 • a is the average length of email addresses, and 3 <= a <= 129 • `S` is the average length of `Subject:` header lines, and 0 <= S<= 80 • `i` is the average length of email message envelope IDs, and 0 <= `i` <= 129 The size for any specific record is influenced by the length of the message's envelope `From:` and `To:` addresses, envelope and message IDs, and the length of the `Subject:` header line. The maximum record length is 910 bytes. When `MAKE_SOURCE_ADDRESS_UNIQUE=0` is used, the size of each record in bytes is: `78+m+3a+S+2i`.
SMPP relay	Each relayed SMPP session consumes two TCP sockets: one with the local SMPP client and another with the remote SMPP server. Approximately 1 Kb of virtual memory is consumed per connection on 32 bit operating systems; 2 Kb on 64 bit operating systems.
SMPP server	Each incoming connection consumes a TCP socket. Approximately 1 Kb of virtual memory is consumed per connection on 32 bit operating systems; 2 Kb on 64 bit operating systems.

For instance, if on average 50 messages per second are expected to be relayed, SMS source addresses are 13 bytes long, SMS message IDs have a typical length of 12 bytes, email addresses 24 bytes, `Subject:` lines 40 bytes, email message and envelope IDs 40 bytes each, and historical data is retained for 7 days, then:

- There will be 30.24 million historical records to store, each on average 58 bytes in memory and 311 bytes long on disk;
- The in-memory consumption of the historical records will be about 1.70 Gb (1.63 Gb + 64 Mb); and
- The on-disk storage will be approximately 8.76 Gb.

While a sufficiency of disk may be supplied to handle any on disk requirements, the virtual memory requirement on a 32-bit machine will be a hard limit of approximately 2 Gb. To reduce the amount of virtual memory or disk storage required, use the `RECORD_LIFETIME` option to reduce the length of time records are retained.

Appendix E

Installation Worksheets

This appendix provides worksheets by which you can plan your installation. The following worksheets are included:

- Directory Server Installation
- Administration Server Initial Runtime Configuration
- Directory Server Setup Script (comm_dssetup.pl)
- Messaging Server Initial Runtime Configuration

Directory Server Installation

You installed Directory Server through the Java Enterprise System installer or through a previous installation. Record your Directory Server installation and configuration parameters in Table E-1 (this is a replica of the worksheet shown in the *Messaging Server Deployment Planning Guide*). You will need these parameters when you install and configure the Administration and Messaging servers.

Table E-1 Directory Server Installation Parameters

Parameter:	Description:	Example:	Used in:	Your Answers:
Directory Installation Root	A directory on the directory server machine dedicated to holding the server program, configuration, maintenance, and information files.	`/var/mps/serverroot/`	`comm_dssetup.pl` Perl script See "To Prepare Directory Server for Messaging Server Configuration" on page 33.	
Host	The host name is the IP host name, which might be either a "short-form" host name (for example, `fiddle`) or a fully qualified host name. The fully qualified host name consists of two parts: the host name and the domain name.	`fiddle.west.sesta.com`	Administration Server Configuration See "To Prepare Directory Server for Messaging Server Configuration" on page 33.	
LDAP Directory Port Number	The default for and LDAP directory server is 389.	389	Administration Server Configuration and Messaging Server Configuration See "To Prepare Directory Server for Messaging Server Configuration" on page 33 and "To Create the Initial Messaging Server Runtime Configuration" on page 43	

Parameter:	Description:	Example:	Used in:	Your Answers:
Administrator ID and Password	Administrator in charge or responsible for configuration information. Password for the Administrator	`Admin` PaSsWoRd	Administration Server Configuration See "To Prepare Directory Server for Messaging Server Configuration" on page 33.	
User and Group Tree Suffix	The distinguished name of the LDAP entry at the top of the directory tree, below which user and group data is stored.	`o=usergroup`	`comm_dssetup.pl` Perl script See "To Prepare Directory Server for Messaging Server Configuration" on page 33.	
Directory Manager DN and Password	The privileged directory administrator, comparable to the superuser user in UNIX. Typically, this administrator is responsible for user and group data. Password for the Directory Manager.	`cn=Directory Manager` `pAsSwOrD`	`comm_dssetup.pl` Perl script and Messaging Server Configuration See "To Prepare Directory Server for Messaging Server Configuration" on page 33 and "To Create the Initial Messaging Server Runtime Configuration" on page 43.	
Administration Domain	A region of administrative control.	`System Lab`	Administration Server Configuration See "To Prepare Directory Server for Messaging Server Configuration" on page 33.	

Administration Server Initial Runtime Configuration

When you run the Administration Server initial runtime configuration program through the Java Enterprise System installer, record your installation parameters in Table E-2 (this is a replica of the worksheet shown in the *Messaging Server Deployment Planning Guide*). You will need some of these parameters for the Messaging Server initial runtime configuration. You might also refer to your "Directory Server Installation" on page 840 checklist to answer certain questions.

Table E-2 Administration Server Initial Runtime Configuration Program Parameters

Parameter	Description	Example	Your Answers:
Fully Qualified Domain Name	Fully Qualified domain for the host machine.	`fiddle.west.sesta.com`	
Server Root Definition	Installation Root of the Administration Server dedicated to holding the server program, configuration, maintenance, and information files.	`/var/mps/serverroot`	
UNIX System User	Certain privileges designated to system users to ensure they have appropriate permissions for the processes they are running.	`inetuser`	
UNIX System Group	The group to which certain UNIX System users belong.	`inetgroup`	
Configuration Directory Server	Host and Port specified during "Directory Server Installation" on page 840.	Host `fiddle.west.sesta.com` Port 389	
Configuration Directory Server Administrator and Password	Administrator ID specified during "Directory Server Installation" on page 840. Password of Administrator ID	`Admin` PaSsWoRd	

Table E-2 Administration Server Initial Runtime Configuration Program Parameters

Parameter	Description	Example	Your Answers:
Administration Domain	A region of administrative control. If you have installed Messaging Server and Directory Server on the same machine, then you should choose the same Administration Domain in "Directory Server Installation" on page 840.	System Lab2	
Administrative Server Port	A unique port number dedicated to the Administration Server.	5555	

Directory Server Setup Script (comm_dssetup.pl)

When you run the Directory Server Setup script (`comm_dssetup.pl`) to prepare Directory Server for Messaging Server configuration, record your installation parameters in Table E-3. You will need some of these parameters for the Messaging Server initial runtime configuration.

Table E-3 `comm_dssetup.pl` Script Parameters

Parameter	Description	Example	Your Answers:
Server Root	Installation Root of the Directory Server dedicated to holding the server program, configuration, maintenance, and information files.	`/var/mps/serverroot/`	
Server Instance	LDAP Directory Server daemon or service that is responsible for most functions. In certain deployments, you might dedicate an instance for maintaining users and groups and maintain a separate instance for configuration.	`slapd-varrius`	
DC Root	If you want to have a two-tree DIT provisioning model (Sun LDAP Schema 1 or Sun ONE LDAP Schema.2 (compatibility mode), the DC Tree mirrors the local DNS structure and is used by the system as an index to the Organization Tree that contain the user and group data entries.	`o=internet`	
User and Group Base Suffix	Top entry in the Organization Tree which holds the namespace for user and group entries.	`o=usergroup`	

Table E-3 `comm_dssetup.pl` Script Parameters *(Continued)*

Parameter	Description	Example	Your Answers:
Directory Manager DN and Password	Administrator who is responsible for the user and group data in the Organization Tree. Should be the same as what was specified in the Java Enterprise System Installer. Password of Directory Manager DN	`cn=Directory Manager` pAsSwOrD	

Messaging Server Initial Runtime Configuration

When you run the Messaging Server initial runtime configuration program, record your installation parameters in Table E-4. You might also refer to your "Directory Server Installation" on page 840 checklist to answer certain questions.

Table E-4 Messaging Server Initial Runtime Configuration Program Parameters

Parameter	Description	Example	Your Answers:
Configuration and Data Directory	Contains all of the Messaging Server configuration files. The *msg_svr_base*/data directory is symbolically linked to this directory.	/var/opt/SUNWmsgsr	
UNIX System User	Certain privileges designated to system users to ensure they have appropriate permissions for the processes they are running. This system user should not be the same user that you specified in the Administration Server Initial Runtime Configuration.	mailsrv	
UNIX System Group	The group to which certain UNIX System users belong.This system group should not be the same group that you specified in the Administration Server Initial Runtime Configuration.	mail	
Configuration Directory LDAP URL, Directory Manager, and Password	Configuration Directory Server, LDAP URL, Bind DN, and Password	ldap://fiddle.west.sesta.com:389 cn=Directory Manager PaSsWoRd	
User and Group Directory LDAP URL, Directory Manager, and Password	User and Group Directory Server, LDAP URL, Bind DN, and Password. It is recommended that you have a separate User and Group directory from your Configuration directory.	ldap://fiddle.west.sesta.com:389 cn=Directory Manager PaSsWoRd	

Table E-4 Messaging Server Initial Runtime Configuration Program Parameters

Parameter	Description	Example	Your Answers:
Postmaster Email Address	Email address of the administrator who will monitor postmaster mail. Address must be a fully qualified address, and must be valid, in that there is a mailbox associated with the address.	`pma@siroe.com`	
Password for Administrator Accounts	Password that will be used for service administrator, user/group administrator, end user administrator privileges as well as PAB administrator and SSL passwords.	`paSSwoRD`	
Default Email Domain	The email default that is used if no domain is specified	`siroe.com`	
Organization Name for Default Email Domain	An organization name under which your organization will reside and is used to construct the organization tree.	For example if your organization name is `Engineering`, all the users for `siroe.com` (your default email domain) will be placed under the LDAP DN `o=Engineering, o=usergroup` Your user and group directory suffix was specified in `comm_dssetup.pl`.	

Glossary

Refer to the Java Enterprise System Glossary (http://docs.sun.com/doc/816-6873) for a complete list of terms that are used in this documentation set.

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